Diseño e implementación de un procesador específico en un FPGA para la ejecución del algoritmo de migración 2D de Kirchhoff.

Ing. Sergio Alberto Abreo Carrillo

Trabajo de investigación presentado como requerimiento parcial para optar al título de:

Magister en Ingeniería Electrónica

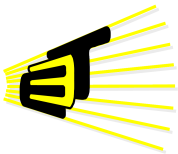
Tesis desarrollada dentro del convenio de cooperación UIS-ICP 005 de 2003

Directora:

PhD(c). Ana Beatríz Ramírez Silva

Co-Director:

PhD. William Mauricio Agudelo Zambrano



Escuela de Ingenierías
Eléctrica, Electrónica
y de Telecomunicaciones

Universidad Industrial de Santander

Facultad de Ingenierías Físico Mecánicas

Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

Febrero, 2011

Diseño e implementación de un procesador específico en un FPGA para la ejecución del algoritmo de migración 2D de Kirchhoff.

Ing. Sergio Alberto Abreo Carrillo

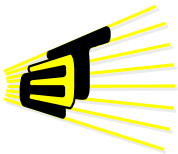
Trabajo de investigación presentado como requerimiento parcial para optar al título de:

Magister en Ingeniería Electrónica

Tesis desarrollada dentro del convenio de cooperación UIS-ICP 005 de 2003

Directora:
PhD(c). Ana Beatríz Ramírez Silva

Co-Director:
PhD. William Mauricio Agudelo Zambrano



Escuela de Ingenierías
Eléctrica, Electrónica
y de Telecomunicaciones

Universidad Industrial de Santander
Facultad de Ingenierías Físico Mecánicas
Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
Febrero, 2011

Agradecimientos

Durante el desarrollo de este trabajo de maestría tuve la oportunidad de contar con muchas personas que de diferentes formas y en distintos momentos hicieron un aporte significativo sobre esta investigación.

En primer lugar quiero agradecerle a la Universidad Industrial de Santander, al grupo de investigación CPS, a su Director el PhD Oscar Gualdrón, al profesor Jorge H. Ramón y en especial a la profesora Ana Beatriz Ramírez por todo su apoyo durante este proceso. Muchas gracias profe Ana por creer en este trabajo y por su gran compromiso con este proyecto.

En el Instituto Colombiano del Petróleo quiero a agradecerle al grupo PETROSÍSMICA, a su director MSc. Andres Calle y en especial al PhD. William M. Agudelo por su apoyo incondicional durante estos tres años. Muchas gracias Doc. William por haber creído en esta propuesta desde sus inicios.

En CPS pude compartir con un grupo de personas que tienen un gran talento humano y que hicieron que esta experiencia fuera muy agradable: Carlos Angulo, Diego Medina, Jose Rugeles, Yadira Felizzola, Jairo Florez, Mauricio Erazo, Daniel Velazco, Julian Rolon y Ricardo Díaz. Muchas gracias muchachos. Adicionalmente quisiera hacer una mención especial a dos compañeros que le hicieron grandes aportes a este trabajo: El profesor MIE. Carlos A. Fajardo A., porque gracias a su metodología de diseño de procesadores específicos y a sus consejos, se pudo elaborar un diseño de calidad. Muchas gracias Don Carlos. Y al profesor MIE William A. Salamanca B. porque gracias a su continua orientación, sus conocimientos de Linux y su gran visión de los sistemas embebidos se logró desarrollar un cluster heterogéneo de gran calidad, el cual elevo el nivel de este proyecto de investigación. Muchas gracias Don William.

En mi familia quisiera agradecerle a mi padre Alberto Abreo Rincón y mi madre Elizabeth Carrillo Rincón porque gracias a su apoyo incondicional, esfuerzo, sacrifico y dedicación he podido alcanzar esta gran meta. Adicionalmente le agradezco a personas como mi tia Martha Elena Abreo Rincón, a su Esposo

Oscar Gomez Zanabria, a mi prima Maria de Jesus Abreo porque siempre me alentaron a soñar y a creer que todo es posible. Quisiera hacer una mención especial a mi tio Elberto Carrillo Rincón porque más que un tio, ha sido un segundo padre y un gran amigo. Gracias tio porque siempre has creído en todas las ideas que se me ocurren y siempre he contado con tu apoyo incondicional y tus grandes consejos.

Finalmente quisiera agradecerle a Dios por darme la fuerza, la salud, la sabiduría y lo más importante de todo por rodearme de un excelente grupo de personas que me ofrecen el apoyo necesario para poder enfrentar cualquier desafío en mi vida. Muchas gracias Dios.

Índice

Lista de Figuras

Lista de Tablas

RESUMEN

TITULO: DISEÑO E IMPLEMENTACIÓN DE UN PROCESADOR ESPECÍFICO EN UN FPGA PARA LA EJECUCIÓN DEL ALGORITMO DE MIGRACIÓN 2D.[1]

AUTOR: SERGIO ALBERTO ABREO CARRILLO[2]

PALABRAS CLAVES: Codiseño, FPGA´s, Migración sísmica, VHDL, HPC.

Este trabajo de investigación se realizó bajo el convenio de cooperación tecnológica UIS-ICP 005 del 2003 y conto con el apoyo de los grupos de investigación CPS y PETROSÍSMICA. El propósito de esta investigación fue evaluar los FPGAs como una alternativa tecnológica para tratar de solucionar problemas de computación de alto rendimiento. El campo de trabajo fue la industria del petróleo en el área de procesamiento de datos sísmicos. Se abordo la migración sísmica por su gran costo computacional. Durante el desarrollo de esta investigación se elaboro un clúster heterogéneo para simular un entorno de trabajo típico y a partir del estudio del proceso de migración sísmica se construyo un procesador específico que permitiera realizar la migración sísmica 2D pre-apilado en profundidad de Kirchhoff. La elaboración del procesador específico se centró en la metodología de Co-diseño, en la cual se busco solucionar el problema mediante la interacción del mismo con el procesador de propósito general Power PC 440. Para ello fue necesario instalar sobre cada uno de los nodos del clúster un sistema operativo (Linux 5.0), la herramienta de Seismic Unix (SU) y desarrollar un driver. Al final del proceso se pudo contrastar el desempeño y la precisión del nodo FPGA con los demás nodos del clúster. Uno de los aportes más importantes de esta tesis de maestría fue que se pudo determinar el desempeño real de los FPGAs cuando se utilizan para la computación de alto rendimiento y se especificó bajo qué condiciones los FPGAs pueden alcanzar su mayor productividad.

[1] Trabajo de Investigación.

[2] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Grupo CPS. Director: PhD(c) Ana B. Ramírez Silva. Codirector: PhD William M. Agudelo Zambrano.

ABSTRACT

TITLE: DESIGNE AND IMPLEMENTATION OF A SPECIFIC PROCESOR ON A FPGA FOR THE EXECUTION OF THE MIGRATION 2D ALGORITHM.[3]

AUTHOR: SERGIO ALBERTO ABREO CARRILLO[4]

KEYWORDS: Co-design, FPGA´s, Seismic Migration, VHDL, HPC

This research work was carried out under the technological cooperation agreement UIS-ICP 005 of 2003 and it counted with the support of the research groups CPS and PETROSISMICA. The purpose of this research was to evaluate the FPGAs as a technological alternative to try to solve problems of High Performance Computing. The field of works was the petroleum industry in the area of seismic data processing. The seismic migration was chosen because of his big computational costs. During the development of this research an heterogeneous cluster was made to simulate an environment of work and from seismic migration process study a specific processor was built that allowed to carried out the 2d pre-stack seismic migration in depth of Kirchhoff. The development of the specific processor was centered in the co-design methodology, in which we sought to resolve the problem through the interaction of it with the general purpose processor Power PC 440. For that it was necessary to install over each node of the cluster an Operative System (Linux 5.0), the Seismic Unix (SU) tool and to develop a driver. At the end of the process we could contrast the performance and precision of the FPGA node with the others nodes of the cluster. One of the most important contributions of this work was that we could determine the real performance of the FPGAs when used for high performance computing and we specified under what conditions the FPGAs can reach its better productivity.

[3]Research work

[4]Physical-Mechanical Engineering Faculty. School of Electric, Electronic and Telecommunications Engineering. CPS group. Director: PhD(c) Ana B. Ramírez Silva. Codirector: PhD William M. Agudelo Zambrano.

Glosario

0.1 Definiciones

Como en el capítulo 1 se va a hablar del procesamiento de datos sísmicos, del proceso de migración sísmica y de la propuesta de investigación, se consideró necesario comenzar aclarando la siguiente terminología:

- Migración: Es un paso en el procesamiento de datos sísmicos en el cual las reflecciones en los datos sísmicos son desplazadas a sus correctas ubicaciones en el espacio x,y,t; incluyendo el tiempo de viaje y la posición relativa a los puntos de disparo (ver figura 1). La migración mejora la interpretación y el mapeo sísmico, porque las ubicaciones de las estructuras geológicas, especialmente las fallas, son más precisas en imágenes migradas.
- Migración de Kirchhoff: Es un método de migración sísmica que usa la forma integral (ecuación de Kirchhoff) de la ecuación de onda. Todos los métodos de migración sísmica suponen que el campo de onda sísmico se propaga hacia abajo desde la región donde este fué medido (normalmente la superficie de la tierra) hasta la región que se quiere ver(dentro de la tierra).
- Migración en tiempo: Es una técnica para procesar datos sísmicos en áreas donde los cambios de velocidad lateral no son muy fuertes, pero las estructuras son complejas.
- Migración en profundidad: Es un paso en el procesamiento sísmico en el cual las reflecciones de los datos sísmicos son desplazados a sus ubicaciones espaciales correctas, teniendo en cuenta la posición relativa a los puntos de disparo, en áreas donde hay variaciones significativas de velocidad (verticales y horizontales) que normalmente distorsionan la imagen obtenida por la migración en tiempo.
- CDP: Punto común de reflección.
- CMP: En adquisiciones sísmicas multicanal, es el punto medio en la superficie entre la fuente y el geófono el cual es compartido por un grupo de parejas fuente receptor.
- Fuentes: Es un dispositivo que provee energía para la adquisición de datos sísmicos tales como una pistola de aire, una carga explosiva ó un vibrador.
- Geófono(receptor): Es un dispositivo que detecta la energía sísmica como un movimiento del suelo ó una onda de presión en un fluido y la transforma en un impulso eléctrico.

- Traza sísmica (traza): Son los datos sísmicos grabados por un canal del geófono. Una traza es una grabación de la respuesta de la tierra a una energía sísmica que viene desde una fuente, atraviesa las diferentes capas del subsuelo y vuelve al geófono.
- Normal moveout(NMO): Es el proceso que compensa la separación entre las fuentes y los receptores sísmicos en el caso de un reflector horizontal (ver figura 2).
- Dip moveout(DMO): Es un proceso similar al NMO solo que ahora compensa el efecto de un reflector buzante (inclinado).
- Apilar(Stack): Se define como la sumatoria de trazas que busca mejorar la relación señal ruido de los datos sísmicos. Normalmente las trazas que se suman son aquellas que se agrupan por CMP; al final después del apilamiento se obtiene una única traza representativa de un conjunto de trazas (ver figura 3).
- interpretación sísmica: Es el análisis de los datos para generar predicciones y modelos razonables acerca de las propiedades y estructuras del subsuelo.

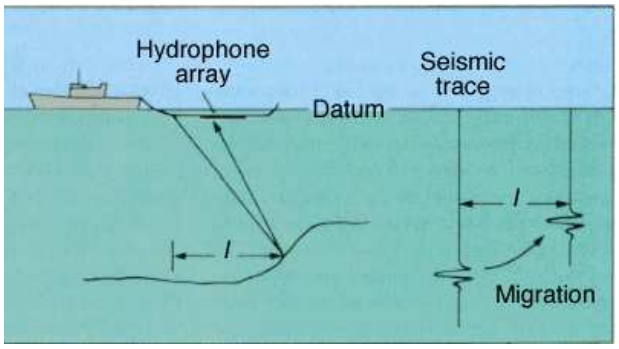


Figura 1: Diagrama de adquisición-migración (adaptada de [1]).

Las definiciones presentadas anteriormente fueron adaptadas de [1].

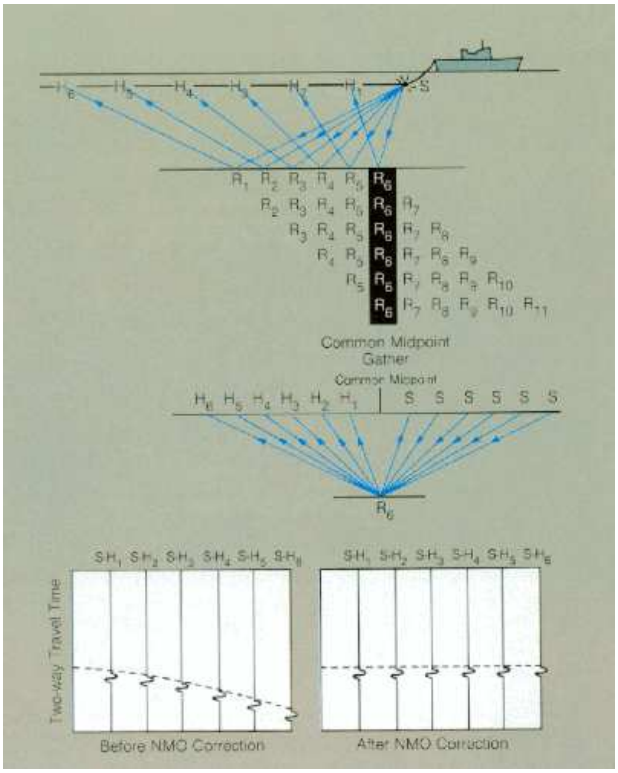


Figura 2: Diagrama de un CMP y NMO (adaptada de [1]).

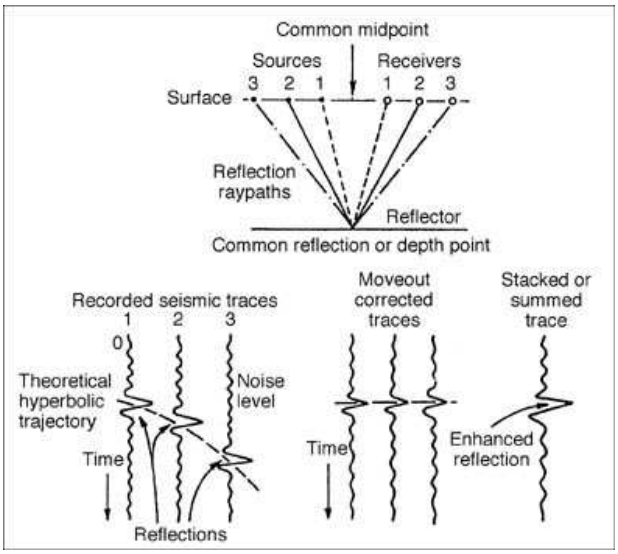


Figura 3: Diagrama del proceso de apilado (adaptada de [1]).

Introducción

Durante la última década las empresas de la industria del petróleo han hecho grandes esfuerzos para mejorar sus técnicas de adquisición y procesamiento de datos sísmicos, debido a que estos esfuerzos se ven reflejados en un incremento de la cantidad y calidad de exploraciones de potenciales regiones con hidrocarburos. A pesar de los avances, la exploración de petróleo continua siendo una actividad económica bastante riesgosa, y es por esto que comúnmente las empresas toman años analizando una región prospecto antes de perforar.

Entre los procesos que requieren mayor costo computacional en las etapas de la exploración sísmica se encuentra la migración sísmica[6]. La migración sísmica se define como el proceso que permite reconstruir la forma del subsuelo a partir del espacio de datos (conjunto de trazas obtenidas en la adquisición sísmica) y otros parámetros de entrada[7]. La importancia de este proceso radica en que éste genera las imágenes sobre las cuales se decide si se realiza o no una perforación en búsqueda de petróleo, dependiendo de la interpretación del geólogo.

El costo computacional de este proceso esta sujeto a factores como el tipo de migración (2D ó 3D), al principio de trabajo de la migración (profundidad, tiempo, corrimiento de fase, etc...) y al momento en el cual se procesan los datos (pre-apilado ó post-apilado). Por esta razón, se encuentran procesos de migración que duran desde un par de días hasta meses[8].

Actualmente las estrategias empleadas para acelerar el proceso de migración son dos: 1. Desde el punto de vista del *hardware*, se utilizan clusters de computadores los cuales son escalados cada cierto tiempo con el fin de ir aumentando su capacidad de procesamiento. 2. Desde el punto de vista del *software*, se realiza la actualización a nuevas versiones de aplicaciones que prometen tener un mejor desempeño que sus versiones anteriores. El propósito de este trabajo consiste en medir el desempeño de una nueva herramienta tecnológica como lo son los FPGAs en la aceleración del proceso de migración sísmica. Para lograr este propósito se elaboró un procesador específico, el cual interactua con el Power PC 440 (procesador de propósito general incorporado en el FPGA) para mejorar el desempeño del FPGA, debido a que combina las ventajas que ofrecen los procesadores de propósito general y las

ventajas que ofrecen los procesadores de propósito específico.

Este libro se organizó de la siguiente forma: El capitulo uno expone un breve marco teórico del proceso de migración sísmica que describe cada una de las etapas contenidas en este proceso y un breve estado del arte en técnicas de computación de la migración sísmica.

El capitulo dos presenta el estudio del proceso de migración sísmica 2D Pre-apilado en profundidad de Kirchhoff, a partir del módulo SUKDMIG2D del *software Seismic Unix SU* de Linux.

El capitulo tres explica la metodología empleada para la realización del *Datapath* y la máquina de estados del procesador específico desarrollado en esta investigación, junto con los diseños finales y cantidad de recursos lógicos utilizados.

En el capitulo cuatro se describen los detalles de la plataforma (Red Okinawa) elaborada para poder medir el desempeño del FPGA cuando realiza el proceso de migración y la forma en la que se hicieron los *drivers* que permiten usar los periféricos desarrollados en *hardware* desde el Sistema Operativo.

Finalmente, en el capitulo cinco se presenta el desempeño obtenido por el FPGA y las conclusiones finales de este trabajo.

Capítulo 1

Marco teórico. Migración Sísmica y su Costo Computacional.

1.1 Adquisición y procesamiento de señales sísmicas.

En la adquisición sísmica se utiliza una fuente para generar vibraciones acústicas o elásticas que viajan dentro de la tierra, pasan a través de estratos con diferentes respuestas sísmicas y efectos de filtrado, y retornan a la superficie para ser grabadas como datos sísmicos por los geófonos. En las Figuras 1.1 y 1.2 podemos ver dos ejemplos de adquisiciones marinas y en la Figura 1.3 podemos ver un conjunto de trazas grabadas por un geófono.

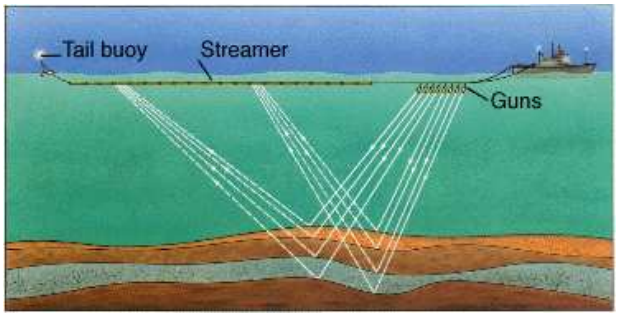


Figura 1.1: Diagrama de una adquisición sísmica marina (tomada de [1]).

Después de obtener las trazas, el siguiente paso en el procesamiento sísmico es trabajar sobre estas grabaciones. Típicamente lo que se hace es filtrar las trazas para mejorar la calidad de las imágenes, hacer un análisis de velocidades y frecuencias, realizar correcciones estáticas, deconvolución, *normal moveout, dip moveout* y finalmente migrar las trazas. Todo este proceso se puede hacer antes y después del apilamiento. En las Figuras 1.4, 1.5 y 1.6 podemos observar un resumen de este proceso. La Figura 1.4 es el modelo desconocido sobre el que se hace la adquisición sísmica; acá se puede apreciar que la apertura de la adquisición es de 4 Km con geófonos ubicados cada 50 metros y la máxima profundidad que se desea analizar es de aproximadamente 2500 metros. En la Figura

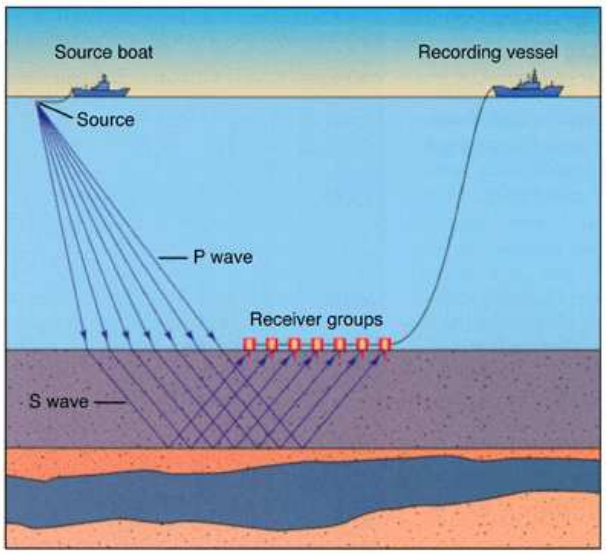


Figura 1.2: Diagrama de otra adquisición sísmica marina (tomada de [1]).

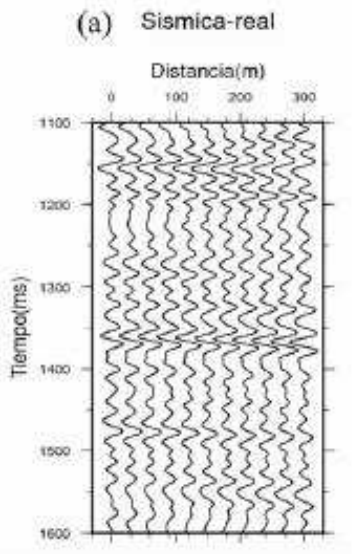


Figura 1.3: Sísmica real (adaptada de [2]).

1.5 se observan las ochenta trazas obtenidas por los geófonos después de la adquisición (estas trazas ya han sido filtradas) con un tiempo de espera de cuatro segundos. Finalmente, en la última gráfica (inferior) vemos la imagen migrada obtenida a partir de las trazas, usando el método en profundidad 2D pre-apilado de Kirchhoff. En la Figura 1.6 se puede ver como la forma del subsuelo (desconocida) empieza a emerger. Entonces, claramente se ve que el objetivo principal del procesamiento sísmico es facilitar el proceso de interpretación resaltando con mayor claridad todos los eventos del subsuelo.

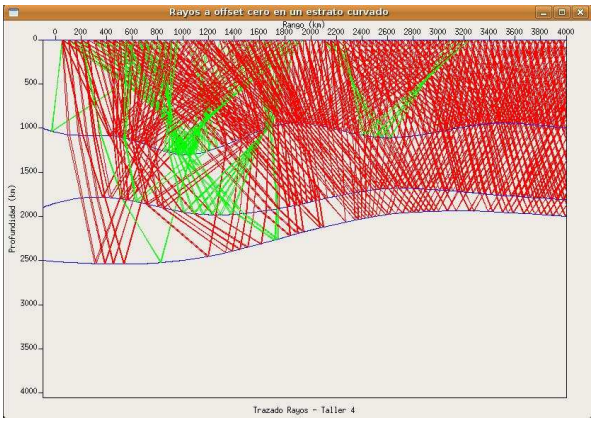


Figura 1.4: Modelo geológico desconocido (Fuente: Autor).

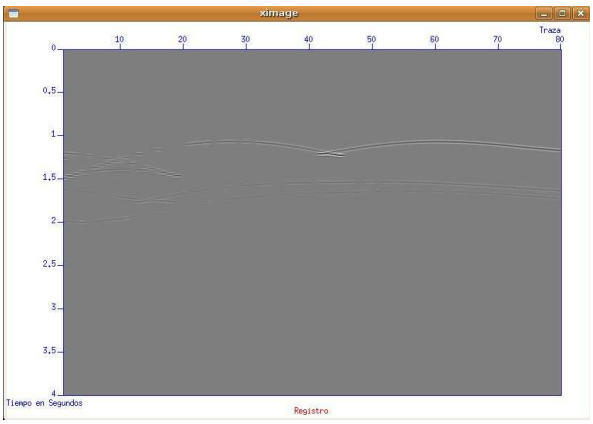


Figura 1.5: Trazas obtenidas por la adquisición (Fuente: Autor).

1.2 Costo computacional

Por otra parte, si se piensa en el costo computacional del procesamiento de datos sísmicos, cada una de las etapas mencionadas anteriormente tienen un gran costo computacional, pero la etapa que más se destaca por este aspecto es la migración sísmica [8]. La causa de su gran costo computacional generalmente gira entorno a muchos factores pero principalmente se pueden referenciar dos:

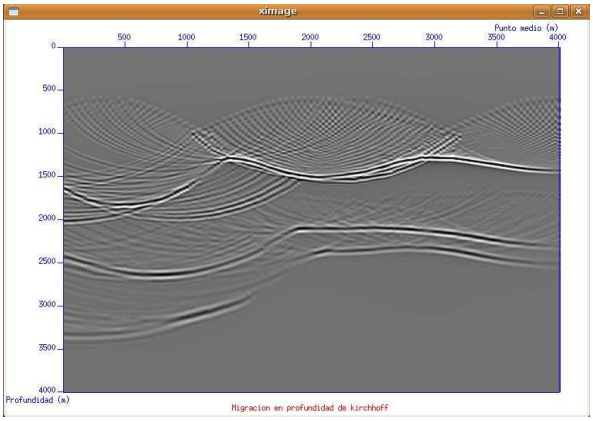


Figura 1.6: Trazas migradas (Fuente: Autor).

- Todos los métodos de migración tienen una componente matemática muy fuerte, es decir realizan ciclos anidados de operaciones como la radicación, logaritmación, potenciación, etc.. sobre datos en formato de punto flotante.
- El volumen de datos que se deben procesar normalmente son del orden de Gigabytes[8].

La razón del elevado volumen de datos que se deben procesar, generalmente está sujeta a características de la adquisición como: La apertura (el área del suelo que se esta estudiando), la resolución de la malla (La cantidad de geófonos que se usaron en la adquisición) y la profundidad que se desea explorar (tiempo de adquisición). Estas características dependen del tipo de suelo que se está estudiando y de la información que se quiera extraer del mismo. En cuanto al método de migración que se debe usar para procesar las trazas obtenidas, también está sujeto a las características del subsuelo. Para poder entender mejor esta dependencia se presenta el siguiente ejemplo. Supongamos que se desea procesar un conjunto de trazas obtenidas de una adquisición sísmica 2D y hay cuatro posibles escenarios geológicos (ver Figura 1.7) de los cuales pudieron ser obtenidas(los escenarios se numeran de izquierda a derecha comenzando por la fila superior). Si el escenario de donde provienen las trazas fué:

- El primero: Como el modelo geológico tiene una estructura simple (capas son casi horizontales y paralelas entre si) y el modelo de velocidades es simple (velocidad aumenta en función de la profundidad), el método de migración que se debería usar es **post**-apilado en **tiempo**[3].
- El segundo: Como el modelo geológico tiene una estructura compleja (capas no horizontales) y el modelo de velocidades es simple, el método de migración que se debería usar es **pre**-apilado en **tiempo**, el cuál tiene un mayor costo computacional que el anterior método[3].
- El tercero: Como el modelo geológico tiene una estructura simple y el modelo de velocidades es complejo (velocidad no aumenta en función de la profundidad, sino se intercalan capas de alta velocidad con baja velocidad), el método de migración que se debería usar es **post**-apilado en **profundidad**, el cuál tiene un mayor costo computacional que los métodos anteriores[3].

- El Cuarto: Como el modelo geológico tiene una estructura compleja y el modelo de velocidades es complejo, el método de migración que se debería usar es **pre**-apilado en **profundidad**, el cuál tiene un mayor costo computacional que todos los métodos anteriores[3].

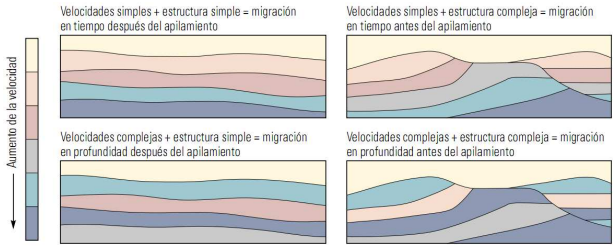


Figura 1.7: Alternativas de subsuelos (Tomada de [3]).

De esta forma podemos entender porque el proceso de migración puede durar desde un par de días hasta unos cuantos meses[8].

1.3 Estado del arte.

Actualmente los diferentes métodos de migración sísmica se implementan utilizando cluster de computadores ó supercomputadores (dependiendo de la cantidad de dinero que tenga la empresa para invertir en recursos de procesamiento), y la metodología empleada es tratar de descomponer el proceso global en un conjunto de sub-procesos, de tal forma que se puedan trabajar de forma independiente (paralelizar) y al final se puedan agrupar los resultados obtenidos.

Para entender un poco mejor la metodología vamos a revisar el trabajo [4], el cual refleja las ideas presentadas anteriormente. En este trabajo, ellos implementaron el proceso de migración 2D pre-apilado en tiempo de Kirchhoff, utilizando un cluster *Beowulf* (Se define como un cluster conformado por computadores de escritorio). La estructura de este cluster se puede ver en la figura 1.8. En él hay un nodo maestro encargado de controlar a los demás nodos trabajadores por medio de una interfaz de paso de mensajes (MPI) y su sistema operativo es Linux.

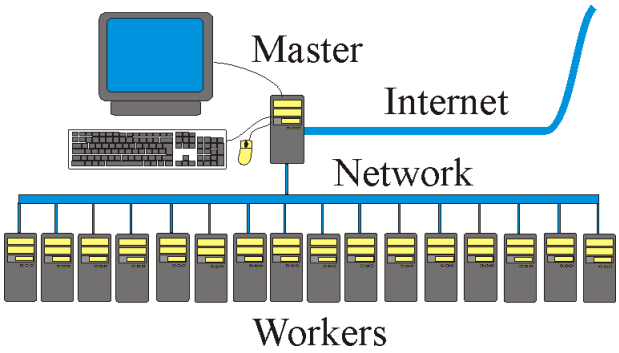


Figura 1.8: Estructura esquemática del Cluster Beowulf (Adaptada de [4]).

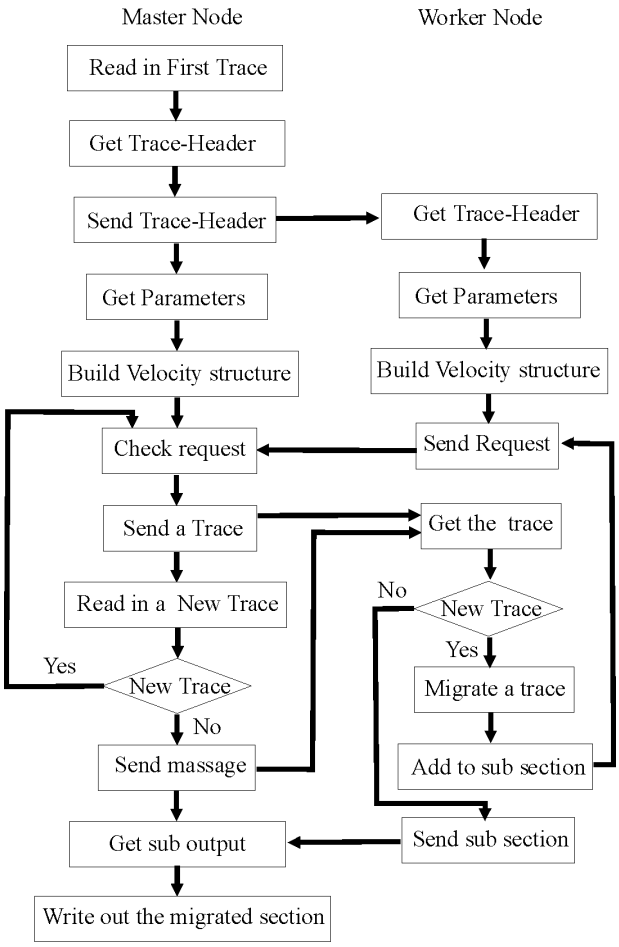


Figura 1.9: Estructura del código paralelizado (Adaptada de [4]).

La forma en la que segmentaron el proceso de migración se muestra en la figura 1.9. En esta figura podemos ver que el nodo maestro es el encargado de tomar cada una de las trazas y enviar los cabeceros de las mismas a cada uno de los nodos donde se van a procesar. Después de que los nodos trabajadores reciben el cabecero, proceden a extraer los parámetros de entrada y a preparar el entorno para iniciar el proceso de migración. Cuando un nodo trabajador esta listo, le envía una solicitud al nodo maestro para que éste le envíe la traza que debe procesar y de esta forma poder iniciar con el proceso de migración. Esta acción se repite continuamente en cada uno de los nodos trabajadores hasta migrar todas las trazas. Cuando se acaban las trazas que se deben procesar, el nodo maestro envía una señal de finalización y de esta forma cada uno de los nodos esclavos comienzan a enviarle los resultados parciales al nodo maestro, el cual finalmente los agrupa y forma la imagen final migrada. Todo esto es posible debido a que el proceso de migración es una aplicación que se deja segmentar.

Tomando como referencia esta visión global de la implementación del proceso de migración en las empresas de la industria del petróleo y el desempeño de los FPGAs como una tecnología emergente en las aplicaciones de computación de alto rendimiento [6], se plantearon los objetivos de este trabajo de investigación.

1.4 Propuesta de investigación.

1.4.1 Objetivo General

Diseñar e implementar un procesador específico en un *Field Programmable Gate Array* (FPGA) que permita la ejecución del algoritmo de migración 2D usado en la sísmica exploratoria sobre un conjunto de datos sintéticos.

1.4.2 Objetivos Específicos

Para poder alcanzar el objetivo general se propusieron los siguientes objetivos específicos:

- Identificar las características más importantes del algoritmo de migración sísmica 2D pre-apilado en profundidad de Kirchhoff.
- Implementar los módulos de las operaciones de punto flotante de acuerdo a las necesidades del algoritmo de migración sísmica 2D pre apilado en profundidad de Kirchhoff.
- Elaborar el procesador específico que permita realizar el proceso de migración sísmica 2D pre apilado en profundidad de Kirchhoff.
- Medir el desempeño del procesador específico usando un grupo de datos sintéticos.

De esta forma finalizamos el primer capitulo del libro, para dar inicio al estudio del módulo SUKDMIG2D de la herramienta Seismic Unix de la Escuela de Minas de Colorado.

Capítulo 2

Estudio del proceso de migración sísmica 2D pre-apilado en profundidad de Kirchhoff, a partir del código de Seismic Unix.

2.1 Introducción

Seismic Unix es un proyecto que nació en el *Center For Wave Phenomena*(CWP) en el año de 1987. Desde sus inicios este proyecto estuvo respaldado por *Gas Research Institute*(GRI), por la *Society of Exploration Geophysicists*(SEG) y por el consorcio internacional CWP(conformado por 23 compañías de la industria del petróleo asi como varias agencias del gobierno). El objetivo de este proyecto fué elaborar una herramienta *Software*(llamada SU) que permitiera hacer de forma sencilla modelamiento sísmico, *Imaging* y métodos de inversión así como mejorar la precisión y eficiencia de algoritmos de procesamiento sísmico, especialmente para aplicaciones con regiones de estructura compleja [9].
En la figura 2.1 se muestran los núcleos que componen esta herramienta. Como cada uno de estos núcleos esta conformado por un conjunto de funciones y el propósito de este capitulo es darle un soporte teórico a este trabajo de investigación, solo detallaremos las funciones que se usaron en este trabajo. Estas son *Cshot*(proviene del módulo de modelamiento sísmico) y *SUKDMIG2D*(proviene del módulo de procesamiento sísmico).

2.2 Módulo Cshot

Este módulo fue creado para generar datos sintéticos de prueba, los cuales luego son usados para entender las diferentes etapas que conforman el *imaging* y para validar nuevas técnicas de procesamiento de datos sísmicos. El módulo está conformado por dos funciones llamadas *Cshot1* y *Cshot2*. La función *Cshot1* permite generar modelos geológicos con la forma y el número de capas que el usuario requiere, mientras que la función *Cshot2* permite simular adquisiciones sísmicas sobre dichos modelos geológicos, entregando el sismograma listo para iniciar con las etapas de procesamiento.

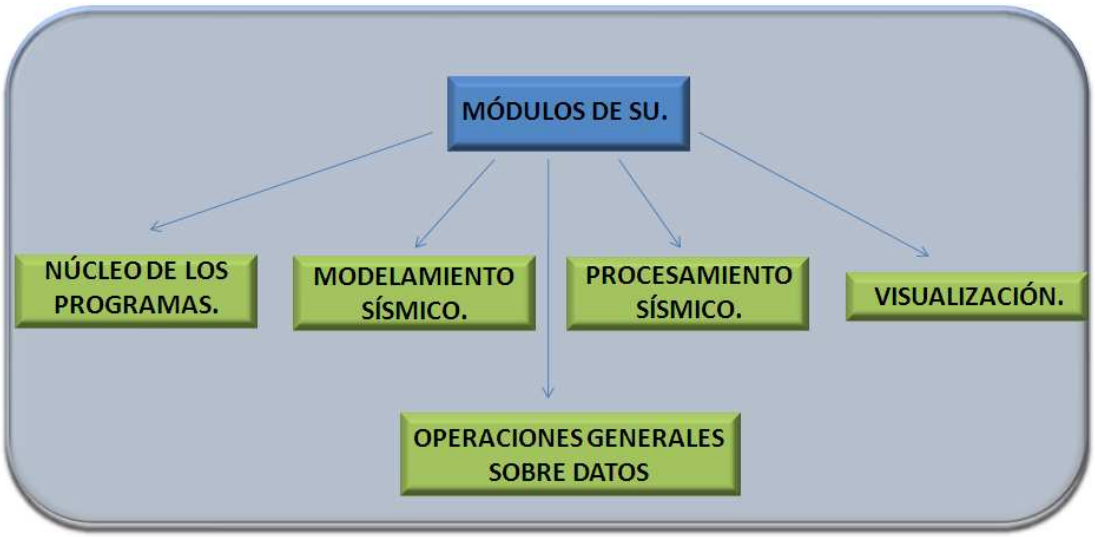


Figura 2.1: Estructura de SU.

Después de configurar la función cshot1 como se presenta en el apéndice A, el modelo geológico que se obtuvo junto con la geometría de la adquisición fue el presentado en la Figura 1.4. Una vez obtenido el modelo se procedió a configurar la función cshot2 como se explica en el apéndice A y las trazas obtenidas fueron las que se mostraron en la Figura 1.5. De esta forma se finalizó la etapa de generación de los datos sintéticos.

2.2.1 Variación de parámetros en la adquisición simulada.

Para aclarar el funcionamiento del módulo *Cshot* se hicieron diferentes pruebas, en las cuales se incrementó el número de geófonos por adquisición. Para lograr esto se modificaron algunos de los archivos mencionados en el apéndice A. A continuación se listan las modificaciones realizadas para la segunda prueba en la que se usaron 800 geófonos:

- Para poder simular una adquisición de 800 geófonos se modificaron los siguientes archivos:

  En el archivo geometría se modificó en el tercer renglón, el cuarto parámetro de izquierda a derecha por un 10 como se puede ver en 2.1. Indicando ahora que por cada disparo en lugar de dos geófonos habrán diez.

  Adicionalmente se modificó en el archivo param2, en el tercer renglón, el segundo parámetro de izquierda a derecha por un diez (ver 2.2) recordandole al *Cshot2* que ahora se deben procesar 10 trazas por cada disparo.

  Finalmente la línea 21 del archivo *Xcshot* se modificó como se puede ver en 2.3. Con esta modificación lo que se hizo fue actualizar los cabeceros de la nueva adquisición, debido a que ahora las trazas se deben agrupar de a diez y se deben incrementar internamente en uno pero externamente en cincuenta.

```
1 1          50.                        :reference station number and x-coord.
2 10.         0.                         :station spacing and receiver depth
3 1   2   2   10        1.    0.          :shot 1 - r1 r2 r3 r4 s sdepth
```

```
4 80 50
```

Script 2.1: geometria para la segunda prueba

```
1 s               :job option(s,r)
2 1 81      :first, last shot for sort
3 1 1             :first, last trace OR first last receiver
4 10. 25. 35. 50. :frequency espectrum of wavelet
5 .050            :wavelet length (secs)
6 0.004             :sample rate (secs)
7 4.      :record length (secs)
8 demoshot        :input filename
9 trazas          :output filename
```

Script 2.2: Param2 para la segunda prueba

```
1 #! /bin/sh
2
3 # Parametros del tamano del modelo
4 x1beg=0 x1end=3000 x2beg=-100 x2end=4000
5
6 #cshot1 calcula los tiempos de llegada de los rayos
7 cshot1 |
8 cshotplot >demoplot  outpar=demopar
9
10 xgraph <demoplot par=demopar windowtitle="Rayos a offset cero en un estrato
11         curvado"\
12         -geometry 600x400+500+500 \
13         title="Trazado Rayos - Taller 4" \
14         label1="Profundidad (km)"  label2="Rango (km)" \
15         x1beg=$x1beg x1end=$x1end x2beg=$x2beg x2end=$x2end &
16
17 #cshot2 genera las trazas (datos) que van a ser migrados
18 cshot2
19 suaddhead <trazas ftn=1 ns=1001 |
20 #sushw key=dt,cdp,sx,gx a=4000,1,0,0 b=0,1,50,50 >registro.su #80 geófonos
21 sushw key=dt,cdp,sx,gx a=4000,1,0,0 b=0,0,0,10 c=0,1,50,50 j=1,1,10,10 >
22 registro.su #800 geófonos
23 suximage <registro.su title="Registro" \
24         xbox=50 ybox=75 \
25         wbox=600 hbox=400 \
26         label1="Tiempo en Segundos" label2="Traza" &
```

Script 2.3: Xcshot para la segunda prueba

2.3 Módulo SUKDMIG2D de SU

Para poder entender como funciona el módulo *SUKDMIG2D* de SU, lo primero que se hizo fue un script de prueba encargado de utilizarlo, con el propósito de familiarizarnos con los resultados que se obtienen después de hacer una

migración sísmica. Pero al consultar el código fuente de este módulo (que se puede ver en el apéndice B.1) nos dimos cuenta que necesitaba adicionalmente a las trazas sísmicas generadas por el comando *Cshot*, las tablas de los tiempos de viaje del modelo geológico sobre el que se hizo la adquisición. Como esta adquisición fue simulada, entonces tuvimos que generar unas tablas de tiempo de viaje a partir de los datos sintéticos y para ello usamos los siguientes comandos:

- *unif2*: Genera un perfil de velocidades uniformemente muestreado desde un modelo de capas. En cada capa, la velocidad es una función lineal de la posición.
- y *rayt2d*: Calcula las tablas de los tiempos de viaje a partir de un trazado de rayos 2D paraxial. Necesita como parámetro de entrada un perfil de velocidades.

Después de que se configuraron los parámetros de uso de los dos comandos mencionados anteriormente y del módulo SUKDMIG2D, se realizo el Script 2.4 para ejecutar de forma automática la migración sísmica 2D en profundidad de Kirchhoff. Este Script básicamente llama al Script *Xcshot* (ver 2.5) para construir el modelo geológico sintético y simular la adquisición sísmica; luego genera las tablas de los tiempos de viaje y finaliza con la migración sísmica. Los parámetros de configuración del trazado de rayos y de la migración se presentan en los scripts 2.6 y 2.7. Luego de ejecutar el Script 2.4 las imágenes que se obtuvieron fueron las presentadas en las figuras 1.4, 1.5 y 1.6.

```
1  #! /bin/sh
2  # shell for uniformly sampling velocity from a layered model
3  set -v
4  . /etc/profile
5  cd /compartido/comun/home/sergio/script-tesis/
6
7  WIDTH=400
8  HEIGHT=600
9  WIDTHOFF1=10
10 WIDTHOFF2=430
11 WIDTHOFF3=860
12 HEIGHTOFF1=20
13 # generacion del sismograma
14 sh Xcshot
15
16 # generacion del modelo de velocidades
17
18 nz=41 dz=50 fz=.0  labelz="Depth (m)"
19 nx=81 dx=50 fx=0.0  labelx="Distance (m)"
20 ninf=0 npmax=201
21 unif2 <input >vfile  ninf=$ninf  npmax=$npmax \
22         nz=$nz dz=$dz fz=$fz nx=$nx dx=$dx fx=$fx \
23         v00=2000
24
25 # trazado de rayos
26
27 rayt2d <vfile par=rayt2d.par
28
29 #Migracion pre-apilada en profundidadd
30
```

```
sukdmig2d<registro.su  par=kdmig.par>kd.data

# Visualizando la migracion y el modelo

nt=501 dt=0.004 ft=0.0  nx=81 dx=50 fx=0
suximage < registro.su d1=$dt f1=$ft d2=$dx f2=$fx  perc=99.5\
        label1="Tiempo (seg)" label2="Punto medio (m)" grid1=solid \
        hbox=$HEIGHT wbox=$WIDTH xbox=$WIDTHOFF2 ybox=$HEIGHTOFF1 \
        title="Datos sinteticos" &

suximage < kd.data perc=99.5\
        label1="Profundidad (m)" label2="Punto medio (m)"\
        d1num=500\
        hbox=$HEIGHT wbox=$WIDTH xbox=$WIDTHOFF3 ybox=$HEIGHTOFF1 \
        title="Migracion en profundidad de kirchhoff" &

suxwigb < kd.data style=seismic perc=80 \
        label1="Profundidad (m)" label2="Punto medio (m)"\
        title="Migracion en profundidad de kirchhoff" &


suxcontour < kd.data label1="Profundidad (m)" label2="Punto medio (m)"\
        title="Migracion en profundidad de kirchhoff" &

exit 0
```

Script 2.4: script migpretiem

```
#! /bin/sh

# Parametros del tamano del modelo
x1beg=0 x1end=3000 x2beg=-100 x2end=4000

#cshot1 calcula los tiempos de llegada de los rayos
#Ver param1 que contiene los parametros del trazado de rayos
#Ver Modelo que tiene la geometria del reflector
cshot1 |
cshotplot >demoplot  outpar=demopar

xgraph <demoplot par=demopar windowtitle="Rayos a offset cero en un estrato curvado"\
        -geometry 600x400+500+500 \
        title="Trazado Rayos - Taller 4" \
        label1="Profundidad (km)"  label2="Rango (km)" \
        x1beg=$x1beg x1end=$x1end x2beg=$x2beg x2end=$x2end &

#cshot2 genera las trazas (datos) que van a ser migrados
cshot2
suaddhead <trazas ftn=1 ns=1001 |
sushw key=dt,cdp,sx,gx a=4000,1,0,0 b=0,0,0,10 c=0,1,50,50 j=1,1,10,10>registro.su

suximage <registro.su title="Registro" \
        xbox=50 ybox=75 \
        wbox=600 hbox=400 \
```

```
26          label1="Tiempo en Segundos" label2="Traza" &
```

Script 2.5: script Xcshot

```
1
2  dt=0.004    nt=501
3  fz=0 nz=41 dz=50
4  fx=0 nx=81 dx=50
5  fxs=0 nxs=81 dxs=50
6  aperx=2000
7  fa=-75 na=76 da=2 amax=75
8  fac=0.01 ms=10 ek=1 npv=0
9  jpfile=jpfile.ray
10 tfile=tfile
```

Script 2.6: script rayt2d.par

```
1
2  fzt=0 nzt=41 dzt=50
3  fxt=0 nxt=81 dxt=50
4  fs=0 ns=81 ds=50
5  aperx=1000 dxm=5
6  off0=1 noff=1
7  offmax=1.5
8  offmax=4000 angmax=60 fmax=55
9  v0=2000 dvz=0
10 jpfile=jpfile.kd
11 ttfile=tfile
```

Script 2.7: script kdmig.par

Después de que se verificó el correcto funcionamiento del módulo de migración, se procedió con el estudio a fondo del código fuente del mismo.

2.3.1 Estructura interna del módulo

Al revisar el código fuente del módulo de migración encontramos que está compuesto por las cinco funciones que se muestran en la figura 2.2.

La interpretación física de cada una de estas funciones es:

- **Resit**: Calcula el tiempo de vuelo residual.
- **Sum2**: Encargada de hacer la ponderación de los datos.
- **Timeb**: Calcula el coseno del ángulo incidente, lateral slowness y el ángulo emergente.
- **MIG2D**: Realiza la migración en profundidad de Kirchhoff.

    Filt: Encargado de realizar el filtrado pasa-bajas. Invocado por la función MIG2D.

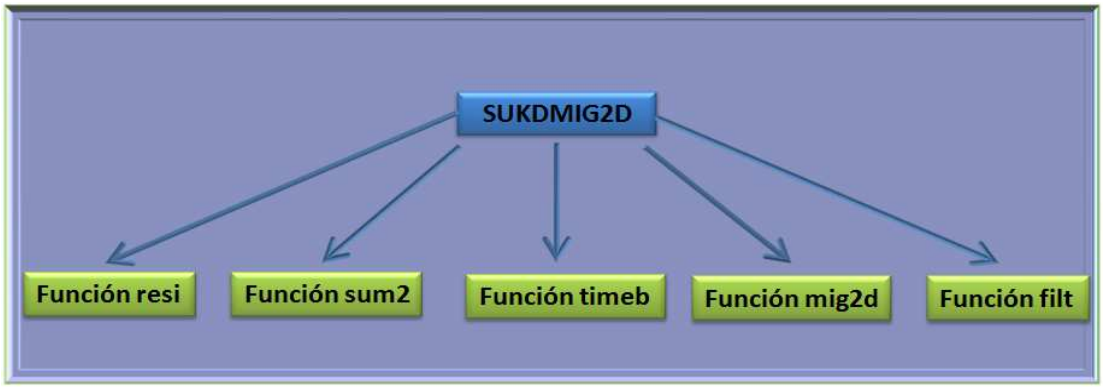


Figura 2.2: Estructura de SUKDMIG2D.

Por otra parte, como no todos los segmentos de código que se mapean dentro de los recursos lógicos del FPGA al final logran alcanzar un índice de aceleración; es necesario identificar aquellos que tengan un elevado costo computacional y que adicionalmente sean usados muchas veces dentro del proceso global.
Con esto en mente, se hizo una revisión detallada de cada una de las funciones mencionadas anteriormente arrojando al proceso de filtrado como el principal candidato a ser mapeado dentro de los recursos lógicos del FPGA. Luego de elegir el proceso que debería ser implementado en los recursos lógicos del FPGA, se profundizó aún más en su código fuente y su composición interna se resume en la figura 2.3

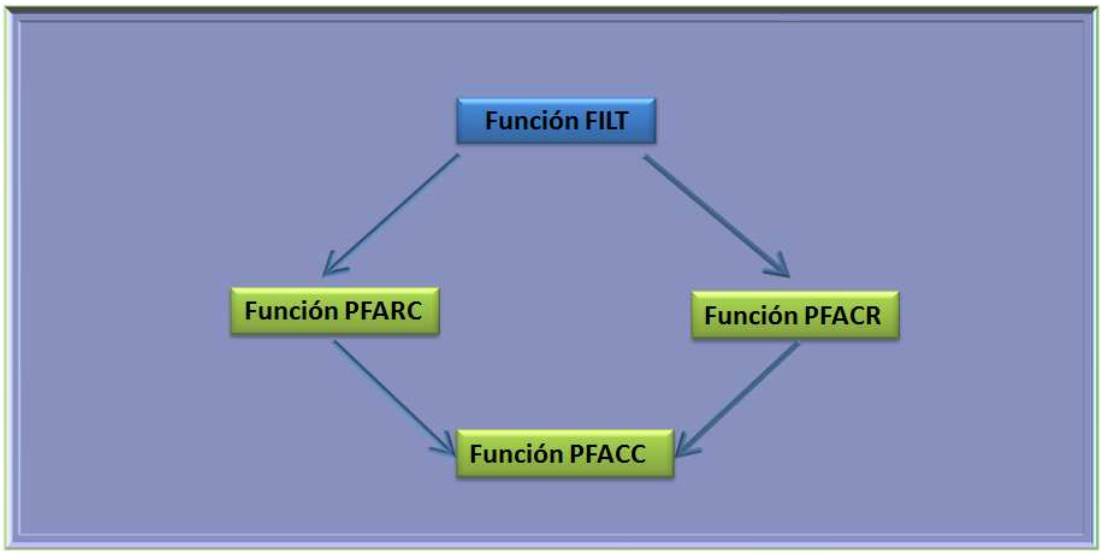


Figura 2.3: Estructura de FILT.

El sentido matemático de cada una de estas funciones es:

- **Pfarc**: Calcula la DFT usando el método de los factores primos en 1D de real a complejo. Usa Pfacc.
- **Pfacr**: Calcula la DFT usando el método de los factores primos en 1D de complejo a real. Usa Pfacc.

- **Pfacc**: Centro de los cálculos de la DFT.

Y los diagramas de dependencia de datos de cada una de estas funciones se muestran en las figuras 2.4, 2.5, 2.7 y 2.6.

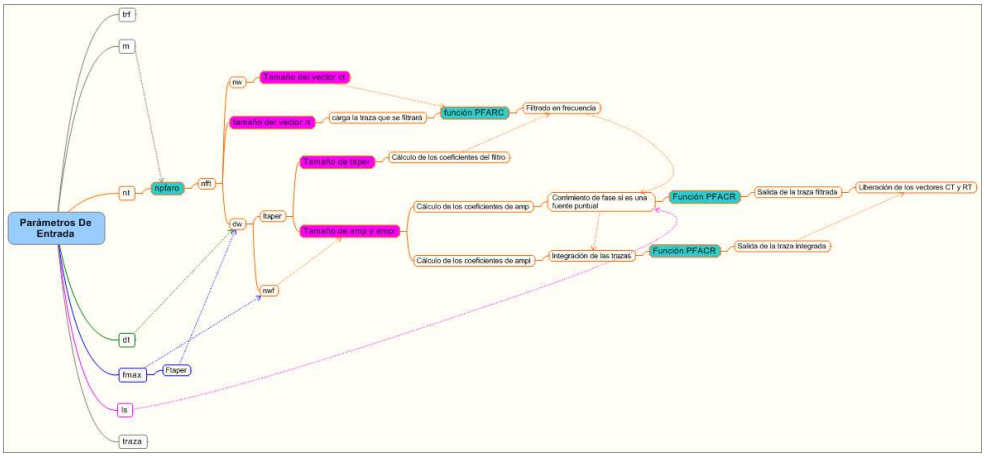


Figura 2.4: Diagrama de la función filt.

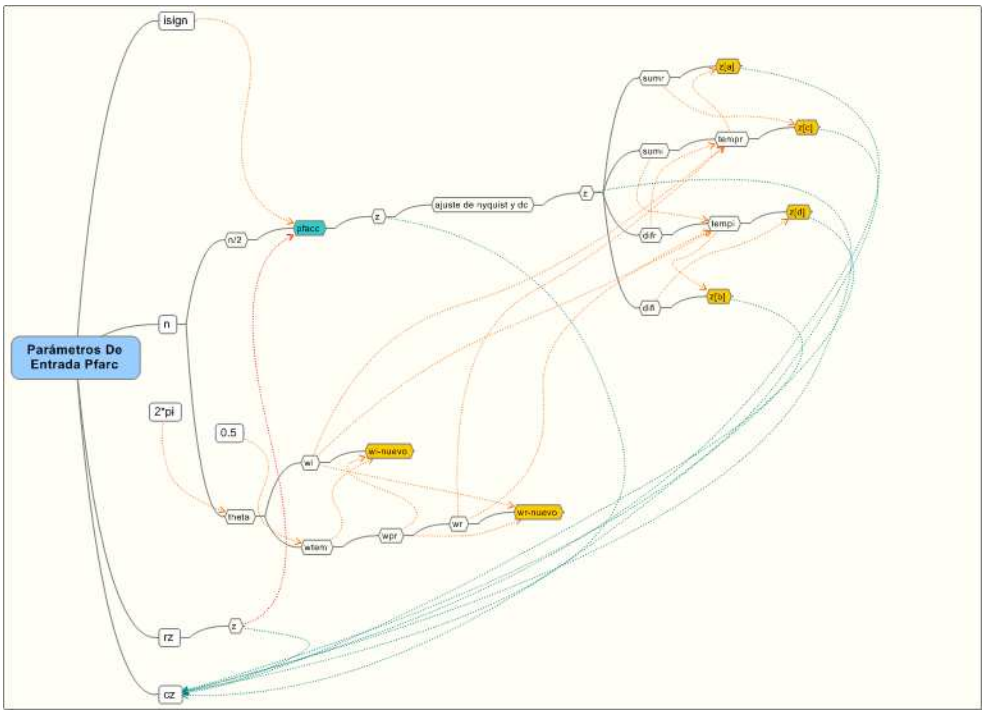


Figura 2.5: Diagrama de la función Pfarc.

El objetivo de hacer los diagramas de dependencias de datos fue tratar de resaltar de una manera visual los niveles de anidamiento de los procesos, la complejidad de los mismos y la dependencia de los datos.

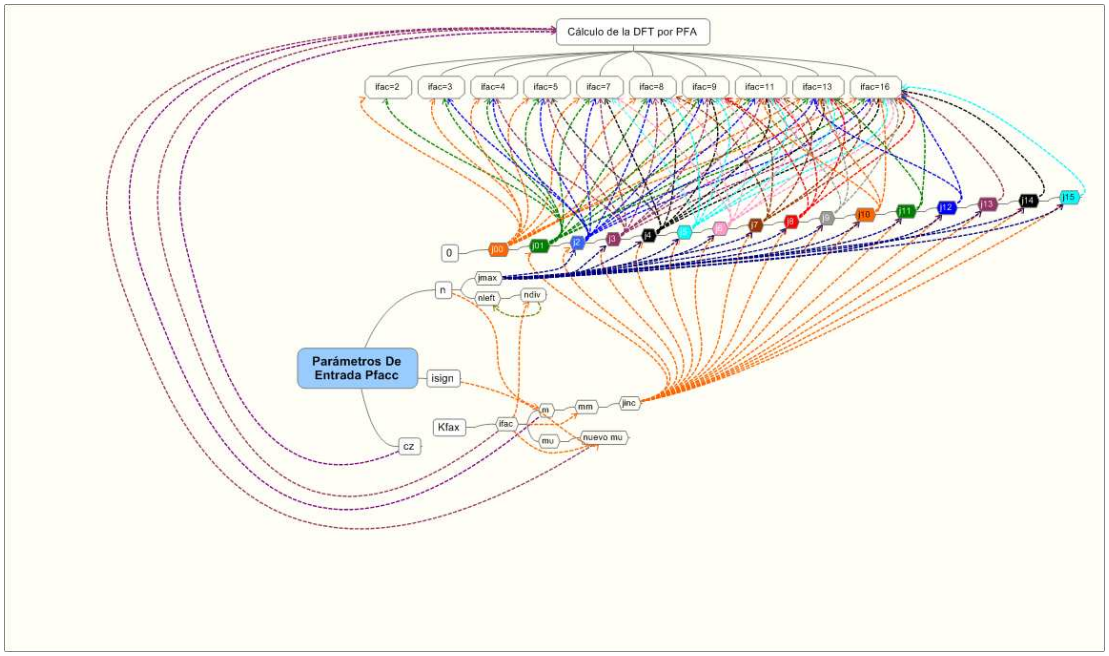


Figura 2.6: Diagrama de la función Pfacc.

Del estudio del código fuente del proceso de filtrado y de los diagramas de dependencia de datos se pudo concluir que para tratar de acelerar el proceso de filtrado basta con mapear en los recursos lógicos del FPGA la función **PFACC**, debido a que esta representa la unidad central de procesamiento de la DFT (basada en la descomposición en factores primos presentada por Clive temperton en [10] y [11]).

2.3.1.1 Función PFACC

Después de que se tomó la decisión de implementar en los recursos lógicos del FPGA la función **PFACC**, el siguiente paso fue revisar detalladamente su código fuente el cual se presenta en el apéndice E.1. De esta revisión se pudo apreciar que este proceso maneja diez grupos de operaciones los cuales están sujetos al factor que se desea calcular (puede descomponer hasta diez factores), que a su vez depende de la longitud de la traza que se desea procesar.

Inicialmente se pensó en elaborar un solo *Datapath*, el cual sería utilizado por diez máquina de estados (una por factor) de forma simultánea. El problema de esta implementación radica en que al final de todo el proceso, ninguna de las diez máquinas de estados terminaría usando de manera eficiente al *Datapath*, lo cual afecta fuertemente su desempeño perdiendo una de las ventajas que ofrecen los procesadores específicos.

La segunda opción que se pensó fue hacer un procesador específico por cada factor, lo que nos llevaría a diez procesadores específicos. El problema de esta idea es que el trabajo se extendería por mucho más tiempo del planteado en el cronograma del proyecto (en el cual se presupuesto hacer un solo procesador específico) sin mencionar que al final no se podrían implementar los diez procesadores de forma simultánea sobre el FPGA debido a la falta de recursos lógicos. Recordemos que las operaciones que se manejan son de punto flotante de precisión sencilla y la

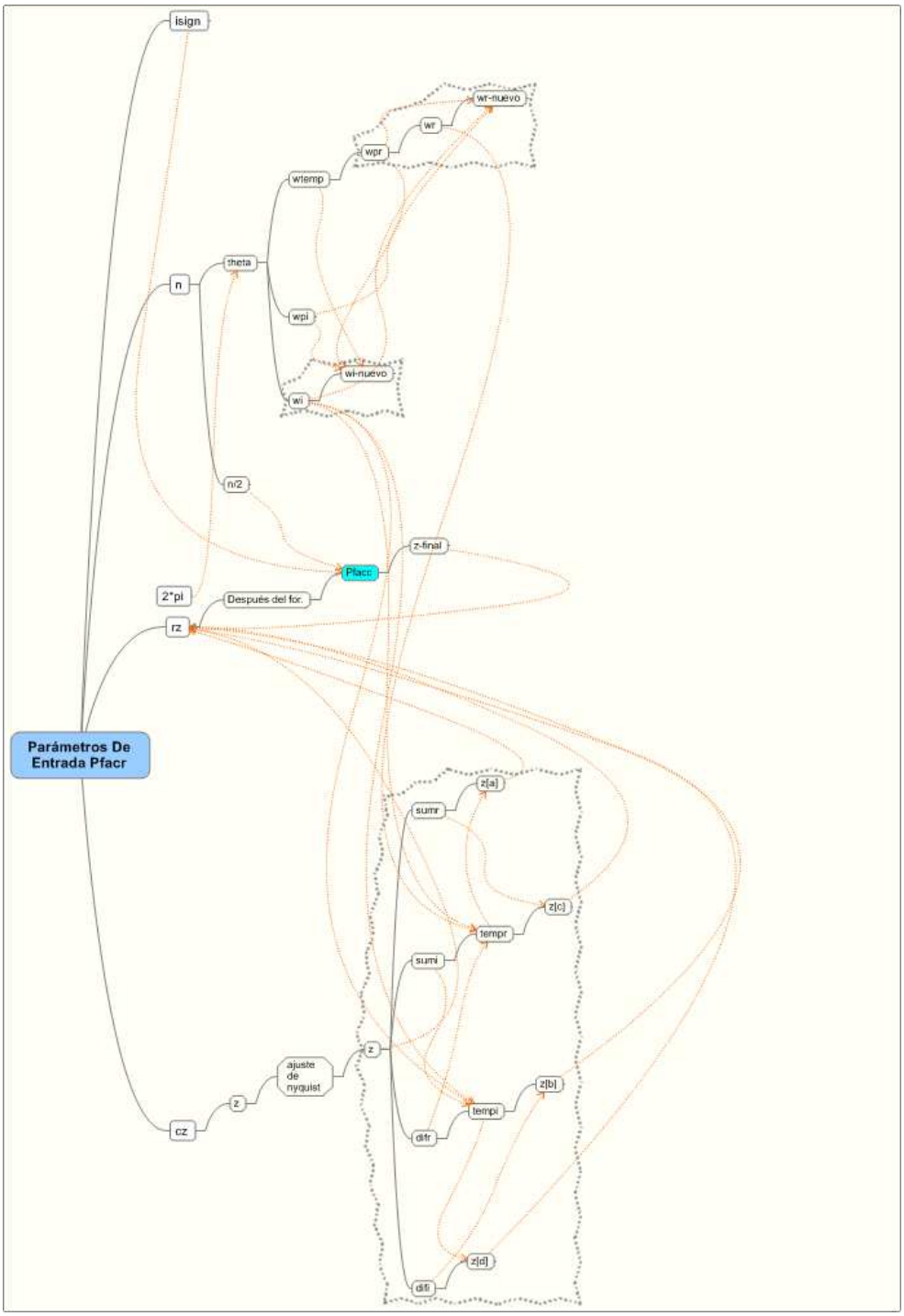


Figura 2.7: Diagrama de la función Pfacr.

cantidad de recursos lógicos que utilizaría cada factor estaría entre el 20% y el 40% de los recursos lógicos de la virtex 5.
Esto nos condujo a una tercera opción que consiste en hacer un procesador específico para un solo factor y luego dependiendo del impacto que este tenga sobre todo el proceso, se podría pensar en desarrollar los demás procesadores específicos por medio de proyectos de pre-grado posteriores a este trabajo de investigación. La ventaja de esta opción es que si más adelante se decide continuar con este trabajo desarrollando los demás procesadores específicos, se podría pensar en comprar la licencia de reconfiguración parcial de Xilinx y tomando como referencia el trabajo de maestría [5] se podría llegar a solucionar el problema de área.

2.3.1.2 Elección del factor

Luego de elegir la tercera opción como camino a seguir, se afrontó una última decisión de diseño y fue cual factor elegir del grupo de diez factores (los factores son 2, 3, 4, 5, 7, 8, 9, 11, 13, 16). Como la idea es que el único procesador específico que se va a hacer tenga el mayor impacto posible sobre el proceso final, necesariamente el factor elegido debería ser el más usado ó por lo menos estar en el grupo de los más usados. Como el uso de los factores depende de la longitud de las trazas a procesar y a su vez estas dependen de la frecuencia de muestreo usada a la hora de la adquisición y del tiempo de captura de los geófonos; se decidió consultar al ICP por las siguientes características:

- Tiempo típico de adquisición: 1 a 10 segundos.
- Frecuencia típica de muestreo: 0,002 Hz.

y a partir de estos datos se procedió a calcular el factor más usado como se muestra en las tablas 2.1 y 2.2.

Factores	16	13	11	9	8	7	5	4	3	2
Número de veces usado.	4	9	10	8	4	13	10	7	3	3

Tabla 2.1: Resumen del uso de los factores.

De la tabla 2.1 se pudo concluir que el factor más usado para los requerimientos típicos del ICP es **7** y por ello éste fue el seleccionado para ser mapeado en los recursos lógicos del FPGA. El código fuente de las operaciones que se deben realizar para el cálculo de este factor se encuentra entre las líneas 232 y 330 del apéndice E.1.

Frecuencia de Muestreo	**Tiempo de adquisición (segundos).**	**número de muestras.**	**nctab[in]=# de muestras /2**	**nctab[min]**	**Factores usados**	**nctab[max]**	**Factores usados**
0,002	1	500	250	240		252	
				15	16	28	9
				3	5	4	7
				1	3	1	4
0,002	2	1000	500	495		504	
				45	11	56	9
				5	9	7	8
				1	5	1	7
0,002	3	1500	750	728		770	
				56	13	70	11
				7	8	10	7
				1	7	2	5
						1	2
0,002	4	2000	1000	990		1001	
				90	11	77	13
				10	9	7	11
				2	5	1	7
				1	2		
0,002	5	2500	1250	1232		1260	
				77	16	140	9
				7	11	20	7
				1	7	4	5
						1	4
0,002	6	3000	1500	1456		1540	
				91	16	140	11
				7	13	20	7
				1	7	4	5
						1	4
0,002	7	3500	1750	1716		1820	
				132	13	140	13
				12	11	20	7
				3	4	4	5
				1	3	1	4
0,002	8	4000	2000	1980		2002	
				180	11	154	13
				20	9	14	11
				4	5	2	7
				1	4	1	2
0,002	9	4500	2250	2184		2288	
				168	13	143	16
				21	8	11	13
				3	7	1	11
				1	3		
0,002	10	5000	2500	2340		2520	
				180	13	280	9
				20	9	35	8
				4	5	5	7
				1	4	1	5

Tabla 2.2: Comparación del uso de los factores.

Capítulo 3

Diseño del Datapath y la máquina de estados.

3.1 Introducción

En este capitulo se presenta de forma detallada el procedimiento empleado durante la elaboración del *datapath* y la máquina de estados. Como el diseño se hizo usando la metodología planteada en [12], durante todo el capitulo se encontraran los datos obtenidos en cada uno de los pasos de dicha metodología.

3.2 Metodología de diseño del procesador

3.2.1 Selección del algoritmo a utilizar y definición de la entidad del procesador.

Al revisar la información presentada en la sección 2.3, se pudo concluir que el proceso que se iba a mapear en los recursos lógicos del FPGA es el factor siete de la función **PFACC**. Como la entidad del procesador es la encargada de definir cuales son las entradas y las salidas del mismo y el primer paso de la metodología de diseño consiste en definir la entidad; lo primero que se hizo fue revisar el código fuente de la función **PFACC** para identificar cuales son los valores de entrada que necesita el conjunto de operaciones que se encarga de calcular el factor siete y cuales son los valores de entrega cuando finaliza los cálculos y a partir de esta información construir el diagrama de la entidad del mismo. Se debe aclarar que este fue un proceso iterativo pero en la Figura 3.1 se muestra la versión final de la misma.

3.2.2 Diagrama ASM del procesador

Luego de definir el proceso que se va a llevar dentro de los recursos lógicos del FPGA, el siguiente paso es realizar el diagrama de máquina de estados algorítmica ó **diagrama ASM**. Debido a que este permite de manera gráfica describir con cierto nivel de detalle como será la implementación máquina de estados + *Datapath* (FSMD) [12]. En las Figuras 3.2, 3.3, 3.4 y 3.5 se muestra la versión final del diagrama ASM.

Al revisar el diagrama ASM se pueden identificar las siguientes características:

- La máquina de estados final tendrá como mínimo catorce estados.

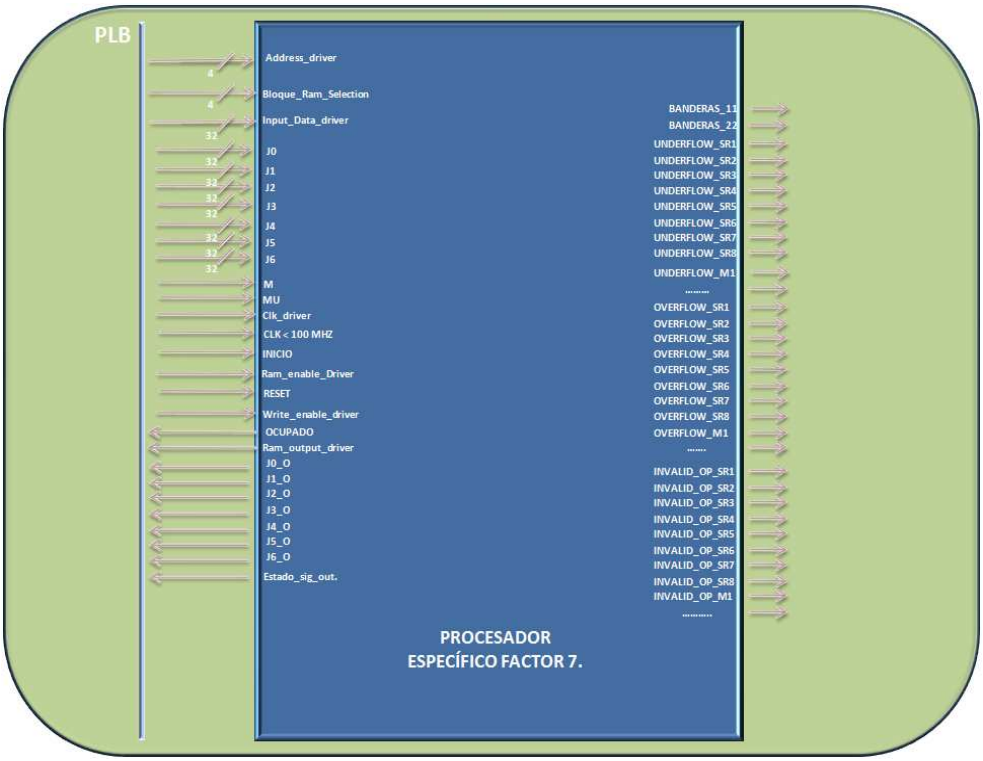


Figura 3.1: Entidad del procesador: Señales de entrada y salida del periférico.

- El número máximo de operaciones suma-resta por estado es de ocho, lo que indica que el procesador tendrá ocho unidades de punto flotante con la capacidad de hacer la operación de suma ó resta de forma simultánea.
- El número máximo de operaciones de multiplicación por estado es ocho, determinando que el procesador tendrá ocho unidades multiplicadoras en formato de punto flotante.
- Los tres primeros estados de la *fsm* estarán encargados de configurar los parámetros de operación del procesador.

Obteniendo de esta manera unas condiciones iniciales para el diseño del *Datapath* y de la *fsm*.

3.2.3 Diseño del Datapath

De acuerdo a la metodología planteada en [12] los pasos para elaborar el *datapath* son:

3.2.3.1 Agrupación de variables.

El objetivo de agrupar variables es determinar el menor número de registros para cumplir con las exigencias del proceso que se desea llevar a los recursos lógicos del FPGA. Para este caso es determinar el menor número de registros para poder implementar el proceso del factor siete.
Para ello se utiliza el "*algoritmo del lado izquierdo*" como se explica en [12]. En las Tablas 3.1 y 3.2 se muestra

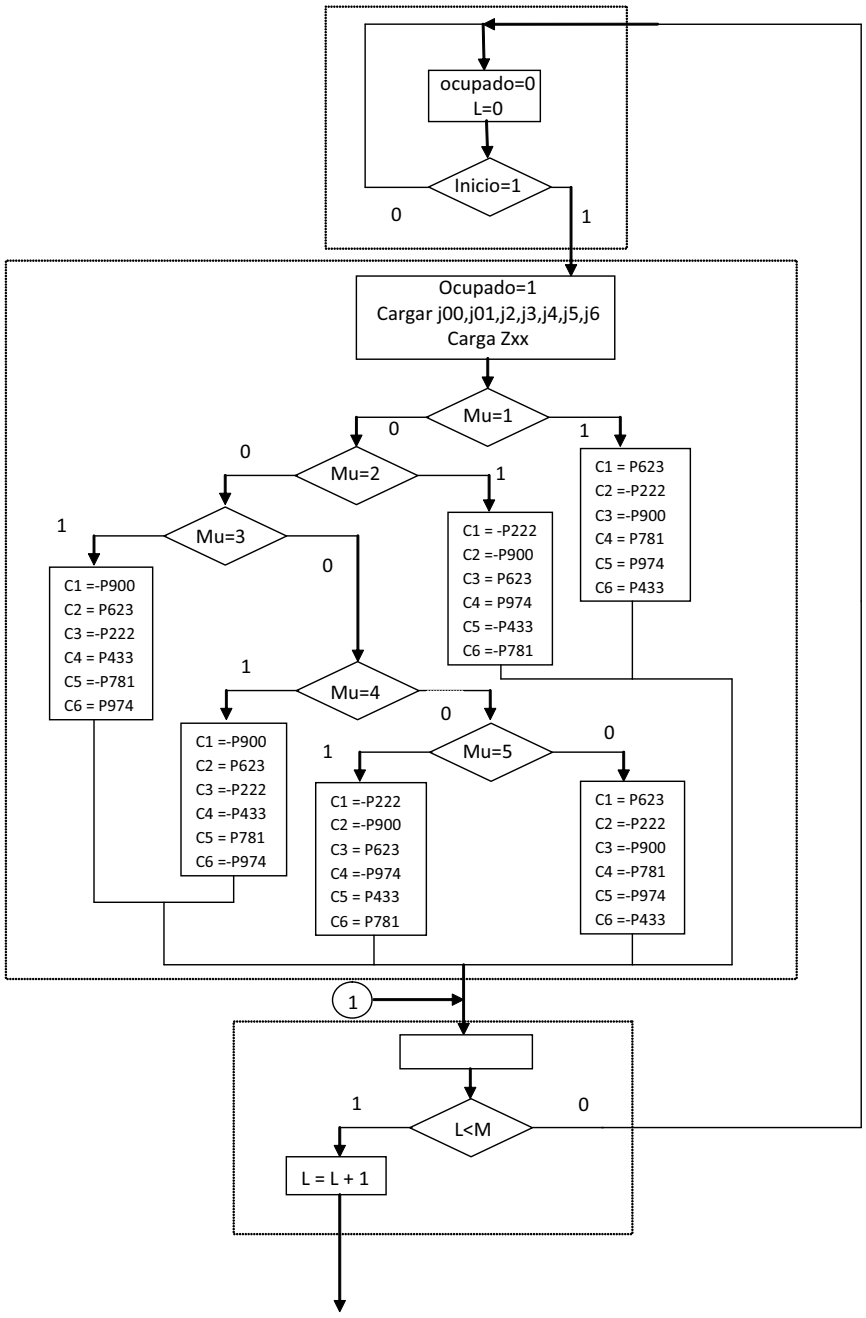


Figura 3.2: Diagrama ASM parte 1.

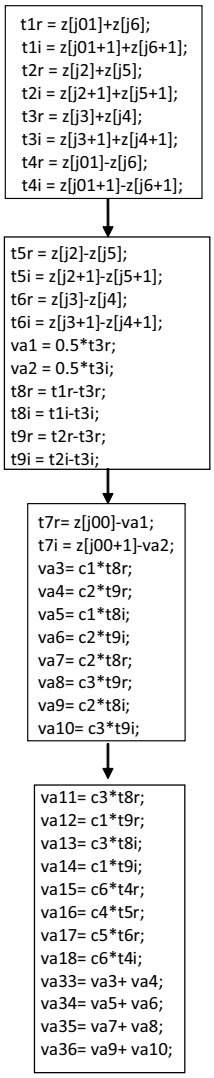


Figura 3.3: Diagrama ASM parte 2.

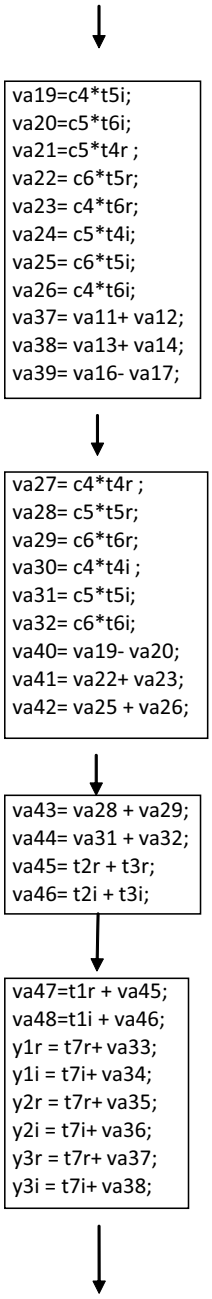


Figura 3.4: Diagrama ASM parte 3.

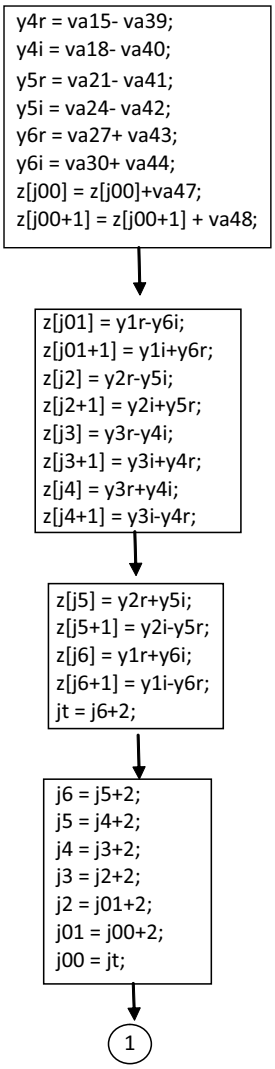


Figura 3.5: Diagrama ASM parte 4.

como esta distribuido el uso de variables del diagrama ASM mostrado en las Figuras 3.2, 3.3, 3.4 y 3.5. En ellas podemos observar que las filas representan las variables, las columnas representan los estados del diagrama ASM y las **x** muestran en cuales estados están activas las variables. Las variables se enumeran según su orden de escritura y cuando hay más de una variable que se escribe en el mismo estado entonces se le da prioridad a la que tiene un mayor tiempo de vida. Si llega a suceder que dos o más variables se escriben en el mismo estado y tienen el mismo tiempo de vida entonces el orden se hace de forma aleatoria.

Para el diseño del procesador se lograron agrupar 107 variables en 59 registros como se puede observar en las Tablas 3.3, 3.4 y 3.5.

3.2.3.2 Agrupación de operaciones en las unidades funcionales.

Para saber cuantas unidades funcionales va a necesitar el *Datapath*, se elabora una tabla en donde las filas representan las operaciones que se deben hacer en todo el diagrama ASM y las columnas representan el número de estados del mismo diagrama. De tal forma que en cada casilla se coloca el número de unidades funcionales que se necesita por estado; esta tabla recibe el nombre de *uso de operaciones* y la correspondiente al diagrama ASM se presenta en la Tabla 3.6 [12].

De esta tabla se puede apreciar que el *Datapath* necesita ocho unidades suma-resta y ocho unidades de multiplicación.

3.2.3.3 Agrupación de operaciones RT.

Después de identificar tanto el número de registros como el número de unidades funcionales que se necesitan para cumplir con las exigencias del diagrama ASM, se deben agrupar todas las operaciones según su registro destino como se indica en [12]. A continuación se muestra la agrupación de operaciones por registro destino para el diagrama ASM de las Figuras 3.2, 3.3, 3.4 y 3.5.

- Operaciones con el registro R1=L:

  R1 ← 0, en S0.

  R1 ← R1 + 1, en S2.

- Operaciones con el registro R2=Z0:

  R2 ← $Z0\_ent$, en S1.

  R2 ← R2 + va47, en S11.

- Operaciones con el registro R3=Z0+1:

  R3 ← $Z0 + 1\_ent$, en S1.

  R3 ← R3 + va48, en S11.

- Operaciones con el registro R4=Z01:

  R4 ← $Z01\_ent$, en S1.

  R4 ← y1r - y6i, en S12.

- Operaciones con el registro R5=Z01+1:

  R5 ← $Z01 + 1\_ent$, en S1.

  R5 ← y1i + y6r, en S12.

- Operaciones con el registro R6=Z02:

  R6 ← $Z02\_ent$, en S1.

  R6 ← y2r - y5i, en S12.

- Operaciones con el registro R7=Z02+1:

  R7 ← $Z02 + 1\_ent$, en S1.

  R7 ← y2i + y5r, en S12.

- Operaciones con el registro R8=Z03:

  R8 ← $Z03\_ent$, en S1.

  R8 ← y3r - y4i, en S12.

Variables	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
L		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z0		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z0+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z01		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z01+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z2		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z2+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z3		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z3+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z4		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z4+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z5		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z5+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z6		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Z6+1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
J5			X	X	X	X	X	X	X	X	X	X	X	X	X
J4			X	X	X	X	X	X	X	X	X	X	X	X	X
J3			X	X	X	X	X	X	X	X	X	X	X	X	X
J2			X	X	X	X	X	X	X	X	X	X	X	X	X
J1			X	X	X	X	X	X	X	X	X	X	X	X	X
J0			X	X	X	X	X	X	X	X	X	X	X	X	X
J6			X	X	X	X	X	X	X	X	X	X	X	X	
C4			X	X	X	X	X	X	X	X	X	X	X	X	X
C5			X	X	X	X	X	X	X	X	X	X	X	X	X
C6			X	X	X	X	X	X	X	X	X	X	X	X	X
C1			X	X	X	X	X	X	X	X	X	X	X	X	X
C3			X	X	X	X	X	X	X	X	X	X	X	X	X
C2			X	X	X	X	X	X	X	X	X	X	X	X	X
t1r					X	X	X	X	X	X	X				
t1i					X	X	X	X	X	X	X				
t2r					X	X	X	X	X	X					
t2i					X	X	X	X	X	X					
t3r					X	X	X	X	X	X					
t3i					X	X	X	X	X	X					
t4r					X	X	X	X	X						
t4i					X	X	X	X	X						
t5r						X	X	X	X						
t5i						X	X	X	X						
t6r						X	X	X	X						
t6i						X	X	X	X						
t8r						X	X								
t8i						X	X								
t9r						X	X								
t9i						X	X								
va1						X									
va2						X									
t7r							X	X	X	X	X				
t7i							X	X	X	X	X				
va3							X								
va4							X								
va5							X								
va6							X								
va7							X								
va8							X								
va9							X								
va10							X								
va15								X	X	X	X	X			
va18								X	X	X	X	X			
va33								X	X	X	X				
va34								X	X	X	X				
va35								X	X	X	X				
va36								X	X	X	X				
va11								X							
va12								X							
va13								X							
va14								X							
va16								X							
va17								X							

Tabla 3.1: Uso de variables.

Variables	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
va21									X	X	X	X			
va24									X	X	X	X			
va39									X	X	X	X			
va37									X	X	X				
va38									X	X	X				
va19									X						
va20									X						
va25									X						
va26									X						
va22									X						
va23									X						
va40										X	X	X			
va41										X	X	X			
va27										X	X	X			
va30										X	X	X			
va42										X	X	X			
va28										X					
va29										X					
va31										X					
va32										X					
va43											X	X			
va44											X	X			
va45											X				
va46											X				
y1r												X	X	X	
y1i												X	X	X	
y2r												X	X	X	
y2i												X	X	X	
y3r												X	X		
y3i												X	X		
va47												X			
va48												X			
y5r													X	X	
y5i													X	X	
y6r													X	X	
y6i													X	X	
y4r													X		
y4i													X		
jt															X

Tabla 3.2: Uso de variables continuación.

Variables	Registro
L	R1
Z0	R2
Z0+1	R3
Z01	R4
Z01+1	R5
Z02	R6
Z02+1	R7
Z03	R8
Z03+1	R9
Z04	R10
Z04+1	R11
Z05	R12
Z05+1	R13
Z06	R14
Z06+1	R15
J6	R28
J5	R16
J4	R17
J3	R18
J2	R19
J1	R20
J0	R21

Tabla 3.3: Número de registros parte 1.

Variables	Registro
C1	R22
C2	R23
C3	R24
C4	R25
C5	R26
C6	R27
t1r	R29
t1i	R30
t2r	R31
t2i	R32
t3r	R33
t3i	R34
t4r	R35
t4i	R36
t5r	R37
t5i	R38
t6r	R39
t6i	R40
t7r	R45
t7i	R46
t8r	R41
t8i	R42
t9r	R43
t9i	R44
va1	R45
va2	R46
va3	R47
va4	R48
va5	R49
va6	R50
va7	R51
va8	R52
va9	R53
va10	R54
va11	R49
va12	R50
va13	R51
va14	R52
va15	R41
va16	R53
va17	R54
va18	R42
va19	R54
va20	R55
va21	R49
va22	R58
va23	R59
va24	R50
va25	R56
va26	R57
va27	R37
va28	R40
va29	R54
va30	R38
va31	R55
va32	R56
va33	R43
va34	R44
va35	R47
va36	R48
va37	R52
va38	R53
va39	R51
va40	R35
va41	R36
va42	R39
va43	R31

Tabla 3.4: Número de registros parte 2.

Variables	Registro
va44	R32
va45	R33
va46	R34
va47	R44
va48	R45
y1r	R29
y1i	R30
y2r	R33
y2i	R34
y3r	R40
y3i	R43
y4r	R37
y4i	R38
y5r	R31
y5i	R32
y6r	R35
y6i	R36
jt	R28

Tabla 3.5: Número de registros parte 3.

Operaciones	**S0**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**Número máximo de unidades**
Suma			1	6			4	3	1	4	8	4	4	3	6	8
Resta				2	8	2			2			4	4	2		8
Multiplicación					2	8	8	8	6							8
unidades simultáneas de suma y resta	0	0	1	8	8	2	4	3	3	4	8	8	8	5	6	8
unidades simultáneas de multiplicación	0	0	0	0	2	8	8	8	6	0	0	0	0	0	0	8

Tabla 3.6: Uso de operaciones.

- Operaciones con el registro R9=Z03+1:

  R9 $\leftarrow Z03 + 1\_ent$, en S1.

  R9 ← y3i + y4r, en S12.

- Operaciones con el registro R10=Z04:

  R10 $\leftarrow Z04\_ent$, en S1.

  R10 ← y3r + y4i, en S12.

- Operaciones con el registro R11=Z04+1:

  R11 $\leftarrow Z04 + 1\_ent$, en S1.

  R11 ← y3i - y4r, en S12.

- Operaciones con el registro R12=Z05:

  R12 $\leftarrow Z05\_ent$, en S1.

  R12 ← y2r + y5i, en S13.

- Operaciones con el registro R13=Z05+1:

  R13 $\leftarrow Z05 + 1\_ent$, en S1.

  R13 ← y2i - y5r, en S13.

- Operaciones con el registro R14=Z06:

  R14 $\leftarrow Z06\_ent$, en S1.

  R14 ← y1r + y6i, en S13.

- Operaciones con el registro R15=Z06+1:

  R15 $\leftarrow Z06 + 1\_ent$, en S1.

  R15 ← y1i - y6r, en S13.

- Operaciones con el registro R16=j5:

  R16 $\leftarrow j5\_ent$, en S1.

  R16 ← j4 + 2, en S14.

- Operaciones con el registro R17=j4:

  R17 $\leftarrow j4\_ent$, en S1.

  R17 ← j3 + 2, en S14.

- Operaciones con el registro R18=j3:

  R18 $\leftarrow j3\_ent$, en S1.

  R18 ← j2 + 2, en S14.

- Operaciones con el registro R19=j2:
  - R19 ← *j2_ent*, en S1.
  - R19 ← j1 + 2, en S14.
- Operaciones con el registro R20=j1:
  - R20 ← *j1_ent*, en S1.
  - R20 ← j0 + 2, en S14.
- Operaciones con el registro R21=j0:
  - R21 ← *j0_ent*, en S1.
  - R21 ← jt, en S14.
- Operaciones con el registro R22:
  - R22 ← C1, en S1.
- Operaciones con el registro R23:
  - R23 ← C2, en S1.
- Operaciones con el registro R24:
  - R24 ← C3, en S1.
- Operaciones con el registro R25:
  - R25 ← C4, en S1.
- Operaciones con el registro R26:
  - R26 ← C5, en S1.
- Operaciones con el registro R27:
  - R27 ← C6, en S1.
- Operaciones con el registro R28=jt:
  - R28 ← *j6_ent*, en S1.
  - R28 ← j6 + 2, en S13.
  - R28 ← j5 + 2, en S14.
- Operaciones con el registro R29:
  - R29 ← Z01 + Z06 (t1r), en S3.
  - R29 ← t7r + va33 (y1r), en S10.
- Operaciones con el registro R30:
  - R30 ← Z01+1 + Z06+1 (t1i), en S3.
  - R30 ← t7i + va34 (y1i), en S10.
- Operaciones con el registro R31:
  - R31 ← Z02 + Z05 (t2r), en S3.
  - R31 ← va28 + va29 (va43), en S9.
  - R31 ← va21 - va41 (y5r), en S11.
- Operaciones con el registro R32:
  - R32 ← Z02+1 + Z05+1 (t2i), en S3.
  - R32 ← va31 + va32 (va44), en S9.
  - R32 ← va24 - va42 (y5i), en S11.
- Operaciones con el registro R33:
  - R33 ← Z03 + Z04 (t3r), en S3.
  - R33 ← t2r + t3r (va45), en S9.
  - R33 ← t7r + va35 (y2r), en S10.
- Operaciones con el registro R34:
  - R34 ← Z03+1 + Z04+1 (t3i), en S3.
  - R34 ← t2i + t3i (va46), en S9.
  - R34 ← t7i + va36 (y2i), en S10.
- Operaciones con el registro R35:
  - R35 ← Z01 - Z06 (t4r), en S3.
  - R35 ← va19 - va20 (va40), en S8.
  - R35 ← va27 + va43 (y6r), en S11.
- Operaciones con el registro R36:
  - R36 ← Z01+1 - Z06+1 (t4i), en S3.
  - R36 ← va22 + va23 (va41), en S8.
  - R36 ← va30 + va44 (y6i), en S11.
- Operaciones con el registro R37:
  - R37 ← Z02 - Z05 (t5r), en S4.
  - R37 ← C4*t4r (va27), en S8.
  - R37 ← va15 - va39 (y4r), en S11.
- Operaciones con el registro R38:
  - R38 ← Z02+1 - Z05+1 (t5i), en S4.
  - R38 ← C4*t4i (va30), en S8.
  - R38 ← va18 - va40 (y4i), en S11.
- Operaciones con el registro R39:
  - R39 ← Z03 - Z04 (t6r), en S4.
  - R39 ← va25 + va26 (va42), en S8.
- Operaciones con el registro R40:
  - R40 ← Z03+1 - Z04+1 (t6i), en S4.
  - R40 ← C5*t5r (va28), en S8.
  - R40 ← t7r + va37 (y3r), en S10.

- Operaciones con el registro R41:
  - R41 ← t1r - t3r (t8r), en S4.
  - R41 ← C6*t4r (va15), en S6.
- Operaciones con el registro R42:
  - R42 ← t1i - t3i (t8i), en S4.
  - R42 ← C6*t4i (va18), en S6.
- Operaciones con el registro R43:
  - R43 ← t2r - t3r (t9r), en S4.
  - R43 ← va3 + va4 (va33), en S6.
  - R43 ← t7i + va38 (y3i), en S10.
- Operaciones con el registro R44:
  - R44 ← t2i - t3i (t9i), en S4.
  - R44 ← va5 + va6 (va34), en S6.
  - R44 ← t1r + va45 (va47), en S10.
- Operaciones con el registro R45:
  - R45 ← 0.5*t3r (va1), en S4.
  - R45 ← Z00 - va1 (t7r), en S5.
  - R45 ← t1i + va46 (va48), en S10.
- Operaciones con el registro R46:
  - R46 ← 0.5*t3i (va2), en S4.
  - R46 ← Z00+1 - va2 (t7i), en S5.
- Operaciones con el registro R47:
  - R47 ← C1*t8r (va3), en S5.
  - R47 ← va7 + va8 (va35), en S6.
- Operaciones con el registro R48:
  - R48 ← C2*t9r (va4), en S5.
  - R48 ← va9 + va10 (va36), en S6.
- Operaciones con el registro R49:
  - R49 ← C1*t8i (va5), en S5.
  - R49 ← C3*t8r (va11), en S6.
  - R49 ← C5*t4r (va21), en S7.
- Operaciones con el registro R50:
  - R50 ← C2*t9i (va6), en S5.
  - R50 ← C1*t9r (va12), en S6.
  - R50 ← C5*t4i (va24), en S7.
- Operaciones con el registro R51:
  - R51 ← C2*t8r (va7), en S5.
  - R51 ← C3*t8i (va13), en S6.
  - R51 ← va16 - va17 (va39), en S7.
- Operaciones con el registro R52:
  - R52 ← C3*t9r (va8), en S5.
  - R52 ← C1*t9i (va14), en S6.
  - R52 ← va11 + va12 (va37), en S7.
- Operaciones con el registro R53:
  - R53 ← C2*t8i (va9), en S5.
  - R53 ← C4*t5r (va16), en S6.
  - R53 ← va13 + va14 (va38), en S7.
- Operaciones con el registro R54:
  - R54 ← C3*t9i (va10), en S5.
  - R54 ← C5*t6r (va17), en S6.
  - R54 ← C4*t5i (va19), en S7.
  - R54 ← C6*t6r (va29), en S8.
- Operaciones con el registro R55:
  - R55 ← C5*t6i (va20), en S7.
  - R55 ← C5*t5i (va31), en S8.
- Operaciones con el registro R56:
  - R56 ← C6*t5i (va25), en S7.
  - R56 ← C6*t6i (va32), en S8.
- Operaciones con el registro R57:
  - R57 ← C4*t6i (va26), en S7.
- Operaciones con el registro R58:
  - R58 ← C6*t5r (va22), en S7.
- Operaciones con el registro R59:
  - R59 ← C4*t6r (va23), en S7.

3.2.3.4 Construcción del *hardware*.

Finalmente, luego de conocer la cantidad de unidades funcionales necesarias, el número de registros y de haber hecho la agrupación de operaciones RT, el último paso es distribuir dichas operaciones entre las unidades funcionales y agregar los elementos necesarios para poder realizar la interconexión como multiplexores, buses, etc..
En la Tabla 3.7 se presenta la distribución de las operaciones RT sobre las unidades funcionales, se utiliza la notación **SR** para representar una unidad funcional suma-resta y la **M** para representar una unidad funcional de multiplicación.

Unidad Funcional	**Registros que la usan**
SR1	R29,R37,R4,R12 y R16.
SR2	R30,R38,R46,R5,R13 y R17.
SR3	R31,R39,R47,R6,R14,R18 y R45.
SR4	R32,R40,R48,R7,R15 y R19.
SR5	R33,R41,R52,R2,R8,R28 y R21.
SR6	R34,R42,R53,R3,R9 y R20.
SR7	R35,R43,R51 y R10.
SR8	R36,R44 y R11.
M1	R55.
M2	R58.
M3	R49.
M4	R50.
M5	R59.
M6	R56.
M7	R57.
M8	R54.

Tabla 3.7: Operaciones RT y unidades funcionales.

Para un mayor detalle en el apéndice F se presenta la versión final del *Datapath* módulo por módulo (ver Figuras F.1,F.2,F.3,F.4,F.5,F.6,F.7,F.8,F.9,F.10,F.11,F.12,F.13,F.14,F.15 y F.16). El código fuente de la descripción en VHDL del *Datapath* se encuentra en el apéndice G.

3.2.3.5 Adición del *hardware* para evitar *meta-estabilidad*.

Como el procesador tiene que interactuar con elementos externos que pueden entregarle información en cualquier instante de tiempo, es posible que en el momento de la captura de dichos datos no se este cumpliendo con los tiempos de *setup* y de *hold* de los flip flops que forman los registros de entrada. Cuando estos tiempos no se cumplen, el comportamiento del flip flop puede llegar a ser aleatorio; este estado se conoce con el nombre de *Metaestabilidad* [13]. Para evitar la *Metaestabilidad* se implementó el protocolo *Start Finish* [14], el cual consiste en ubicar los datos en los puertos de entrada (ó en las memorias de entrada para este caso) y una vez que la información está lista se procede a activar la carga de los datos en los registros mediante la señal de **Inicio**. De esta manera en el siguiente flanco de reloj se realiza la captura de los datos y se evita la *Metaestabilidad*. Durante la ejecución del proceso la señal **ocupado** se activa indicándole a los dispositivos externos que deben esperar para hacer la próxima carga de datos.
El único aspecto que faltaría por solucionar es la sincronización de la señal de **Inicio** con el reloj del procesador, debido a que como ésta puede ser activada en cualquier instante de tiempo, también podría generar un estado de *Metaestabilidad* en el flip flop que la captura. Para evitar esta situación se utilizó el esquema propuesto en el numeral **G** de [15] y la descripción en verilog del mismo se muestra en G.29.

3.2.4 Diseño de la FSM

Después de finalizado el *Datapath*, se debía elaborar la unidad de control encargada de dirigir el flujo de los datos de acuerdo a la necesidad del proceso. Para ello, basándonos en el diagrama ASM se pudo apreciar que como mínimo la unidad de control debía tener 14 estados; pero como existe una dependencia de datos entre la mayoría de los estados del diagrama ASM, se concluyo que a partir del tercer estado es necesario agregar un estado de espera entre todos los estados posteriores para garantizar que las unidades funcionales alcancen a entregar los resultados. En la Figura 3.6 se presenta la versión final de la máquina de estados utilizada para controlar el flujo de los datos. Luego de finalizado el diseño de la máquina de estados se procedió a la descripción en *Hardware* de la misma. En ese momento fue necesario revisar las diferentes técnicas de codificación de los bits de estado, encontrando que la forma más idónea de hacer la implementación es la mostrada en [14], la cual en lugar de tener una lógica de salida, utiliza los bits de estado directamente como salidas y de esta forma elimina el problema de los *glitches*. La implementación final de la máquina de estados usó 273 bits de estado y se puede apreciar en el Apéndice G.18.

3.2.5 Depuración del procesador

Luego de finalizada la implementación del procesador se verificó su correcto funcionamiento dentro del FPGA. Para ello se usaron los siguientes módulos de la herramienta *ChipScope* de Xilinx:

- **Chipscope core inserter**: Por medio de este módulo se insertó dentro del procesador una *Implemented Logic Analizer (ILA)* para capturar un conjunto de señales internas del FPGA bajo unas condiciones de disparo y número de muestras definidas previamente. La principal característica de esta *ILA* es que permite capturar estas señales bajo tiempo de ejecución.
- **Chipscope Analizer**: Por medio de esta interfaz gráfica se puede observar toda la información capturada por la *ILA* (buses de datos, registros, banderas, etc..) y se puede modificar de forma dinámica las condiciones de captura. En la Figura 3.7 se puede observar una imagen de ejemplo de la herramienta.

La metodología empleada para depurar el funcionamiento del procesador fue cargar previamente unas condiciones iniciales en las catorce memorias de entrada y a partir de estos valores hacer un seguimiento a cada uno de los registros en cada uno de los estados, garantizando que los resultados obtenidos en cada estado por las diferentes unidades funcionales hayan sido capturados correctamente por sus respectivos registros. De esta manera se pudo verificar que el procesador funciona correctamente y se finalizó la etapa de diseño e implementación del procesador.

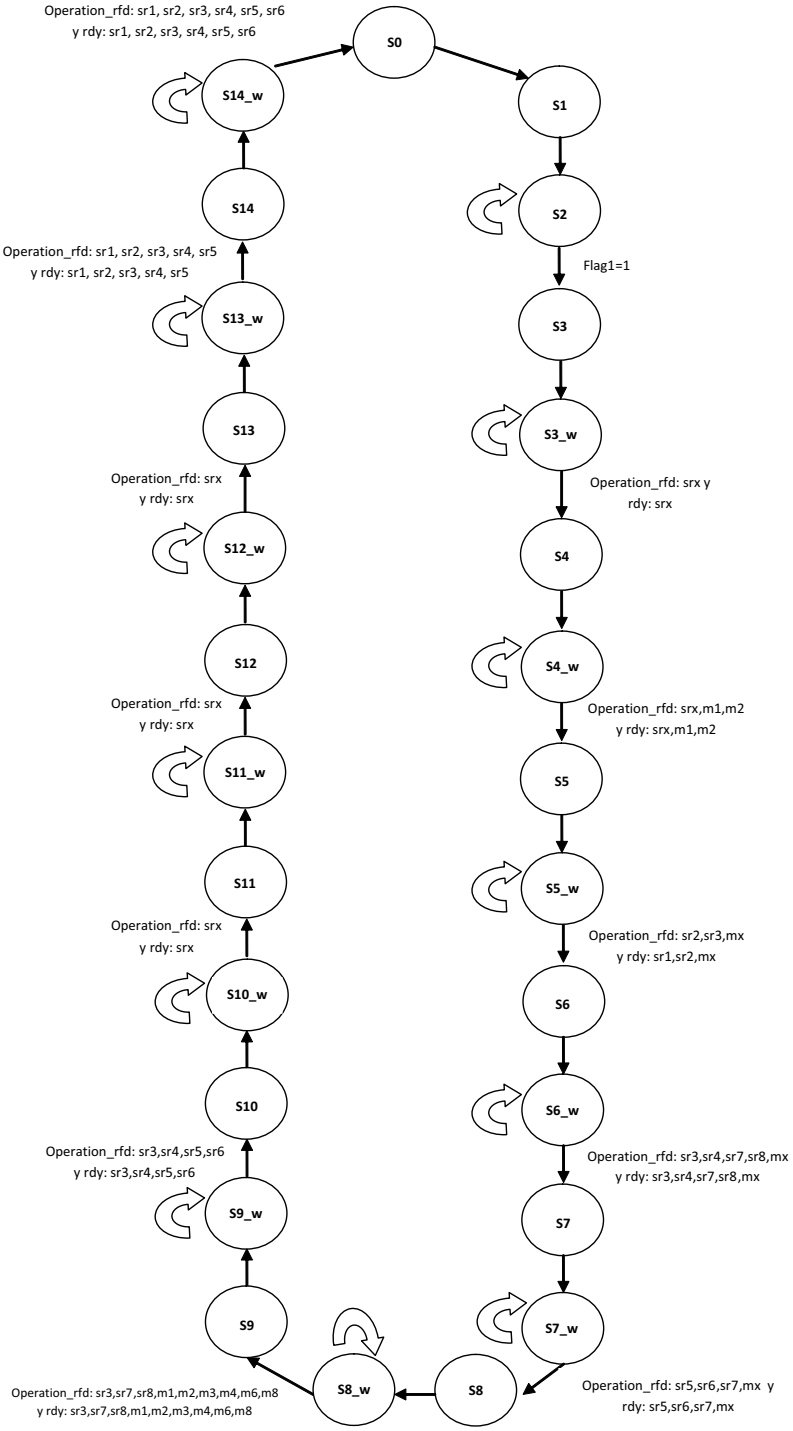


Figura 3.6: Máquina de estados del procesador

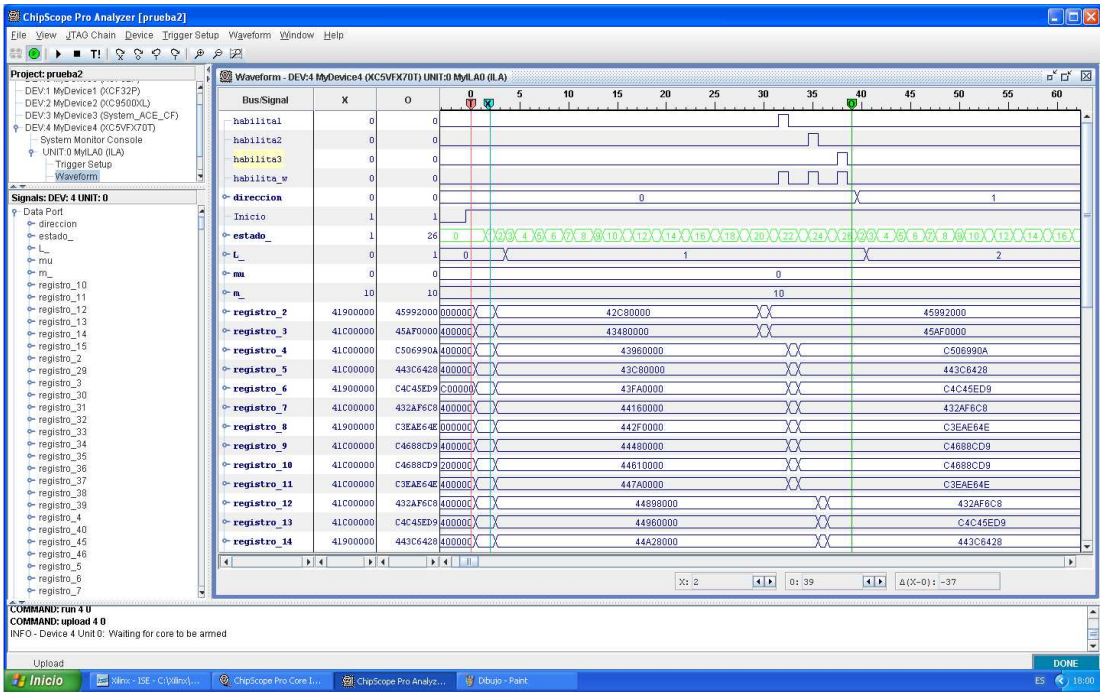


Figura 3.7: Captura con el Chipscope

Capítulo 4

Elaboración de la interfaz entre el Power PC y los periféricos de migración.

4.1 Introducción

Este capitulo describe el trabajo realizado para la configuración y montaje del Cluster Okinawa (plataforma de trabajo elaborada para poder medir el desempeño del FPGA cuando ejecuta aplicaciones de alto desempeño, como el proceso de migración sísmica) y además describe la plantilla del driver elaborado para integrar el procesador específico dentro de linux manteniendo la filosofía de sistema de archivos. Como este trabajo se desarrolló de forma conjunta con el proyecto de maestría [5], los aspectos técnicos específicos se encuentran definidos en el apéndice A de [5].

4.2 Configuración y Montaje del Cluster Okinawa

Debido a la necesidad de contar con una plataforma de trabajo que permita medir el desempeño del FPGA cuando ejecuta el proceso de migración sísmica, se decidió instalar y configurar una red conformada por tres procesadores de propósito general y 2 FPGA´s. Esta red recibió el nombre de red Okinawa y en la Figura 4.1 se puede observar el esquema del montaje final.

La red cuenta con un router D-Link DIR-600 como dispositivo de red, el cual tiene como propósito aislar a Okinawa de la red externa para evitar la interacción de la misma con elementos externos indeseados. Dentro de la red se encuentra un computador de escritorio minimac, el cual cumple el papel de Maestro (llamado Okinawa-00), dos computadores portátiles Hp, cada uno con una máquina virtual sobre la cual se encuentra instalada Linux Debian 5.0 (llamados Okinawa-01 y Okinawa-02) y dos sistemas de desarrollo ML507 de la empresa Xilinx (llamados Okinawa-21 y Okinawa-22) que contienen los FPGAs Virtex5 FX70T sobre los cuales se está desarrollando el trabajo de investigación.

La decisión de agregar estos dos procesadores de propósito general (Okinawa-01 y Okinawa-02) dentro del Cluster Okinawa se debe a la necesidad de reconstruir un ambiente de trabajo típico de un Cluster heterogéneo.

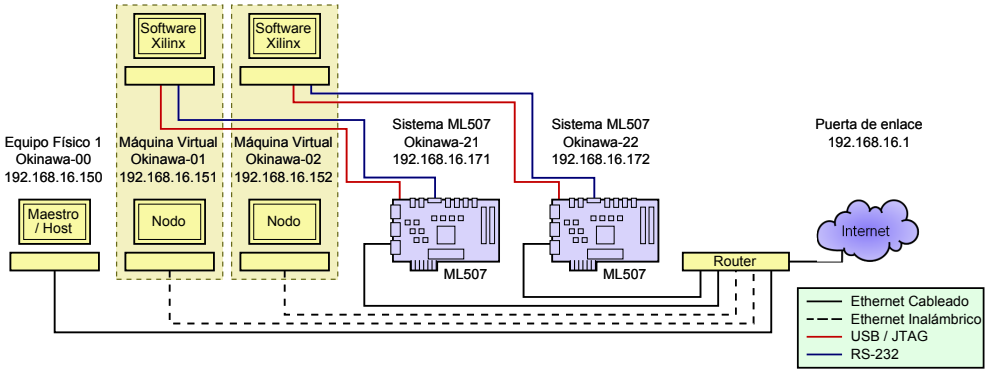


Figura 4.1: Diagrama de la implementación física del sistema (tomada de [5]).

4.2.1 Instalación y configuración del Sistema operativo sobre el Cluster Okinawa.

Como se esta trabajando sobre un cluster heterogéneo, la instalación del sistema operativo (S.O.) sobre cada uno de los nodos es diferente. Por ello se realizan dos procedimientos diferentes. El primero muestra la forma como se instaló el S.O. sobre los procesadores de propósito general (PPG) y el segundo explica el procedimiento realizado para poder instalar el S.O. sobre los sistemas de desarrollo de los FPGAs o procesadores de propósito específico (PPE).

4.2.1.1 Instalación del S.O. sobre los PPG.

El sistema operativo que se decidió instalar sobre todos los nodos del Cluster fue Linux Debian 5.0, debido a que era necesario tener un sistema operativo gratuito y con el código abierto para poder configurar los parámetros necesarios durante la instalación. En la Figura 4.2 se puede ver que los nodos que tienen procesadores de propósito general son llamados Okinawa-00, Okinawa-01 y Okinawa-02. La instalación del sistema operativo sobre estos tres nodos se hizo de la forma tradicional usando el sistema base que ofrece linux (paquetes mínimos para que el sistema funcione) e instalando manualmente los paquetes que se requieren para de esta forma tener un mayor control sobre el sistema completo.

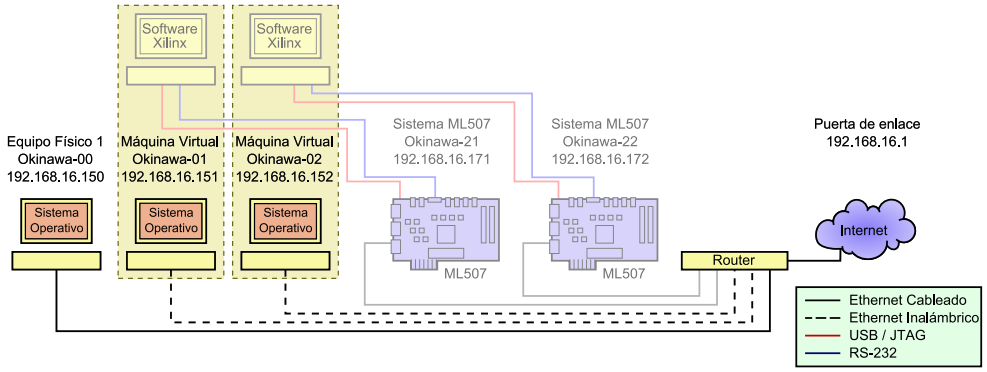


Figura 4.2: Nodos involucrados en esta etapa (tomada de [5]).

Luego de instalado el S.O. base se procedió a configurar algunos aspectos del cluster, los cuales se resumen a continuación. Para un mayor detalle sobre la instalación ó configuración de dichos servicios consultar el apéndice A de [5].

- **Configuración del servidor DHCP.** Aunque se podía instalar el servidor DHCP en uno de estos tres nodos, se decidió configurar al router para esta tarea.
- **Configuración de la interfaz de red.** Como es necesario que todos los nodos tengan acceso a Internet, en cada uno de los mismos se configuró la interfaz de red. Al nodo maestro se le asigno una dirección estática, mientras que para el resto de los nodos la dirección se configura mediante DHCP, facilitando la clonación de nodos dentro del cluster pensando en una expansión del mismo.
- **Configuración del repositorio de Internet.** La filosofía de trabajo de linux es tener un repositorio en Internet en el cual se encuentran todos los paquetes que soportan las distintas aplicaciones que se desean usar sobre este sistema operativo. Como desde un principio se instaló el sistema base de linux sobre este cluster, es necesario tener acceso a dicho repositorio para poder ir instalando los paquetes que se vayan necesitando. Por lo tanto se configuró el sistema para que usara el repositorio principal de debian (debian.org).
- **Servidor DNS y RDNS.** Los servidores DNS y RDNS son los que permiten relacionar el nombre en la red de un equipo con su respectiva dirección IP. En Okinawa Cluster se instalaron estos servidores como se muestra en la figura 4.3 pensando en que con este servicio algunas herramientas del Cluster funcionan más fácil y adicionalmente se facilita el escalado del mismo.
- **Servidor y clientes NFS.** El sistema de archivos de red (NFS), es un protocolo de nivel de aplicación que permite formar un sistema de archivos distribuido en un entorno de red, facilitando el flujo de la información entre los nodos de la misma. Con la instalación de este servicio sobre el cluster, a través de las carpetas compartidas, se logró hacer más sencilla la interacción entre los nodos.
- **Instalación de SSH.** La instalación de este servicio fue necesaria debido a que bastantes herramientas de trabajo sobre clusters la necesitan, además ofrece como ventaja el acceso remoto a cada uno de los nodos del cluster sin necesidad de ubicarse físicamente cerca de él.
- **Servidor TFTP.** Esta herramienta es fundamental para instalar el sistema operativo sobre los nodos FPGA, debido a que permite descargar el Kernel y los archivos de configuración durante el arranque de dichos nodos. El servidor TFTP se instaló en el nodo maestro y el cliente TFTP sobre los nodos FPGA.

4.2.1.2 Instalación del S.O. sobre los nodos FPGA.

El sistema base con el cual se configuraron los nodos FPGA del cluster Okinawa se puede descargar de https://secure.xilinx.com/webreg/
Este archivo contiene un proyecto en EDK con el mapa de memoria configurado como se muestra en la Tabla 4.1. Para la prueba inicial se descargo el bitstream a cada uno de los nodos como se muestra en la Figura 4.4. Luego de descargar estos bitstreams se procedió a instalar el bootloader sobre cada uno de los nodos FPGA.

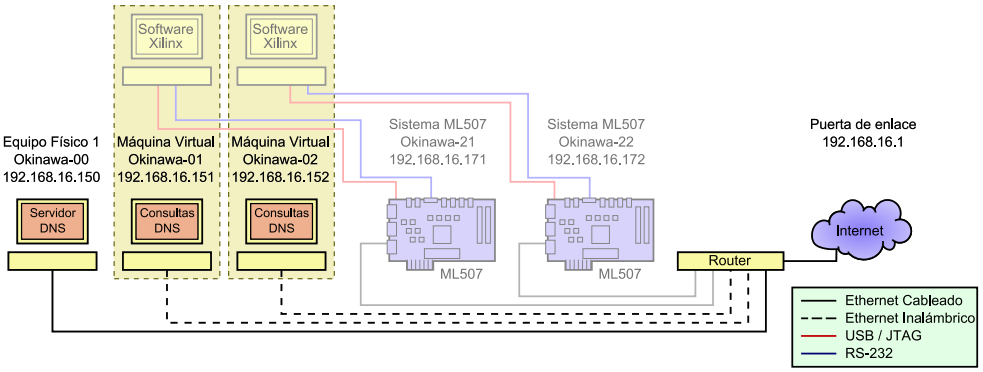


Figura 4.3: Nodos sobre los cuales se instalo este servicio (tomada de [5]).

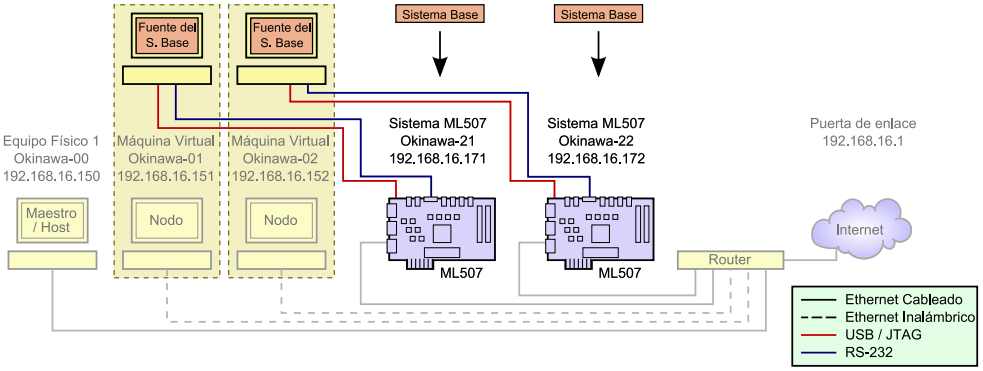


Figura 4.4: Elementos involucrados en la implementación del sistema base (tomada de [5]).

Elemento	Dirección Base	Tamaño Rango
Procesador PowerPC®440		
Bloque Ram	0xFFFF0000	64K
GPIO Leds	0x81400000	64K
GPIO Positions	0x81420000	64K
GPIO PushButtons	0x81440000	64K
GPIO DipSwitch	0x81460000	64K
IIC_EEPROM	0x81600000	64K
Controlador de Interrupciones	0x81800000	64K
MAC HW Ethernet	0x81C00000	64K
SysACE_CompactFlash	0x83600000	64K
Temporizador Watchdog	0x83A00000	64K
Temporizador	0x83C00000	64K
Puerto Serie RS-232	0x83E00000	64K
RAM DDR2	0x00000000	256MB
Memoria Flash	0xFC000000	32MB

Tabla 4.1: Mapa de memoria del sistema base (tomado de [5]).

4.2.1.3 Instalación del Bootloader

La instalación del bootloader sobre los nodos FPGAs se realizó de la siguiente manera:

- Se descargó la fuente de u-boot de la dirección http://git.xilinx.com/cgi-bin/gitweb.cgi?p=u-boot-xlnx.
- Se modificaron algunos parámetros de la fuente para que coincidiera con las especificaciones del proyecto.
- Luego se compiló la fuente del u-boot modificada en el nodo maestro para obtener el bootloader.
- Finalmente se descargó el archivo u-boot.srec (bootloader) en la memoria Flash de cada uno de los nodos FPGA como se muestra en la Figura 4.5.

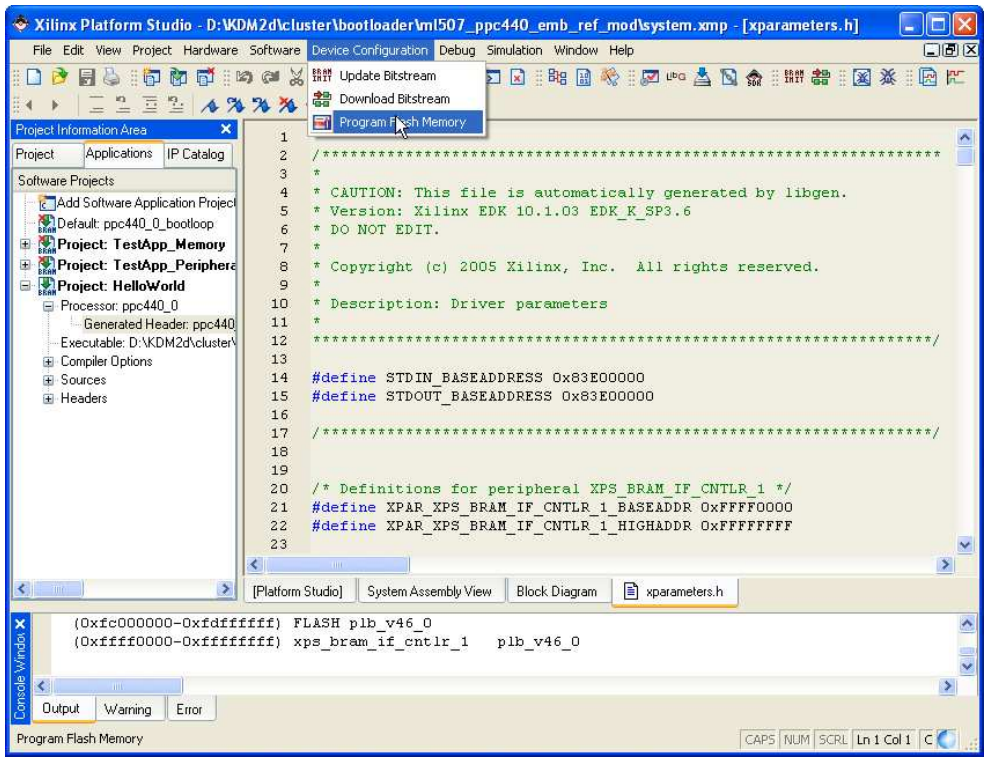


Figura 4.5: Función de EDK para descargar un archivo a la memoria Flash (tomada de [5]).

Asi los nodos FPGAs quedaron cargados con el sistema base dentro del FPGA y con el bootloader dentro de la memoria Flash externa como se puede observar en la Figura 4.6. Como la configuración de los FPGAs se pierde cada vez que se apaga el sistema, se decidió guardar el archivo de configuración del sistema base en la memoria PROM para que cuando se energice el sistema se configure automáticamente. El procedimiento detallado de los pasos anteriores se puede consultar en el apéndice A de [5].

Se debe aclarar que los nodos FPGAs no tienen disco duro, por lo tanto no tienen en donde almacenar el sistema operativo, pero con el trabajo realizado hasta este momento cada nodo FPGA tiene la capacidad de arrancar el sistema operativo de forma remota (función del bootloader). Luego se compiló el kernel de linux en el nodo maestro (usando un compilador cruzado) para la arquitectura PowerPC440 (ver figura 4.7) y se guardaron los ejecutables

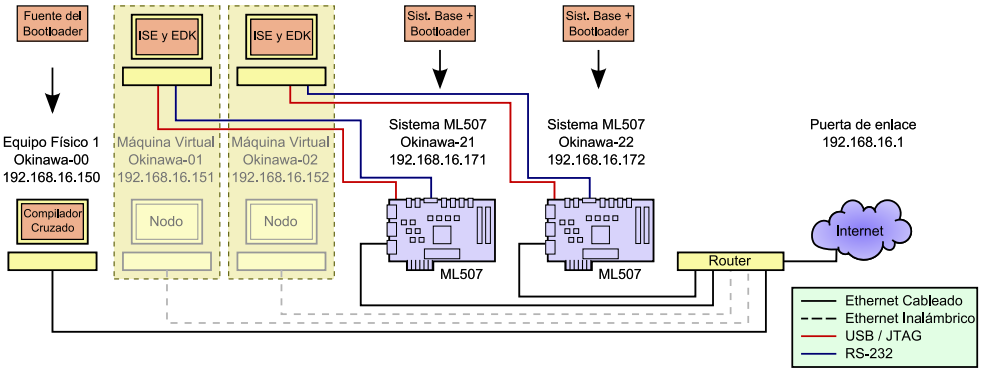


Figura 4.6: Elementos involucrados en la instalación del bootloader (tomada de [5]).

generados en el servidor TFTP. De esta forma cada nodo FPGA a través del bootloader y de la herramienta TFTP puede cargar el *Embedded Linux* gracias al sistema de archivos generado por el ELDK. Finalmente se verificó el correcto funcionamiento del *Embedded Linux* sobre cada nodo FPGA reiniciando las tarjetas.

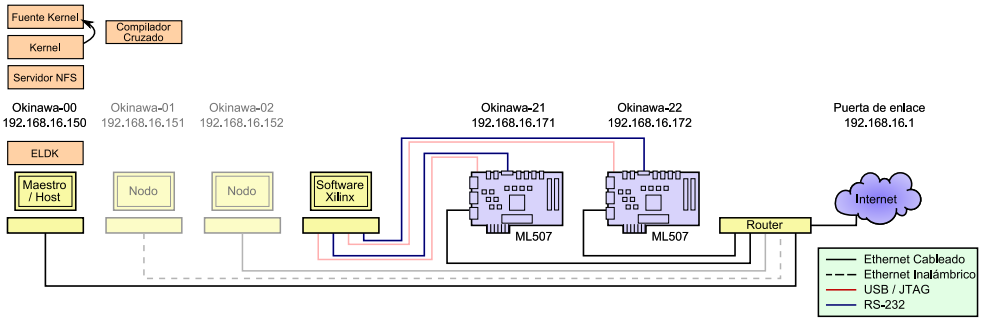


Figura 4.7: Elementos involucrados en la compilación del kernel (tomada de [5]).

4.2.1.4 Creación del sistema de archivos para debian

Como se mencionó anteriormente el kernel que tomamos como referencia tiene el sistema de archivos propio del ELDK (Embedded Linux Development Kit), el cual es diferente de la distribución de Linux Debian 5.0. y como sobre los tres nodos de PPG del Cluster tenemos instalado Linux Debian 5.0, el siguiente paso fue modificar el sistema de archivos del ELDK por el típico de Debian, pensando en la compatibilidad del sistema. El procedimiento realizado fue:

- Descargar los paquetes y crear la estructura base de los directorios.
- Montar desde la tarjeta el sistema de archivos de debian sobre un directorio del *Embedded Linux*.
- Desde el nodo maestro se modificaron los archivos de configuración del sistema de desarrollo y se reinicio la tarjeta.

Un resumen de este proceso se puede apreciar en la Figura 4.8.

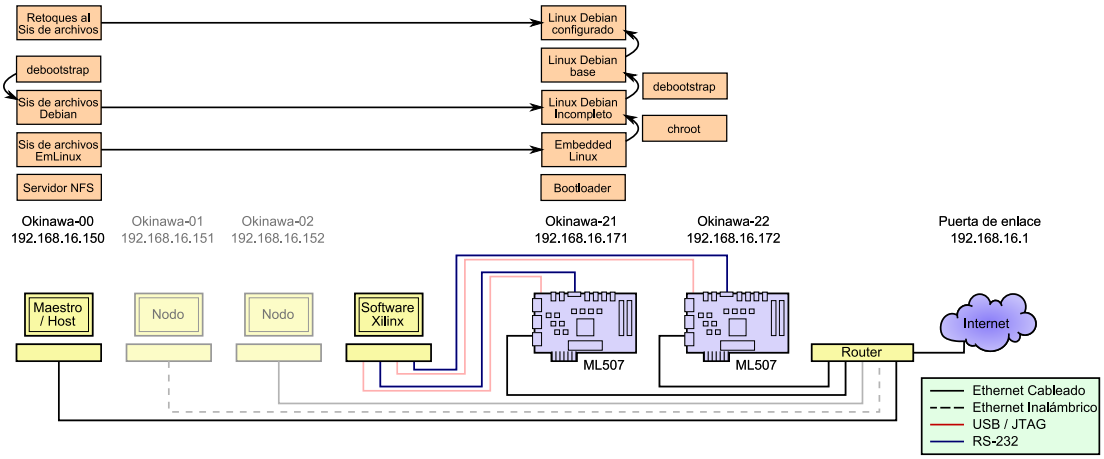


Figura 4.8: Elementos involucrados en la creación del sistema de archivos para debian (tomada de [5]).

4.2.2 Configuración de los nodos para que puedan trabajar como un Cluster.

Luego de haber instalado el S.O. Linux Debian 5.0 en cada uno de los nodos del Cluster Okinawa, se configuraron los siguientes aspectos generales:

- **Repositorio del S.O.:** Como se mencionó anteriormente sobre cada uno de los nodos se hizo la instalación básica del S.O. linux, por lo tanto es necesario configurar un repositorio al cual cada nodo pueda acceder cada vez que se le solicite la instalación de algún paquete especial. Por ello se configuró cada nodo para trabajar sobre el mismo repositorio. El repositorio elegido fue el oficial de linux Debian 5.0.
- **Sincronización :** Se sincronizaron los relojes de todo el sistema.
- **Servidor DNS :** Se actualizó el servidor DNS que se encuentra en el nodo maestro con la tabla de direcciones IP y nombres de dominios de los cuatro nodos esclavos.
- **Paquetes NFS :** Se configuró el nodo maestro como servidor NFS (Network File System) y los demás como clientes NFS. Esto con el propósito de poder compartir una carpeta en la red para que todos los nodos puedan enviar, recibir y editar archivos de trabajo común.

Con la configuración de los aspectos mencionados anteriormente, el Cluster quedó preparado para comenzar con la instalación de algunas herramientas de trabajo para Clusters heterogéneos como PVM (Parallel Virtual Machine) y LAM-MPI (Local Area Multicomputer Message Passing Interface).

4.2.2.1 Instalación de las herramientas sobre el Cluster.

Como Okinawa es un Cluster heterogéneo, para poder instalar cualquier herramienta sobre cada uno de los nodos es necesario:

- Descargar la fuente de la herramienta.
- Compilar dicha fuente sobre el nodo maestro (Arquitectura x86).

- Compilar nuevamente la fuente sobre un nodo FPGA (Arquitectura PowerPC 440).
- Instalar los archivos compilados x86 sobre Okinawa-00, Okinawa-01 y Okinawa-02.
- Instalar los archivos compilados PowerPC 440 sobre Okinawa-21 y Okinawa-22.
- Configurar la herramienta con la información del Cluster.

Luego de instalar la herramienta se procede a realizar un script de prueba, el cual debe ser compilado sobre cada uno de los nodos. Finalmente se lanza la aplicación sobre todo el Cluster.

El procedimiento previo se hizo tanto para PVM como para LAM-MPI, funcionando correctamente. Con el trabajo realizado hasta esta parte del proceso, el Cluster Okinawa quedo en capacidad de lanzar aplicaciones en paralelo sobre cada uno de sus nodos.

4.3 Instalación de una herramienta de monitorización sobre el cluster Okinawa

La herramienta elegida para monitorizar el desempeño del cluster Okinawa fue Ganglia, debido a que es de código abierto, gratuita, esta muy bien documentada y permite medir el desempeño del Cluster en tiempo real.

4.3.1 Como funciona ganglia

Ganglia en su estructura esta compuesta por dos demonios (programas) llamados gmond, gmetad y el Ganglia Web Frontend (como se puede ver en la figura 4.9); los cuales interactuan de la siguiente manera:

- **Gmond** (Ganglia Monitoring Daemond): Es un demonio multi-hilo el cual corre sobre cada uno de los nodos del Cluster que se desean monitorear. Su función es recolectar la información de las métricas de interés para luego enviarla de forma periódica al gmetad.
- **Gmetad** (Ganglia Meta Daemon):Es el demonio encargado de recibir la información recopilada por los gmond en intervalos regulares de tiempo. La información llega en formato XML y este demonio se encarga de almacenarla en una base de datos Round-Robin, la cual luego comparte con el servidor web.
- **Ganglia Web Frontend**: Es el encargado de presentar la información contenida en la base de datos Round-Robin mediante una página web dinámica. Esta visualización se realiza como un HTML en tiempo real.

Por lo tanto se instaló sobre todos los nodos del Cluster el demonio *gmond* y sobre el nodo maestro el demonio *gmetad* junto con el *web frontend* como se observa en la Figura 4.10. El procedimiento realizado durante la instalación se tomó de [16] y de forma general se resume así:

- Instalación de las herramientas *rrdtool* y *PHP5* en el nodo maestro.
- Instalación del ganglia monitor *(gmond)* sobre todos los nodos del Cluster.
- Instalación del *gmetad* sobre el nodo maestro.

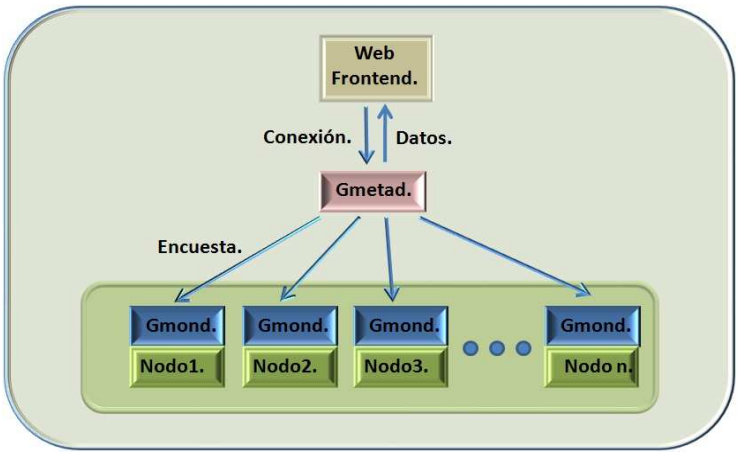


Figura 4.9: Esquema de trabajo de Ganglia.

- Descargar la fuente de ganglia para luego compilarse sobre el nodo maestro.
- Configurar y reiniciar el servidor *Apache* en el nodo maestro.

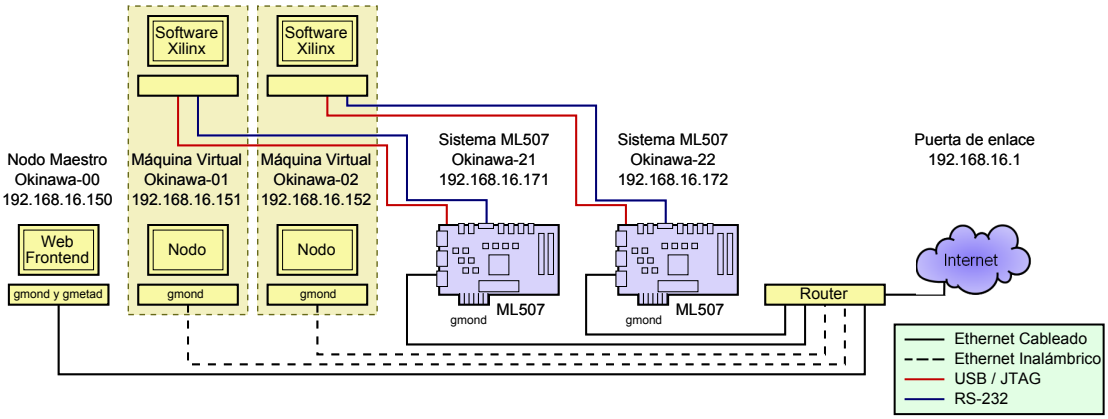


Figura 4.10: Instalación de ganglia sobre el Cluster Okinawa.

Luego de reiniciar el servidor *Apache*, a través del explorador de Okinawa-01 o Okinawa-02 se consultó la interfaz gráfica de ganglia por medio de la dirección *IP* del nodo maestro (192.168.16.150). La imagen obtenida en ese momento fue similar a la Figura 4.11.

Después de verificar su correcto funcionamiento, se procedió a modificar su apariencia para que la interfaz gráfica fuera más adecuada al montaje físico de Okinawa Cluster.

4.3.2 Personalización de ganglia

Las primeras modificaciones hechas al servidor web de ganglia cambiaron algunos aspectos de su apariencia (como logos y colores de fondo) y redistribuyeron los nodos por arquitecturas como se puede observar en las Figuras 4.12

Figura 4.11: Ganglia sobre el Cluster Okinawa (Web frontend).

y 4.13. Con las primeras pruebas de desempeño del Cluster se observó que esta herramienta nos permite medir 37 métricas (ver Tabla 4.2) sobre cada uno de los nodos de Okinawa con algunas limitantes. Como ganglia ofrece una visión global del cluster tiene la limitación de no poder diferenciar entre el desempeño interno de cada uno de los procesos que se están ejecutando dentro de los nodos. Y como se desea medir el desempeño que alcanza el FPGA cuando ejecuta un proceso en particular (por ejemplo la migración sísmica) fue necesario mejorar este aspecto de ganglia por medio de un parche elaborado por [17] el cual permite desglosar el desempeño de un nodo por procesos. Esta herramienta es conocida como *Gappmon*.

El sistema completo consiste de dos programas llamados *gappmon* y *gcleanup* los cuales realizan el seguimiento de un proceso y la limpieza de la base de datos respectivamente. Adicionalmente cuenta con un paquete de actualización de la interfaz web llamado *gappmon web frontend* que permite ver la nueva información enviada por *gappmon*. Para hacer uso de esta herramienta se descargan las fuentes sobre cada nodo y se realiza el procedimiento indicado en [17]. Luego de instalada, se debe conocer el nombre del proceso al que se le desea hacer seguimiento y se utiliza de la siguiente manera (cshot1 es un ejemplo de nombre de proceso y el & es para que libere la consola):

- *gappmon cshot1 &*

#	Métrica	Descripción
1	boottime	Fecha y hora del arranque del sistema.
2	bytes_in	Número de bytes que entran por segundo.
3	bytes_out	Número de bytes que salen por segundo.
4	cpu_aidle	Porcentaje de tiempo inactivo desde el arranque de la CPU.
5	cpu_idle	Porcentaje de inactividad de la CPU.
6	cpu_nice	Porcentaje de CPU agradable.
7	cpu_num	Número de CPUs.
8	cpu_report	Reporte de uso de la CPU.
9	cpu_speed	Velocidad en MHz de la CPU.
10	cpu_system	Porcentaje de uso de la CPU del sistema.
11	cpu_user	Porcentaje de uso de la CPU del usuario.
12	disk_free	Espacio del disco libre.
13	disk_total	Espacio total del disco.
14	gexec	Sistema de ejecución remota escalable para cluster.
15	load_fifteen	Carga promedio en los últimos 15 minutos.
16	load_five	Carga promedio en los últimos 5 minutos.
17	load_one	Carga promedio en el último minuto.
18	load_report	Reporte de carga.
19	machine_type	Arquitectura de cada nodo.
20	mem_buffers	Cantidad de memoria en buffer.
21	mem_cached	Cantidad de memoria en Cache.
22	mem_free	Cantidad de memoria disponible.
23	mem_report	Reporte de todo el uso de memoria.
24	mem_shared	Cantidad de memoria compartida.
25	mem_total	Memoria total.
26	mtu	Unidad de transmisión máxima de red.
27	network_report	Reporte del uso de la red.
28	os_name	Nombre del sistema operativo.
29	os_release	Nombre de la versión.
30	part_max_used	Máximo porcentaje usado para todas las particiones.
31	pkts_in	Paquetes que ingresan por segundo.
32	pkts_out	Paquetes que salen por segundo.
33	proc_run	Número total de procesos que están corriendo.
34	proc_total	Número total de procesos.
35	swap_free	Cantidad de memoria *swap* disponible.
36	swap_total	Cantidad total de memoria *swap*.
37	sys_clock	Frecuencia de trabajo del procesador.

Tabla 4.2: Las métricas que ofrece ganglia.

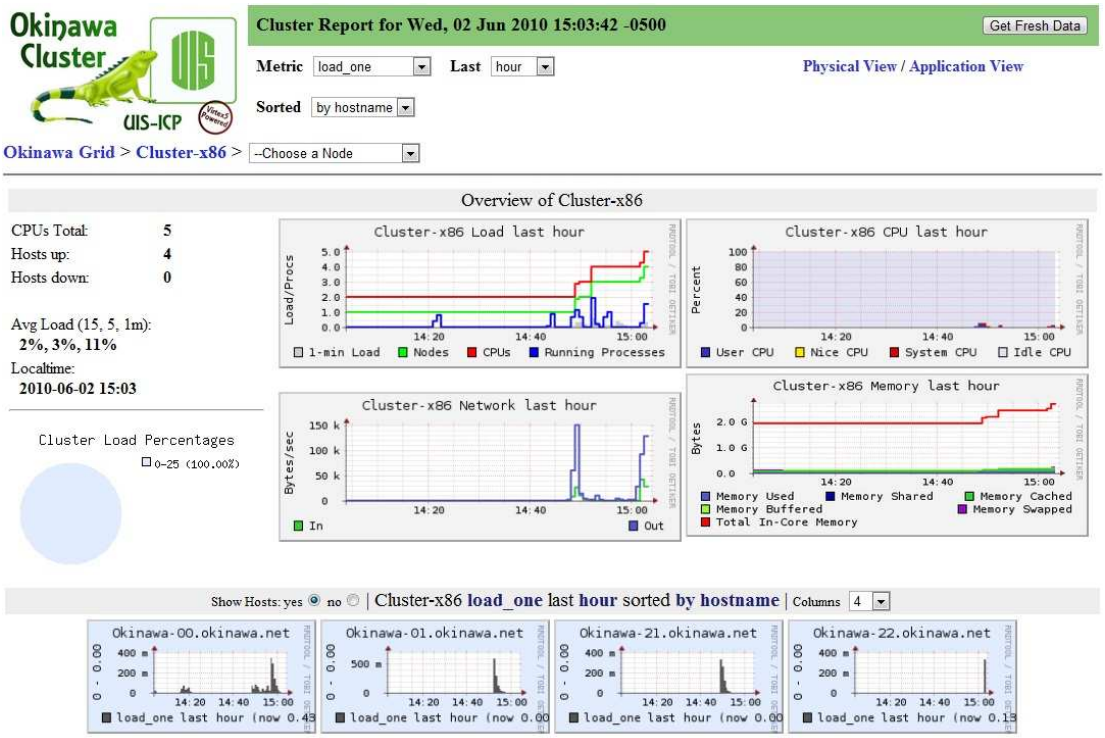


Figura 4.12: Ganglia personalizado sobre el Cluster Okinawa.

Cuando se ejecuta esta línea, en la consola se imprime el mensaje *monitoring execution of "cshot1"(ID)....*, con esto se da inicio a la aplicación que permite monitorear el desempeño de un proceso en particular. Para verificar que el proceso de monitorización comenzó se puede ver en la interfaz de red de ganglia en la pestaña *Application View* y aparece el nombre del proceso como se muestra en la Figura 4.14. Al hacer click sobre el nombre del proceso al que se le está haciendo seguimiento nos lleva a las características de desempeño del mismo especificandonos sobre cuales nodos se está ejecutando (ver Figura 4.15). Al dar click sobre un nodo específico la interfaz nos muestra las gráficas de las métricas aplicadas pero ahora sobre el proceso que estamos siguiendo. Cuando el proceso termina de ejecutarse en el nodo, automáticamente el demonio *gappmon* finaliza su ejecución. Cabe aclarar que esta herramienta permite medir el desempeño de varios procesos de forma simultánea.

Otra ventaja que ofrece esta herramienta es que permite hacer un seguimiento sobre las métricas para intervalos de tiempo más pequeños que los valores iniciales de ganglia. En la Figura 4.16 podemos apreciar que los nuevos intervalos de tiempo son 1, 3, 5, 10, 15 y 30 minutos.

Con la instalación de *gappmon* sobre ganglia el sistema de monitorización del Cluster Okinawa es finalizado, dando paso a las fases de interconexión física y desarrollo de los drivers.

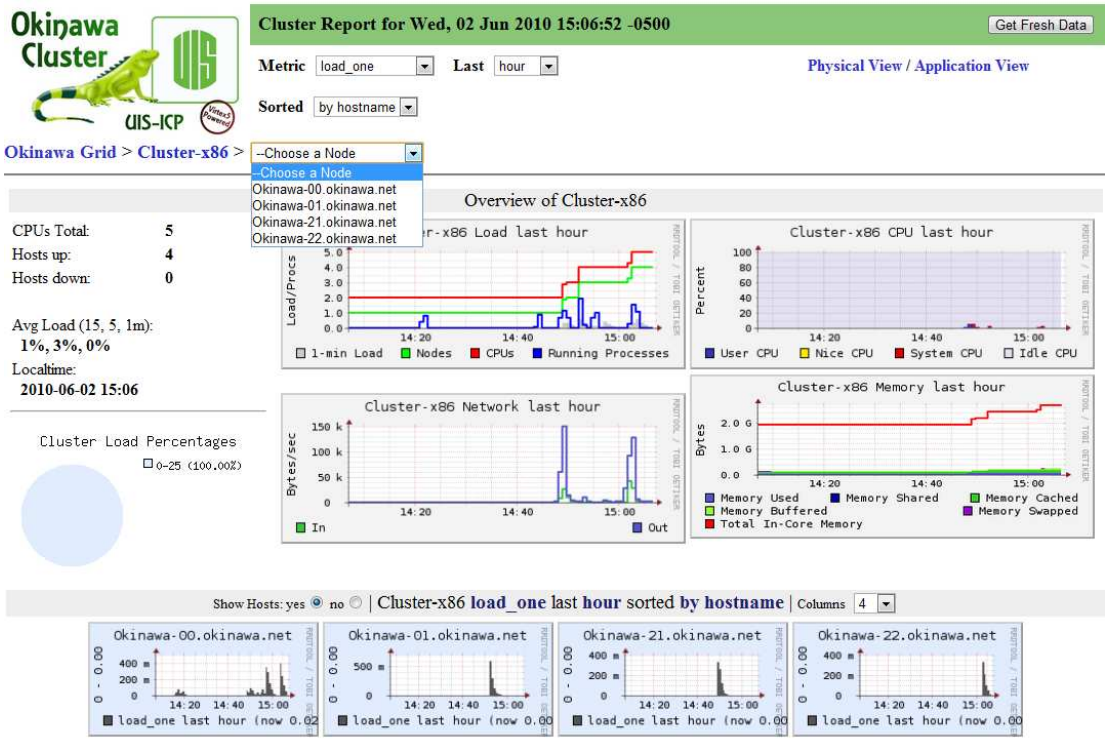


Figura 4.13: Ganglia personalizado sobre el Cluster Okinawa (Nodos).

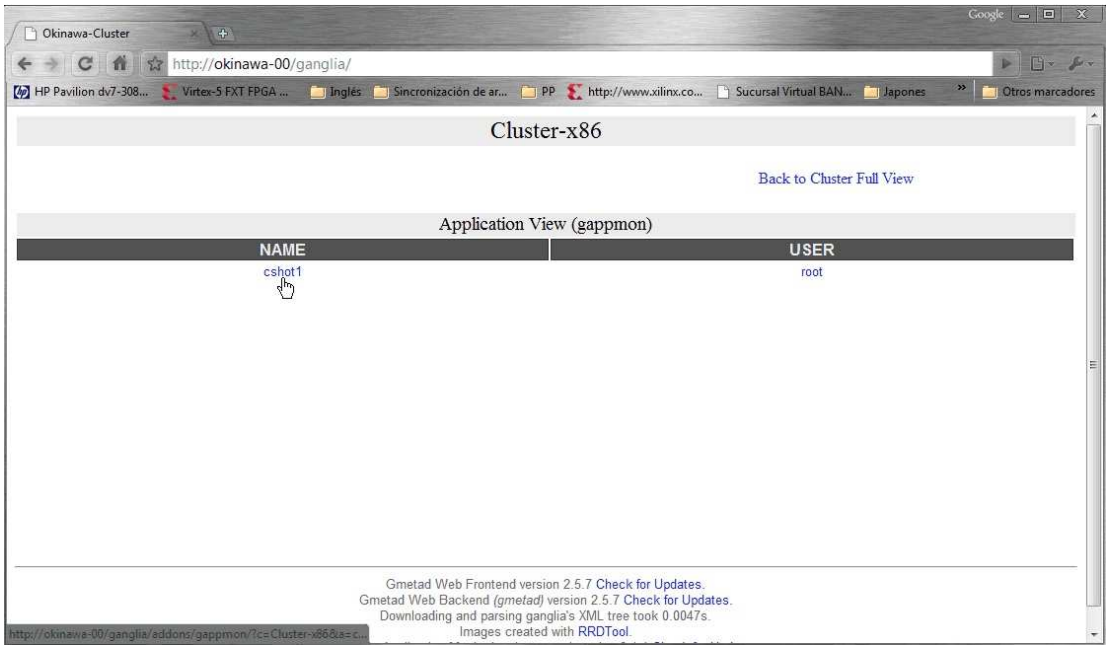


Figura 4.14: Ganglia trabajando junto con gappmon.

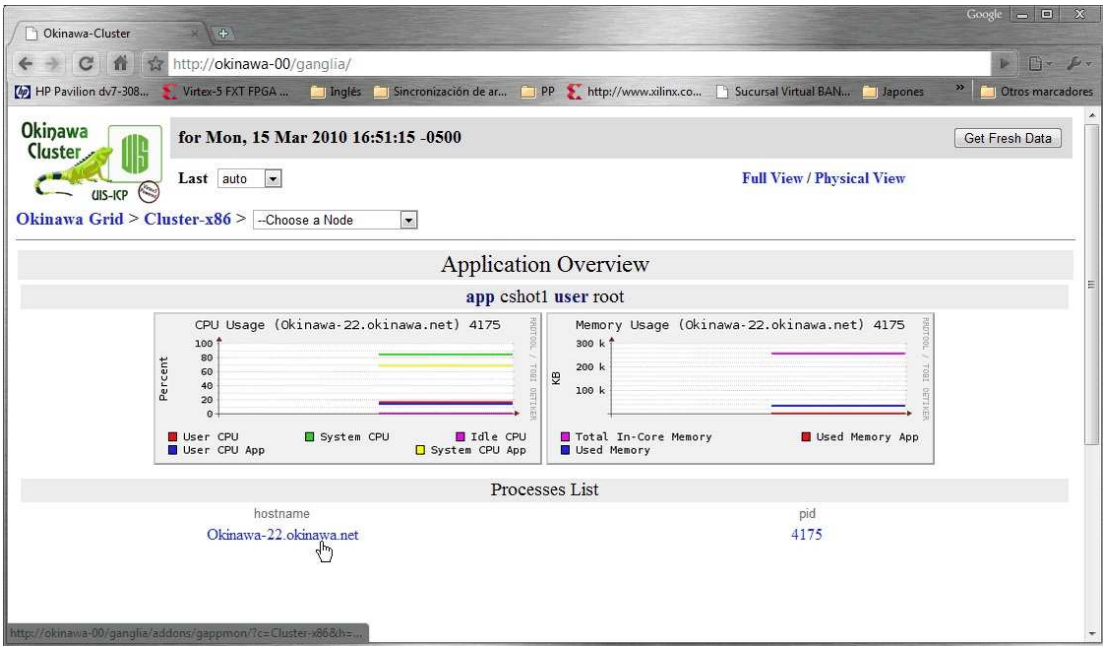


Figura 4.15: Ganglia trabajando junto con gappmon parte 2.

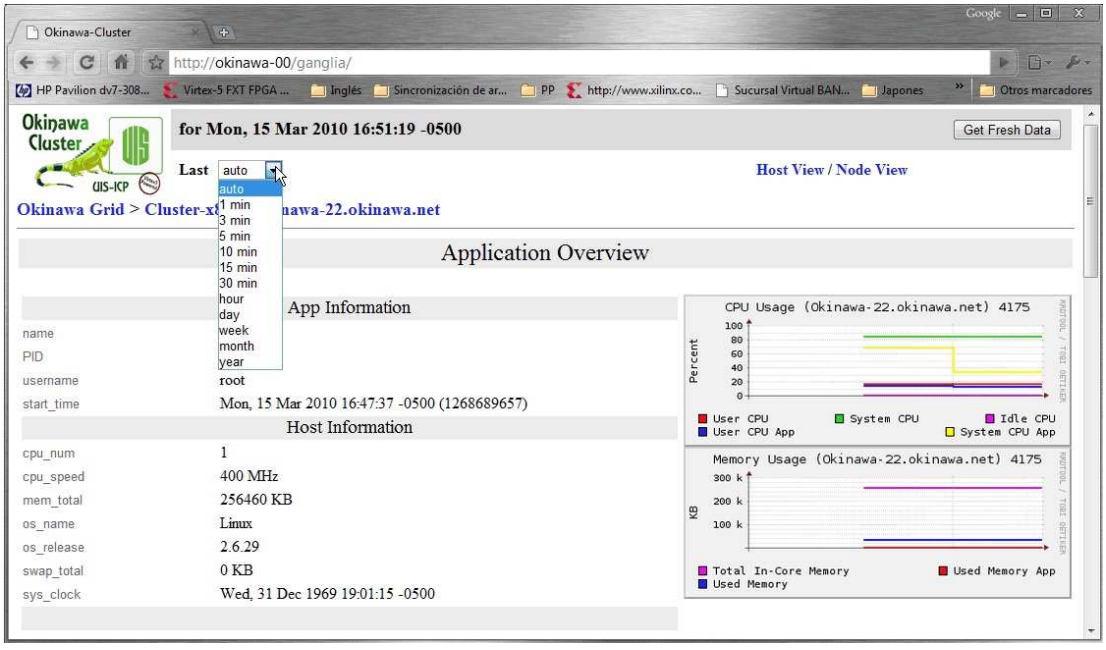


Figura 4.16: Ganglia trabajando junto con gappmon parte 3.

4.4 Interconexión del periférico con el PowerPC.

Para la interconexión del procesador específico con el sistemas base (como se muestra en la Figura 4.17) se utilizó la herramienta *Embedded Development Kit* (EDK) de *Xilinx*. La metodología usada es la que plantea el laboratorio tres del *Xilinx University Program* (XUP) llamado *"Lab3: Adding Custom IP to an Embedded System"* [18] y a continuación se resume el procedimiento realizado resaltando las partes más importantes del proceso de interconexión.

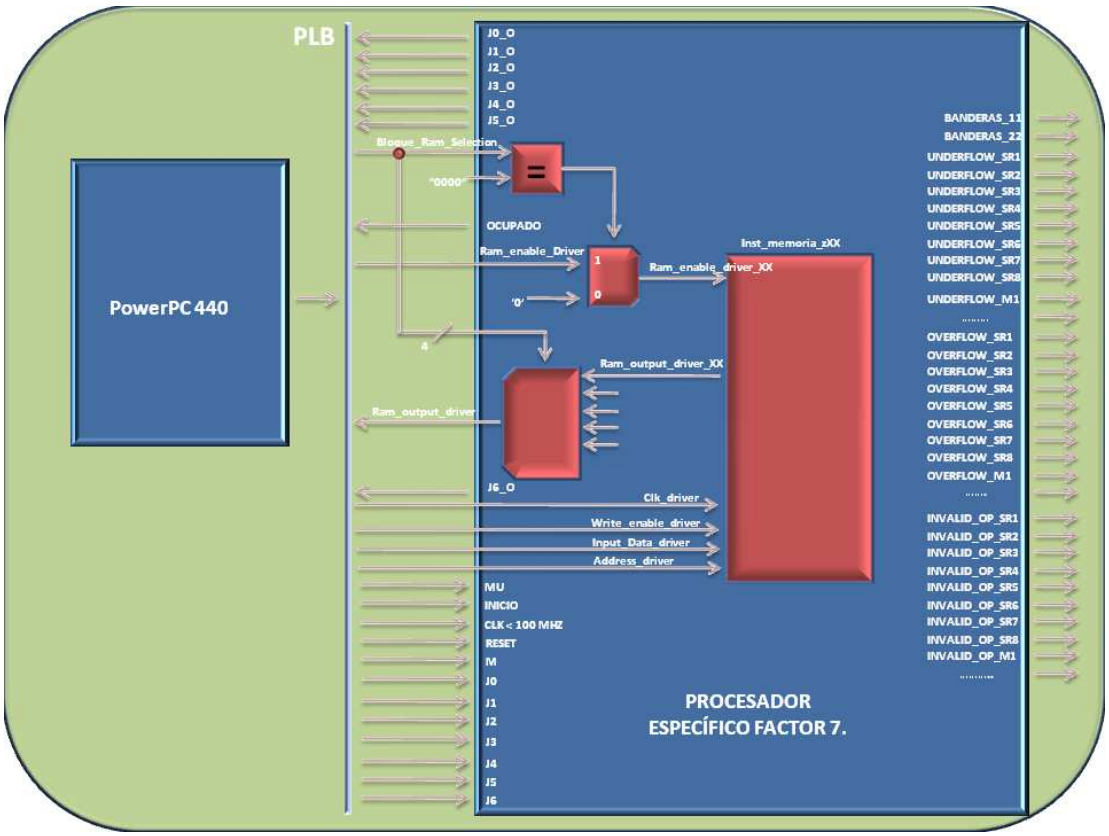


Figura 4.17: Diagrama de interconexión final entre el PPC y el PPE.

4.4.1 Generando la plantilla del periférico

Para agregarle un periférico al sistema base se crea una plantilla que permita hacer esta interconexión. La modificación que se tuvo que realizar en esta primera etapa fue en el paso ocho de [18], debido a que la guía sugiere agregar solo un registro a la plantilla y para nuestro caso es necesario tener 19 registros para poder interactuar de forma correcta con el procesador específico. La relación de los registros con las entradas del procesador específico se puede apreciar en las Tablas 4.3, 4.4 y 4.5.

De estas tablas se puede ver que:

- Cada uno de los registros es de 32 bits.
- Los registros que van desde slv_reg1 hasta slv_reg16 representan directamente las entradas ó salidas del procesador.
- Los registro slv_reg0 y slv_reg18 son los registros de configuración del procesador específico.
- El registro slv_reg17 es el registro de depuración del procesador específico.

			Bits de cada registro					
User_Logic_Register	**# de Registros.**	**Nombre**	**0**	**1**	**2**	**3**	**4**	**5**
Numeración del procesador								
slv_reg0	1	Control.	**No conectados**					
slv_reg1	2	Ram_output_Driver.						
slv_reg2	3	Input_Data_Driver.						
slv_reg3	4	j0.						
slv_reg4	5	j1.						
slv_reg5	6	j2.						
slv_reg6	7	j3.						
slv_reg7	8	j4.						
slv_reg8	9	j5.						
slv_reg9	10	j6.						
slv_reg10	11	j0_o.						
slv_reg11	12	j1_o.						
slv_reg12	13	j2_o.						
slv_reg13	14	j3_o.						
slv_reg14	15	j4_o.						
slv_reg15	16	j5_o.						
slv_reg16	17	j6_o.						
slv_reg17	18	Banderas-Debug.	**Banderas22**					
slv_reg18	19	M.	**No conectado**					

Tabla 4.3: Registros de la interconexión.

	Bits de cada registro														
Nombre	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
					4	**3**	**2**	**1**	**0**				**3**	**2**	**1**
Control.	**No conectados**				**Estado_sig_out**					**Ocupado**	**Reset**	**Inicio**	**MU**		
Ram_output_Driver.															
Input_Data_Driver.															
j0.															
j1.															
j2.															
j3.															
j4.															
j5.															
j6.															
j0_o.															
j1_o.															
j2_o.															
j3_o.															
j4_o.															
j5_o.															
j6_o.															
Banderas-Debug.	**Banderas22**										**Banderas11**				
M.	**No conectados**						**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**

Tabla 4.4: Registros de la interconexión (parte 2).

4.4.2 Creando el periférico

Luego de crear la plantilla de interconexión, la herramienta *EDK* genera un grupo de archivos dentro de los cuales el usuario debe identificar y modificar la estructura de algunos de ellos (dependiendo de las necesidades del periférico

	Bits de cada registro										
Nombre	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	0	**3**	**2**	**1**	**0**			**3**	**2**	**1**	**0**
Control.	**MU**	**Address_driver.**				**Write_enable_Driver**	**Ram_enable_Driver**	**Bloque_Ram_Selection**			
Ram_output_Driver.											
Input_Data_Driver.											
j0.											
j1.											
j2.											
j3.											
j4.											
j5.											
j6.											
j0_o.											
j1_o.											
j2_o.											
j3_o.											
j4_o.											
j5_o.											
j6_o.											
Banderas-Debug.	**Banderas11**										
M	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Tabla 4.5: Registros de la interconexión (parte 3).

que se le esta agregando al sistema base). Para nuestro caso los archivos que se modificaron fueron:

- **Procesador_factor_7_v2_1_0.pao**: En este archivo se le especifica a la herramienta *EDK* los nombres de las fuentes .vhd o .v que contienen la descripción en *Hardware* del periférico que se agregó al sistema base. Se debe aclarar que las fuentes agregadas a este archivo se deben guardar en *pcores/nombre_del_periférico/HDL/vhd/* y en *pcores/nombre_del_periférico/HDL/verilog/*. El archivo con la modificación se puede ver en el apéndice H en el listado H.1.
- **Procesador_factor_7_v2_1_0.mpd**: En este archivo se definen los puertos de salida del periférico. Tomando como referencia la Figura 4.17 y las Tablas 4.3, 4.4 y 4.5 se concluye que la única salida externa del periférico debía ser la de **CLK < 100MHz** debido a que la frecuencia del bus del sistema es de 100MHz. Por lo tanto a este archivo solo se le agregó la línea dos que se muestra en el script H.2.
- **Procesador_factor_7.vhd**: Se confirman los puertos adicionados en el *.mpd. La modificación se muestra en el script H.3.
- **User_logic.vhd**: Se hace la instancia del procesador específico sobre la plantilla. En este archivo es en el que se realiza la interconexión física de la plantilla con el procesador específico como se describe en las Tablas 4.3, 4.4 y 4.5. La modificación final del mismo se muestra en el script H.4.

Finalmente el último paso consiste (no mencionado en el laboratorio 3 [18]) en agregar los archivos *.ngc de los módulos de suma, resta y multiplicación de punto flotante generados con el *Core Generator* a la ruta *nombre_del_proyecto/implementación/*. De esta forma el procesador específico se encuentra disponible para ser conectado físicamente al bus del sistema base.

4.4.3 Agregando el periférico al sistema base.

Para poder conectar de forma correcta el periférico al bus del sistema base se debe importar el módulo generado y se debe configurar la conexión de la siguiente manera:

1. Conectar el módulo como esclavo del bus local (SPLB).
2. Asignarle un tamaño de memoria de 64K y por medio del generador de direcciones, conseguirle una dirección al periférico. Para nuestro caso la dirección asignada va desde **814C0000** hasta **814CFFFF**.
3. Conectar el puerto **CLK< 100MHz** a una señal de reloj de 60MHz. Para ello se le agregó al sistema base un "*PLL for PPC440 Clocks*" y se configuró para que a partir de la señal de reloj de 100MHz del bus del sistema generara una señal de reloj de 60MHz. La conexión entre el PLL y el procesador específico se puede ver en la Figura 4.18.

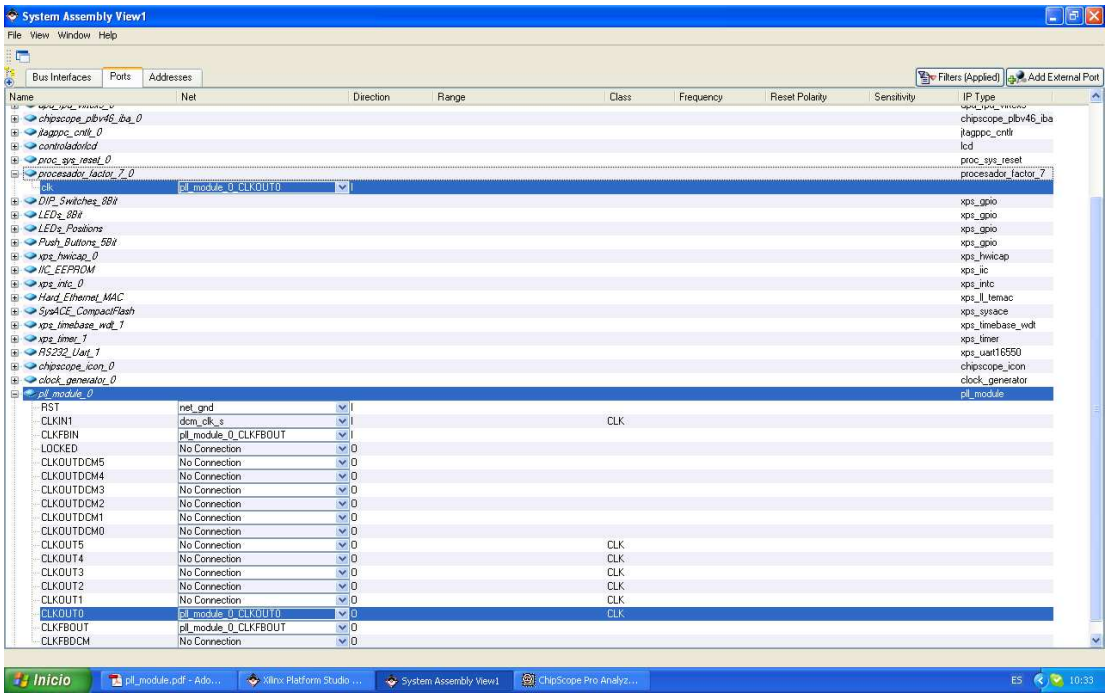


Figura 4.18: Conexión de la señal de reloj del procesador específico.

Finalmente en la herramienta *EDK* se selecciona **Device Configuration → Update Bitstream** para que esta revise la interconexión y ejecute la integración del procesador específico al sistema base.

4.4.4 Agregando el módulo de depuración.

Antes de descargar el *bitstream* completo (sistema base + procesador específico) dentro del FPGA, se decidió agregarle un módulo de depuración al sistema. Este módulo consiste en un **Analizador de Bus Integrado** (IBA) el cual se conecta al bus local del procesador (PLB) para monitorear en tiempo real todas las señales que se están transmitiendo y de esta forma poder revisar la interacción entre el procesador específico y el sistema base.

Para poder hacer la inserción del módulo de depuración dentro del sistema base se utilizó la herramienta de **Xilinx** llamada *chipscope* y se usó como guía el laboratorio seis de **XUP** llamado *HW/SW System Debug*. Después

de que se realizó esta inserción se procedió a configurar el FPGA con el *bitsream* final (sistema base + procesador específico + IBA) culminando con el trabajo de *Hardware* del proyecto.

4.5 Elaboración de la Plantilla del Driver

En esta etapa del proyecto se requirió crear una herramienta que permitiera acceder y administrar los periféricos, esta herramienta se conoce como driver. Como los drivers son muy dependientes de las características de cada periférico, se decidió hacer una plantilla del driver para poder facilitar el acople de futuros periféricos al sistema [19]. A continuación se explica el contenido de la plantilla del driver.

4.5.1 Funcionamiento del Driver.

Para entender más fácilmente el funcionamiento de la plantilla, se explicará brevemente como es la interacción del driver con el kernel de Linux, partiendo de su forma de uso [19].

Si observamos el Script 4.1 el primer paso es compilar la fuente de la plantilla del driver (archivo en **C**) por medio del comando **make**, el cual usa el archivo **Makefile** mostrado en el Script 4.2 para compilar la fuente a través de **gcc**. Como resultado de esta compilación se genera el módulo driver.ko como se puede observar en la Figura 4.19.

```
1  Okinawa-21: make
2  Okinawa-21: ./montar.sh
3  Okinawa-21: lsmod
4  Module  Size  Used by
5  primerdriver 5424 0
6  Okinawa-21: ./enviar.sh
7  ++ echo ciao
8  Okinawa-21: ./recibir.sh
9  ++ cat /dev/LCD
10 ciao
11 Okinawa-21: rmmod primerdriver
12 Okinawa-21: lsmod
13 Module  Size  Used by
14 Okinawa-21:
```

Script 4.1: Uso del driver

```
1  obj-m   += primerdriver.o
2
3  all:
4          make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
5  #       make -C /usr/src/linux-headers-2.6.29-powerpc M=$(PWD) modules
6  #       make -C /compartido/powerpc/linux-2.6-xlnx.git-Okinawa-21 M=$(PWD) modules
7
8  clean:
9          make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
10 #       make -C /usr/src/linux-headers-2.6.29-powerpc M=$(PWD) clean
11 #       make -C /compartido/powerpc/linux-2.6-xlnx.git-Okinawa-21 M=$(PWD) clean
```

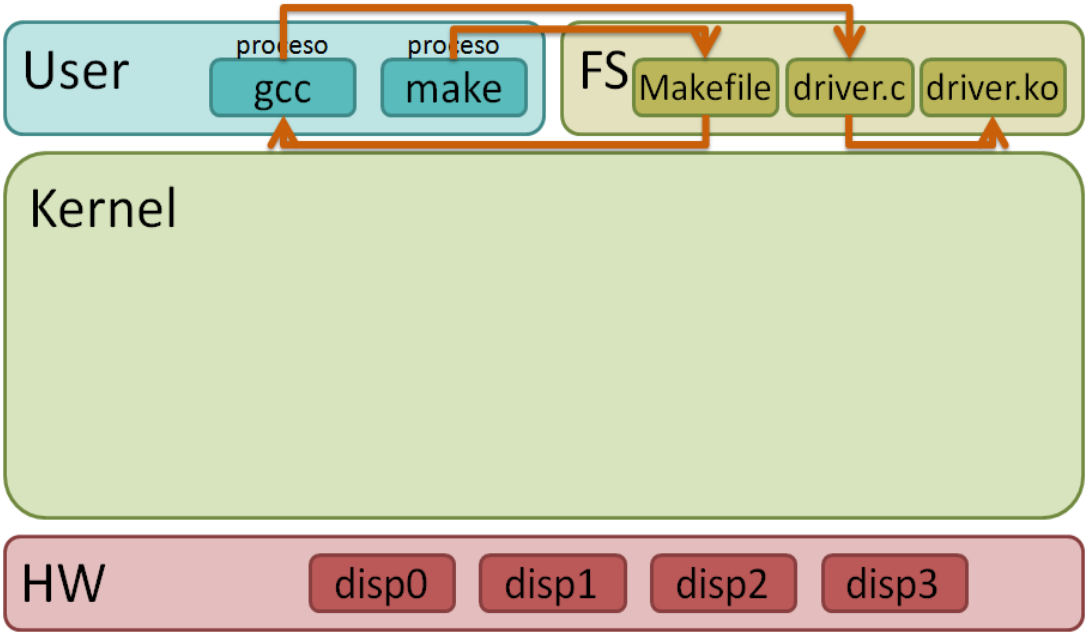


Figura 4.19: Proceso de compilación de la fuente del driver (tomada de [5]).

Script 4.2: Makefile

Luego de generado el módulo driver.ko se procede al montaje del mismo dentro del kernel por medio del Script 4.3. Este Script usa la función **insmod** para añadir el módulo al kernel, el cual ejecuta en ese instante la rutina init del driver para hacer su registro por medio de los **major y minor numbers**[1]. Después de que el driver está registrado correctamente los **major and minor numbers** se pueden consultar en **/proc/devices** como se muestra en la Figura 4.20. Adicionalmente la rutina init crea la estructura driver_cdev, la cual se añade al kernel y permite que las funciones del driver sean invocadas como se muestra en la Figura 4.21. Con ésto tanto la rutina init como el proceso **insmod** dan por terminada su participación.

Así el driver queda a la espera de que lo usen, pero para ello necesita un vínculo entre el espacio del kernel y el espacio del usuario. Este vínculo normalmente se logra mediante la creación de un nodo imagen en el sistema de archivos. Por lo tanto el siguiente paso es crear dicho nodo usando el comando **mknod**.

```
1 #!/bin/sh
2 #set -x
3 module="primerdriver"
4 device="LCD"
5 mode="664"
6
```

[1]el **major number** se usa para identificar el driver asociado con el dispositivo mientras que el **minor number** es usado por el kernel para determinar exactamente cuales dispositivos están siendo referidos.

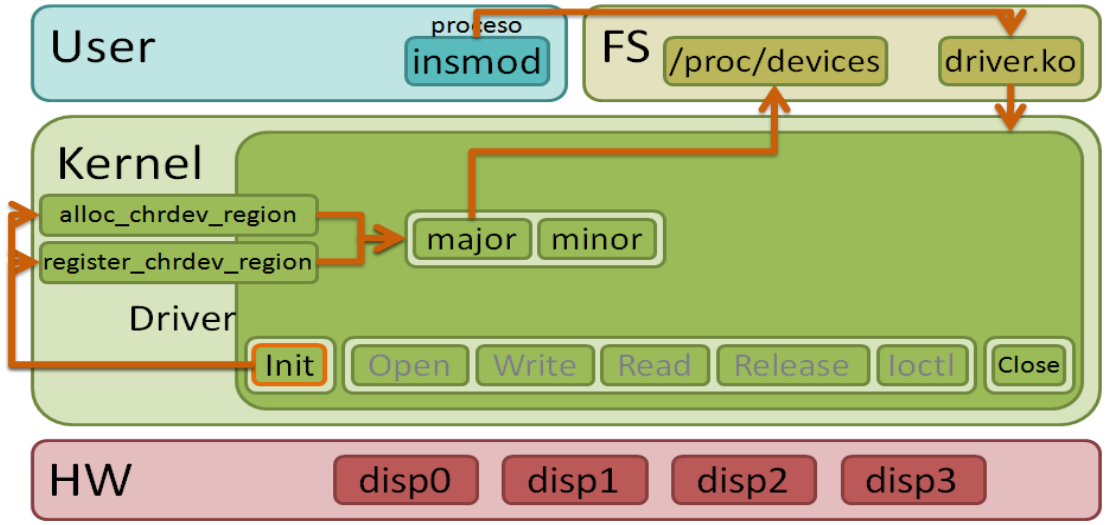


Figura 4.20: Montaje del driver en el kernel de Linux parte 1 (tomada de [5]).

```
7  rm -f /dev/${device}
8  # invoke insmod with all arguments we got
9  # and use a pathname, as newer modutils don't look in . by default
10 /sbin/insmod ./$module.ko $* || exit 1
11
12 # remove stale nodes
13 rm -f /dev/${device}
14 # major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)
15 major=$(cat /proc/devices | awk '{if($2=="LCD") print $1 ; else print pailas}' | grep -v pailas)
16 mknod /dev/${device} c $major 0
17 #mknod /dev/${device}1 c $major 1
18 #mknod /dev/${device}2 c $major 2
19 #mknod /dev/${device}3 c $major 3
20
21 # give appropriate group/permissions, and change the group.
22 # Not all distributions have staff, some have "wheel" instead.
23 #group="staff"
24 #grep -q '^staff:' /etc/group || group="wheel"
25 #chgrp $group /dev/${device}[0-3]
26 #chmod $mode  /dev/${device}[0-3]
```

Script 4.3: montar.sh

Cuando este comando se ejecuta, el kernel crea un **inode**[2], el cual queda directamente vinculado con un nodo del sistema de archivos logrando la comunicación entre los dos espacios. Adicionalmente el **inode** queda cargado con el **cdev** del driver, como se puede apreciar en la Figura 4.22. De esta manera el montaje del módulo dentro del kernel de linux realizado por el Script 4.3 finaliza su trabajo. Para poder consultar si el módulo quedo correctamente montado dentro del kernel se puede ejecutar el comando **lsmod** en consola como se puede ver en el Script 4.1.

[2]Es una estructura usada internamente por el kernel para representar archivos del espacio de usuario.

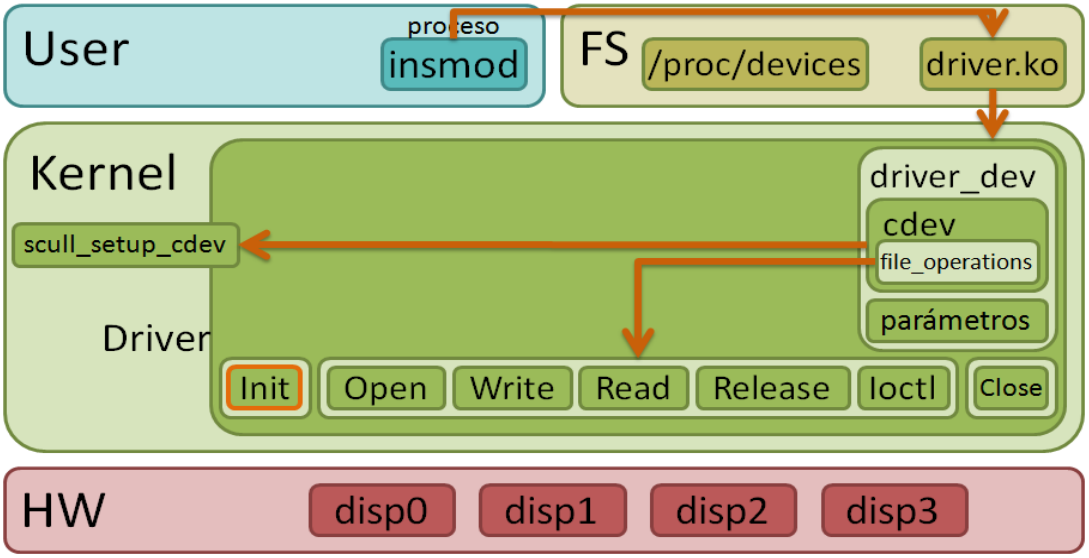


Figura 4.21: Montaje del driver en el kernel de Linux parte 2 (tomada de [5]).

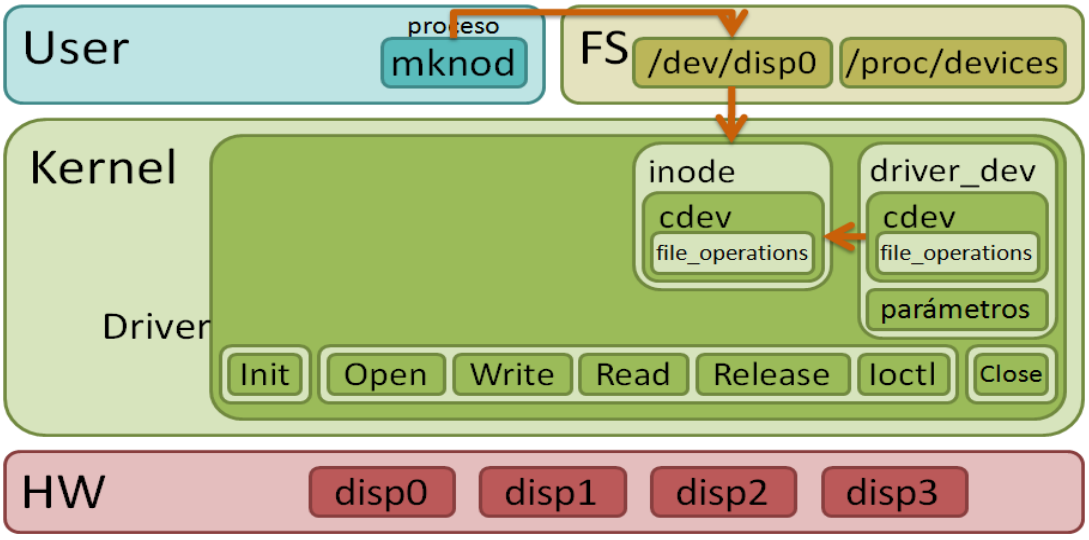


Figura 4.22: Creación del nodo para el uso del driver (tomada de [5]).

Luego de que el módulo queda montado dentro del kernel, se procede a usarse mediante los Scripts 4.4 y 4.5. Claramente se puede apreciar que estos dos Scripts hacen los llamados al sistema de escribir (con el comando **echo** 4.4) y leer (con el comando **cat** 4.5). Como los eventos que suceden dentro del kernel cuando se hace un llamado al sistema son muy similares, solo revisaremos lo que sucede cuando se ejecuta el comando **cat**.

```
1 set -x
2 echo "ciao" > /dev/LCD
```

Script 4.4: enviar.sh

```
1 set -x
2 cat /dev/LCD
```

Script 4.5: recibir.sh

La segunda línea del Script 4.5 permite observar que el nodo del sistema de archivos que esta relacionado con nuestro driver esta en /dev/LCD. Cuando el comando **cat** se ejecuta sobre este nodo la secuencia de eventos que suceden en el kernel son:

1. **cat** genera un llamado al sistema **open**, el cual a su vez crea un **struct file** que apunta a la posición del driver (ver figura 4.23).
2. Luego **cat** genera un llamado al sistema **read**, el cual accede al dispositivo y transfiere los datos mediante la función del kernel **copy to user** al espacio del usuario (ver figura 4.24).
3. Finalmente **cat** genera un llamado al sistema **close**, el cual ejecuta la función **release** para eliminar el **struct file** (ver figura 4.25).

Luego de que el driver es usado, el último paso que queda es liberarlo del kernel, y para ello se debe utilizar el comando **rmmod** como se muestra en la línea 11 del Script 4.1. Cuando este comando se ejecuta en el espacio de usuario, los siguientes eventos ocurren en el kernel:

1. Se ejecuta la función **close** del driver para liberar el **driver_dev** creado.
2. Luego se desengancha el módulo del kernel, dejando únicamente al **inode** que representa la imagen del nodo que se encuentra en el sistema de archivos (ver Figura 4.26).

Luego se puede volver a ejecutar el comando **lsmod** en consola para verificar que el módulo halla sido removido correctamente del kernel (ver líneas 12 y 13 del Script 4.1). Con este último paso se finaliza el proceso estándar para el uso de cualquier driver, dando paso al contenido de la plantilla del driver (en lenguaje C) la cual mediante comentarios va resumiendo la función de cada una de sus partes. Para poder tener un mayor entendimiento de la plantilla se recomienda complementar los comentarios de la misma con los capítulos (1-4, 9) en [19].

4.5.2 Plantilla del Driver.

En el Apéndice I se presenta la plantilla final del driver (va desde el listing I.1 hasta I.14). El funcionamiento de la misma se verificó sobre periféricos I/O del sistema de desarrollo ML507 (como los leds y los dipswitchs)

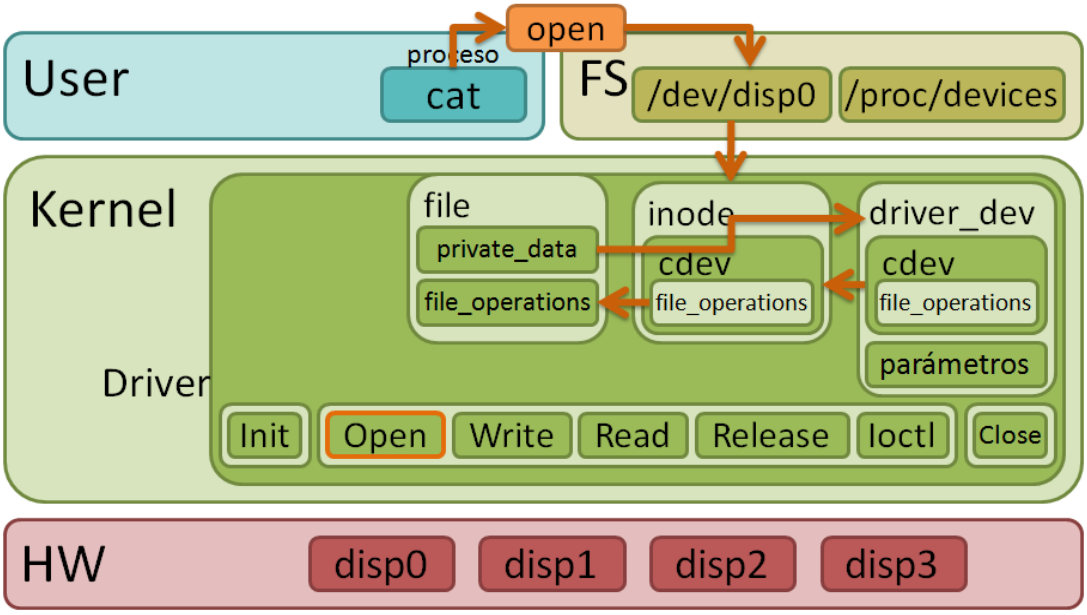


Figura 4.23: Proceso de lectura del driver parte 1 (tomada de [5]).

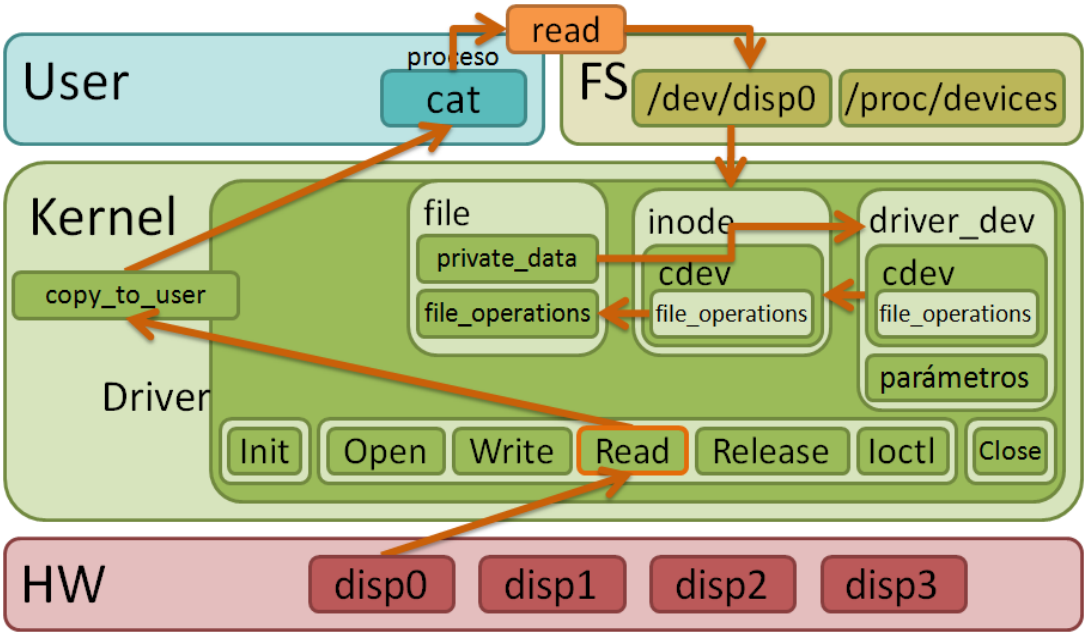


Figura 4.24: Proceso de lectura del driver parte 2 (tomada de [5]).

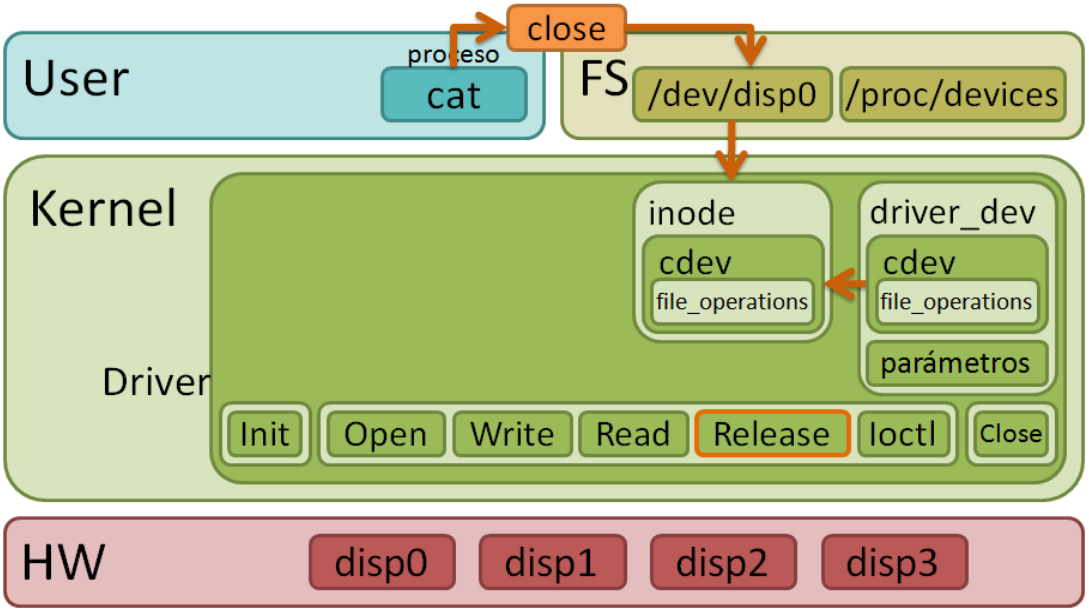


Figura 4.25: Proceso de lectura del driver parte 3 (tomada de [5]).

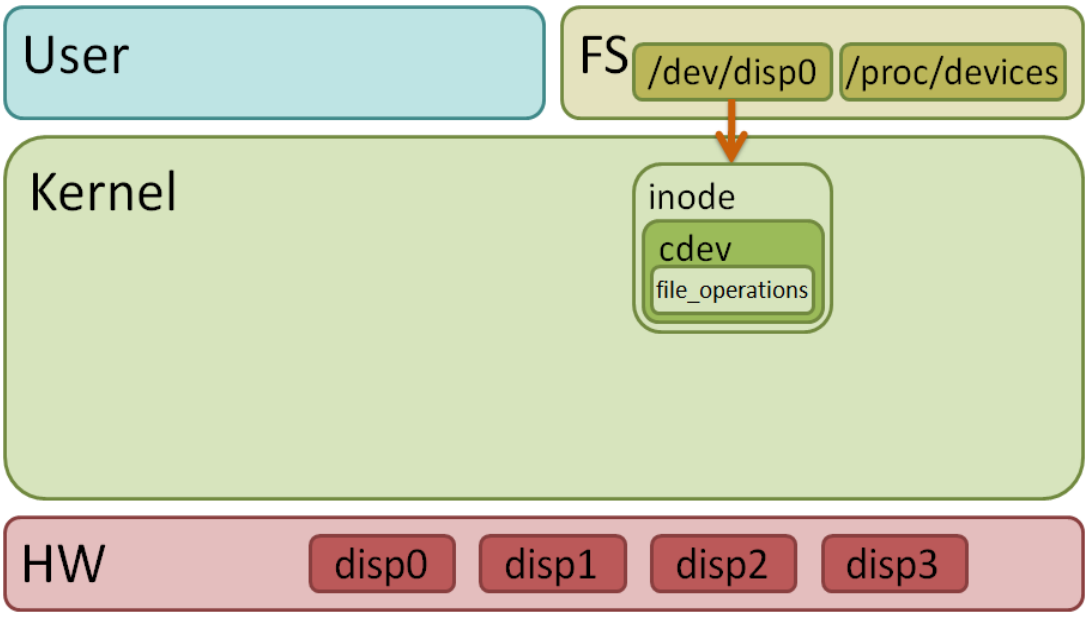


Figura 4.26: Liberación del driver del kernel (tomada de [5]).

y sobre un periférico descrito en Verilog encargado de administrar el módulo LCD del sistema de desarrollo. Se considera que los cambios que se le deben realizar a la plantilla para poder trabajar sobre los módulos de migración o cualquier otro módulo serán mínimos y estarán centrados en las secciones **read** y **write**. De allí su importancia para implementaciones futuras.

4.6 Adaptación de la plantilla para usar los periféricos de migración.

En el apéndice I en la sección I.2 se listan las modificaciones realizadas a la plantilla para usar desde el sistema operativo, el procesador específico. Las únicas secciones de la plantilla que no se modificaron fueron la sección en donde se definen las librerías de linux que se van a usar y la sección en donde se define la función **ioctl**. Es por ello que estas dos partes no se muestran en esta sección del apéndice.

4.7 Desarrollo de funciones y librerías

Luego de finalizar el desarrollo del Driver se elaboraron un conjunto de funciones y librerías en **C** para facilitar la interacción del usuario final con el Driver desde el sistema operativo. Estas funciones se hicieron pensando en un modelo a capas como se muestra en la figura 4.27. Por lo tanto se pueden clasificar en tres grupos de acuerdo a su cercanía con el driver:

- **Capa 0:** Son las funciones que interactuan directamente con el Driver y son las encargadas de hacer los llamados al sistema.

  PERIFERICO_seek_set: Permite seleccionar con cual registro del periférico (el periférico tiene 19 registros) se desea interactuar.

  PERIFERICO_write: Esta función realiza la escritura de un entero sobre el registro seleccionado.

  PERIFERICO_write_f: Esta función realiza la escritura de un flotante sobre el registro seleccionado.

  PERIFERICO_read: Extrae la información del registro seleccionado como un entero.

  PERIFERICO_read_f: Extrae la información del registro seleccionado como un flotante.

- **Capa 1:** Son las funciones intermedias encargadas de hacer procesos un poco más complejos utilizando las funciones de la capa cero.

  PERIFERICO_stop: Detiene el funcionamiento del procesador específico por medio de la activación de la bandera de reset en el registro de configuración (slv_reg_0).

  PERIFERICO_load: Es la encargada de enviarle al procesador específico el vector de datos que se debe procesar junto con los demás parámetros como **mu**, **m** y **jx**.

  PERIFERICO_start: Se encarga de generar el flanco ascendente en la bandera inicio del registro de configuración para que el procesador específico empiece a trabajar.

  PERIFERICO_extract: Se encarga de extraer los resultados obtenidos por el procesador específico cuando este finaliza. Estos resultados los almacena en un vector y se los entrega a **SU**

PERIFERICO_debug_ocupado: Esta función es la encarga de revisar si el procesador específico aún esta trabajando o si ya termino de procesar los datos. Esto lo hace usando una bandera del registro de configuración.

PERIFERICO_debug_estado: Esta función esta encargada de revisar en que estado se encuentra la máquina de estados del procesador específico por medio de unas banderas del registro de configuración.

- **Capa 2:** Las funciones de esta capa son las que interactuan con el usuario y con las funciones de la capa 1 para facilitar el manejo del procesador específico. Su objetivo es hacer ver al procesador específico como una función en **C**.

  PERIFERICO_principal: Es la encargada de hacer los llamados al sistema **Open y close**, pedirle al usuario la información necesaria para poder hacer uso del procesador específico y usar las funciones de la capa 1 en la secuencia correcta. Al final retorna en el mismo vector de entrada los resultados de los cálculos realizados.

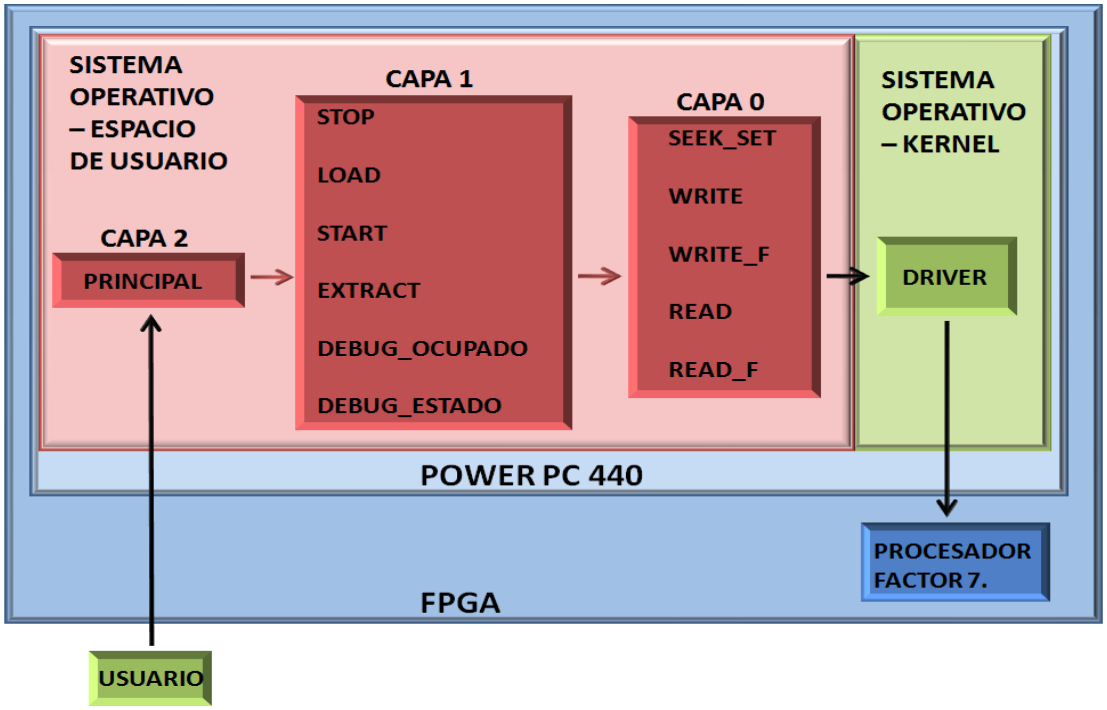


Figura 4.27: Diagrama de capas.

La versión final de estas funciones se muestra en el apéndice J desde la sección J.1 hasta J.14. Luego de finalizadas las funciones se hizo un programa de prueba para usar el procesador específico desde linux y verificar su correcto funcionamiento. El programa de prueba con su respectivo **Makefile** se pueden ver en el apéndice J en las secciones J.15 e J.16. Finalmente, luego de verificar que el Driver, las funciones en **C** y el procesador específico funcionaran correctamente se procedió a modificar el código fuente de la función **Pfafft.c** de **SU** para de esta forma poder usar el procesador específico cada vez que **SU** llame la función **Pfafft.c**. La versión modificada de **Pfafft.c**

se puede ver en el apéndice J en la sección J.17. Luego de que la función **Pfafft.c** se modificó fue necesario volver a compilar **SU**.

De esta forma finalizamos este capitulo para dar inicio a las pruebas de desempeño.

Capítulo 5

Medición del desempeño de los nodos Okinawa-2x.

5.1 Introducción

En este capitulo se presenta de forma detallada el desempeño del FPGA dentro del Cluster Okinawa cuando ejecuta el proceso de migración sísmica 2D. Además se compara dicho desempeño con el alcanzado por los otros nodos del Cluster para al final identificar los factores que influyeron en el desempeño del proceso.

La estrategia utilizada para medir el desempeño del cluster fue lanzar una serie de scripts de prueba sobre cada nodo para identificar el patrón de su comportamiento.

5.2 Elaboración de los scripts de prueba

Como la duración del proceso de migración esta sujeta a la cantidad de trazas que debe procesar el módulo *Sukdmig2d*, se elaboraron tres scripts de prueba, que haciendo uso del comando *cshot* (referirse a la sección 2.2) simularon tres adquisiciones para 80 (prueba corta), 800 (prueba media) y 1600 (prueba larga) geófonos. Luego partiendo de las trazas adquiridas por las simulaciones, se procedió a realizar el proceso de migración sobre cada grupo de datos.

Como se mencionó anteriormente, la tarjeta no tiene un disco duro y trabaja usando un sistema de archivos *Network File System (NFS)*. El nodo Okinawa-01 si tiene disco duro pero está conectado inalámbricamente al cluster como se puede ver en la Figura 4.1. Por lo tanto es necesario identificar los tiempos de transferencia de datos cuando los scripts de prueba estuvieran corriendo sobre estos nodos. Estos tiempos de transferencia influencian fuertemente los resultados obtenidos por la prueba. Por ello se decidió usar un sistema de archivos Ram (*RamFS*) para tratar de aislar este evento y poder identificar los tiempos de procesamiento reales.

El único inconveniente de hacer las pruebas sobre *RamFS*, es que los tiempos de transferencia de datos solo se pueden aislar siempre y cuando los datos a procesar sean inferiores a la capacidad de memoria RAM de cada nodo. Por lo tanto se hicieron las tres pruebas sobre cada nodo usando los dos sistemas de archivos (*NFS* y *RamFS*) para así poder identificar el porcentaje de tiempo de la prueba correspondiente a la transferencia de datos.

El script utilizado para hacer el montaje del sistema de archivos Ram se muestra en 5.1. Se puede apreciar en la línea 2 que la cantidad de memoria Ram reservada para estas tres pruebas fue 128 Megas.

```
1 mkdir -p carpeta-ram
2 mount -t tmpfs -o size=128m tmpfs carpeta-ram
3 df -k
4 cp script-tesis-prueba3/* carpeta-ram/
5 cd carpeta-ram
```

Script 5.1: montar-ram

5.3 Análisis de los resultados

Luego de realizadas las pruebas sobre el cluster se pudo apreciar que:

- El tiempo de transferencia de datos en la tarjeta es aproximadamente el 20% del tiempo de la aplicación (ver Figura 5.1, color verde).
- Para el caso del nodo Okinawa-01 que está conectado inalámbricamente los tiempos de transferencia de datos son más críticos porque están por encima del 60% del tiempo de la aplicación (ver Figura 5.2, color verde).
- Para el caso del Frontend los tiempos de transferencia de datos están por debajo del 2% del tiempo de la aplicación (ver Figura 5.3, color verde), concordando con la ventaja que tiene este nodo frente a los demás nodos del sistema, debido a que el NFS está montado sobre él.

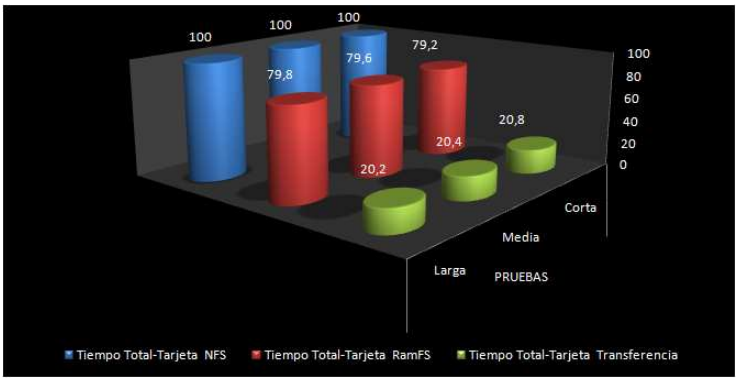


Figura 5.1: Comparación de los tiempos de ejecución de la tarjeta sobre sistemas de archivos NFS y RamFS.

Adicionalmente, de la información obtenida por las tres pruebas, se pudo medir el desempeño de tres de los nodos del sistema tanto para *NFS* como para *RamFS* y los resultados se muestran en las Tablas 5.1, 5.2, 5.3 y 5.4. De las Tablas 5.1 y 5.2 se pudo observar que el nodo que tuvo una mejoría notable en su desempeño fue Okinawa-01 debido a la gran reducción en los tiempos de transferencia de datos. Por otra parte Okinawa-00 fue el caso opuesto debido a la poca incidencia de los tiempos de transferencia de datos sobre el proceso de migración. Y en la tarjeta se refleja una mejoría del 20% bajo los cuatro ambientes de prueba.

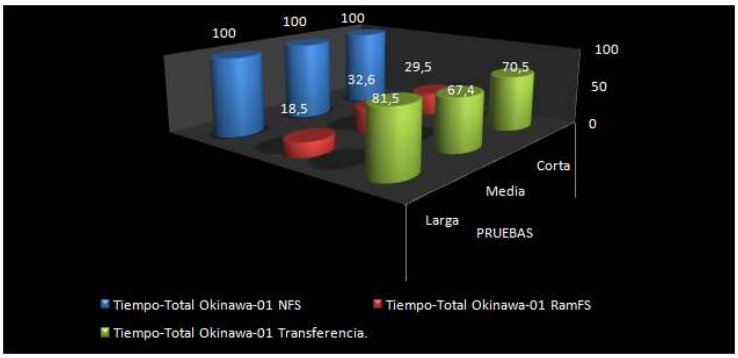


Figura 5.2: Comparación de los tiempos de ejecución del nodo Okinawa-01 sobre sistemas de archivos NFS y RamFS.

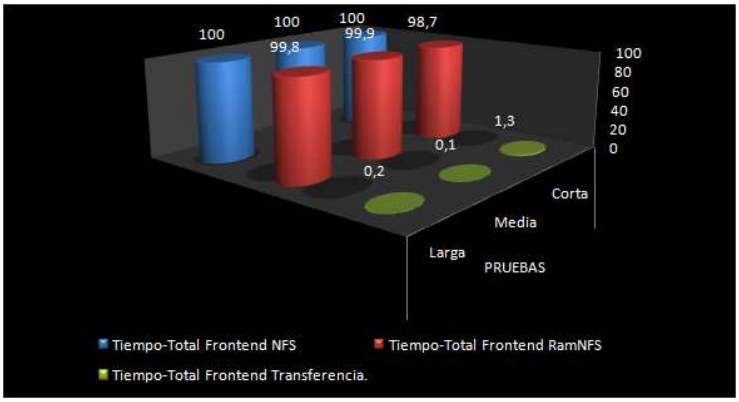


Figura 5.3: Comparación de los tiempos de ejecución del nodo frontend sobre sistemas de archivos NFS y RamFS.

Las Tablas 5.3 y 5.4 muestran una comparación de desempeños entre la tarjeta y los otros dos nodos para las tres pruebas sobre los dos sistemas de archivos. La medida de desempeño que se utilizó fue el **speed up** (tiempo Okinawa-00/tiempo FPGA), debido a que nos permite identificar si el proceso se acelero (>1), se desaceleró (<1) o si el desempeño fue igual (=1). De allí se evidencia que cuando el escenario de prueba es el ideal de la tarjeta (sum2=0 y mig2D=0) y el sistema de archivos es *NFS*, el desempeño del FPGA siempre es superior al de Okinawa-01; pero cuando el sistema de archivos es *RamFS* el desempeño del FPGA decae si se compara con el de Okinawa-01 y solo lo supera a partir de la prueba intermedia.

Por otra parte se pudo ver que bajo ningún escenario el FPGA logró superar el desempeño del Frontend (para estas tres pruebas), aunque fue mejorando a medida que la prueba aumentó el número de trazas a procesar. Si esta tendencia se mantiene a medida que el número de trazas se incrementa, es probable que a partir de cierto número de trazas el desempeño del FPGA supere el del frontend. Cabe aclarar que este entorno de prueba ideal es equivalente a la cota máxima de desempeño del FPGA, permitiéndonos identificar cual será el rango de trabajo de dicho nodo.

Tabla 5.1: Desempeño de los nodos del Cluster Okinawa al ejecutar los scripts de prueba usando NFS.

	Tiempos de ejecución del algoritmo de migración sísmica en segundos.					
Prueba	Okinawa-00	Okinawa-01	en el FPGA cuando			
			Solo el PowerPC	sum2=0	mig2D=0	sum2=0 y mig2D=0
Corta	15,01	124,25	498,81	418,07	156,16	75,43
Media	133,24	1041,27	4565,97	3759,65	1112,76	306,45
Larga	260,93	3507,72	9010,11	7414,96	2153,29	558,15

Tabla 5.2: Desempeño de los nodos del Cluster Okinawa al ejecutar los scripts de prueba usando RamFS.

	Tiempos de ejecución del algoritmo de migración sísmica en segundos.					
Prueba	Okinawa-00	Okinawa-01	en el FPGA cuando			
			Solo el PowerPC	sum2=0	mig2D=0	sum2=0 y mig2D=0
Corta	14,81	36,68	395,21	333,19	118,65	56,64
Media	133,11	339,39	3634,02	3011,77	902,24	279,99
Larga	260,28	648,23	7190,03	5966,29	1635,99	412,25

Tabla 5.3: Comparación de desempeños entre la tarjeta y los otros nodos cuando el sistema de archivos es NFS (Tiempo Okinawa-00 / Tiempo FPGA).

	prueba	Solo el PowerPC	sum2=0	mig2D=0	sum2=0 y mig2D=0
FPGA vs. Okinawa-00	Corta	0,03	0,04	0,1	0,2
	Media	0,03	0,04	0,12	0,43
	Larga	0,03	0,04	0,12	0,47
FPGA vs. Okinawa-01	Corta	0,25	0,3	0,8	1,65
	Media	0,23	0,28	0,94	3,4
	Larga	0,39	0,47	1,63	6,28

Tabla 5.4: Comparación de desempeños entre la tarjeta y los otros nodos cuando el sistema de archivos es RamFS (Tiempo Okinawa00 / Tiempo FPGA).

	prueba	Solo el PowerPC	sum2=0	mig2D=0	sum2=0 y mig2D=0
FPGA vs. Okinawa-00	Corta	0,037	0,044	0,125	0,261
	Media	0,037	0,044	0,148	0,475
	Larga	0,04	0,04	0,16	0,63
FPGA vs. Okinawa-01	Corta	0,093	0,11	0,309	0,648
	Media	0,093	0,113	0,376	1,212
	Larga	0,09	0,109	0,396	1,572

Un último aspecto que se pudo observar en los datos recolectados por estas pruebas, fue el costo computacional de las cinco funciones que componen el módulo de migración de SU. Mostrándonos que las dos que tienen un mayor costo computacional son *MIG2D* y *SUM2* como se puede apreciar en la Figura 5.4. Además si se revisa de una manera más detallada la distribución del costo computacional de la función *MIG2D*, encontramos que más del 60% es consumido por la subfunción *FILT* como se puede ver en la Figura 5.5.

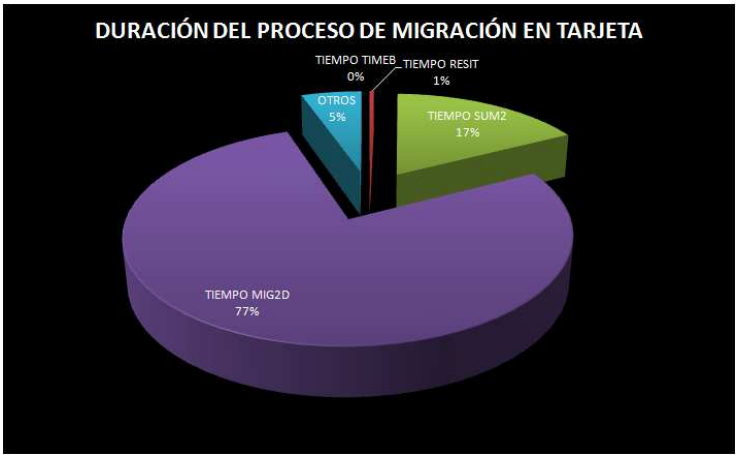


Figura 5.4: Distribución de la duración del proceso de migración dentro de la tarjeta.

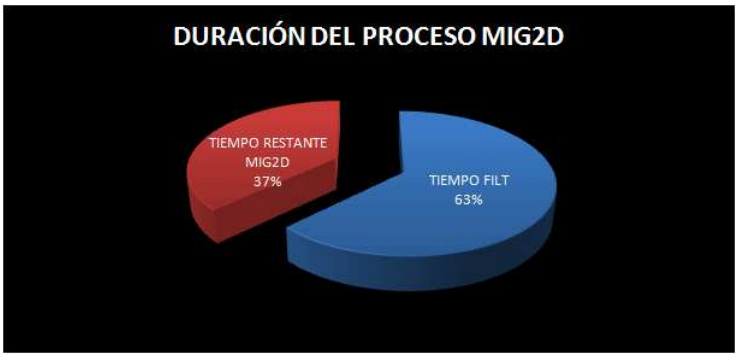


Figura 5.5: Distribución de la duración del proceso de mig2d dentro de la tarjeta.

Es por ello que esta función es una de las llamadas a ser mapeadas dentro de los recursos lógicos del FPGA junto con la función *SUM2*.

5.4 PowerPC 440 + procesador específico

Luego de que se caracterizó el desempeño de cada uno de los nodos del cluster Okinawa cuando ejecutan el proceso de migración sísmica, se procedió a medir el desempeño del nodo FPGA pero ahora trabajando en conjunto con el procesador específico que realiza todos los cálculos de la función **Pfafft.c** para el factor siete. Para poder hacer uso

del mismo se debió modificar el código fuente de la función **Pfafft.c** como se indicó en el capitulo anterior, luego recompilar **SU** y por último montar el driver ubicado en la ruta */compartido/comun/home/sergio/driverprocesador/* mediante el comando *./montar* como se muestra en el Script 5.2.

```
1 Okinawa-21:~\$ cd /compartido/comun/home/sergio/driverprocesador/
2 Okinawa-21:~/compartido/comun/home/sergio/driverprocesador\$ ./montar.sh
3 Okinawa-21:~/compartido/comun/home/sergio/driverprocesador\$ lsmod
4 Module Size Used by
5 primerdriver 6020 0
```

Script 5.2: Uso del driver modificado para el procesador del factor siete.

Una vez el driver fue montado se procedió a realizar las mismas tres pruebas anteriores (corta, media y larga) sobre el nodo FPGA y los resultados se muestran a continuación:

Tabla 5.5: Desempeño del nodo FPGA al ejecutar los scripts de prueba usando NFS.

	Tiempos de ejecución del algoritmo de migración sísmica en segundos.		
Prueba	**ppc alone**	**ppc with driver**	**speed up**
Corta	498,807779	503,19336	0,991285
Media	4565,965615	4595,10472	0,993658664
Larga	9010,11092	9055,45738	0,99499236

Tabla 5.6: Desempeño del nodo FPGA al ejecutar los scripts de prueba usando RamFS.

	Tiempos de ejecución del algoritmo de migración sísmica en segundos.		
Prueba	**ppc alone**	**ppc with driver**	**speed up**
Corta	395,20558	398,382353	0,99202582
Media	3634,022709	3681,43166	0,987122143
Larga	7190,02819	7238,74802	0,99326958

Adicionalmente a las tres pruebas anteriores se decidió realizar la prueba corta pero modificando el número de muestras por traza, para tratar de medir el desempeño del (PPC + procesador específico) cuando la transferencia de datos varía. Para modificar el número de muestras por traza en **SU** se utilizo el Script 5.3.

```
1 #! /bin/sh
2 suresamp < registro.su nt=7000 > registro3.su
3 suresamp < registro.su nt=10000 > registro2.su
4 suresamp < registro.su nt=20000 > registro4.su
5 suresamp < registro.su nt=500 > registro5.su
6 suresamp < registro.su nt=250 > registro6.su
7 suresamp < registro.su nt=125 > registro7.su
```

Script 5.3: Cambio del número de muestras por traza.

Y en la Tabla 5.7 se presentan los resultados de estas seis pruebas. En esta Tabla, **M** representa el número de veces que se tienen que hacer las operaciones del factor siete, **N** representa la longitud de la transformada y el #

de transferencias indica cuantos paquetes de datos se le deben enviar al procesador específico para que pueda hacer todos los cálculos.

Tabla 5.7: Desempeño del nodo (FPGA + Procesador específico) para diferente número de muestras por traza.

Prueba	Archivo de datos	# de muestras	M	N	# de transferencias	speed up
P1	Registro3.su	7000	528	3696	33	0,96076831
P2	Registro2.su	10000	720	5040	45	0,94992287
P3	Registro4.su	20000	1584	11088	99	0,91788572
P4	Registro5.su	500	36	252	2,25	0,99775947
P5	Registro6.su	250	18	126	1,125	1,11457099
P6	Registro7.su	125	12	84	0,75	1,12716117

Finalmente se procede a realizar una prueba de medición del error numérico obtenido al ejecutar la migración sísmica en el PowerPC alone y en el PowerPC junto con el driver. Estos resultados se comparan con el resultado obtenido al ejecutar la migración sísmica en el Frontend. Los resultados de la migración sísmica son obtenidos en SU y el error se define usando la medida del error cuadrático medio (ECM). El Script que se utilizo para extraer esta información se muestra en el Script 5.4.

```
1 #! /bin/sh
2 rm kd.bin
3 rm kd.bin.ascii
4 rm kd.headers.ascii
5 rm kd.data.ascii
6 sustrip < kd.data head=kd.headers  > kd.bin
7 b2a < kd.bin n1=1 > kd.bin.ascii
8 b2a < kd.headers n1=1 > kd.headers.ascii
9 b2a < kd.data n1=1 > kd.data.ascii
```

Script 5.4: Extracción de las datos de Seismic Unix.

Una vez obtenidos los resultados de las tres pruebas se calculó el **ECM** utilizando la ecuación 5.1, en donde

- Xi: Representa los valores de referencia.
- xi: Son los valores obtenidos.
- n: Son el número de muestras.

$$EMC = \frac{\sqrt{\sum_{i=0}^{n}(Xi - xi)^2}}{n} \tag{5.1}$$

Los resultados obtenidos se presentan en la Tabla 5.8.

Con la medición de los tiempos de procesamiento y del **ECM** se finalizó este trabajo de investigación. A continuación se presentan las conclusiones obtenidas a partir de las mediciones y pruebas realizadas en este capitulo. De igual forma se presentan algunas recomendaciones para futuras investigaciones relacionadas con este trabajo.

Tabla 5.8: Erro cuadrático medio (EMC) de los resultados obtenidos.

	Error cuadrático medio en las tres pruebas.		
Prueba	**Frontend Vs. PPC alone**	**Frontend Vs. PPC driver**	**PPC alone Vs. PPC driver**
Corta	48,5323369106	48,5323399298	0,0001544056
Media	212,7694680167	212,7694949827	0,0009480475
Larga	340,1750101907	340,1749970024	0,0018897441

5.5 Conclusiones y Trabajo futuro

A continuación se presentan las conclusiones obtenidas durante la investigación junto con las recomendaciones para trabajos futuros.

5.5.1 Conclusiones

- El ECM entre el PPC y el Frontend es considerable. Aunque ambos sistemas utilizan una unidad de punto flotante de precisión doble, se piensa que esta diferencia es debido a que el frontend tiene una arquitectura de 64 bits mientras el PPC trabaja con una arquitectura de 32 bits.
- El ECM entre el PPC y el procesador específico para el cálculo del factor siete esta en el orden de las milésimas y aparece debido a que el procesador específico trabaja en formato de punto flotante de precisión sencilla mientras que el PPC trabaja en punto flotante de precisión doble.
- El tiempo de transferencia de datos internos para la plataforma ml507 esta limitado por la máxima frecuencia de operación del bus del PPC (100 MHz). Por lo tanto, a medida que el número de transferencia de datos aumenta el tiempo de transferencia se hace superior al tiempo de procesamiento, mostrando una desaceleración en el desempeño del PPC.
- El desempeño que obtuvo el Power PC 440 cuando trabaja en conjunto con el procesador específico para el cálculo del factor siete (PPC + procesador específico) es inversamente proporcional al número de muestras por traza, debido a que a mayor número de muestras la cantidad de transferencias aumenta.
- Los procesos que son más favorables para ser acelerados sobre la plataforma ML507, son aquellos en donde se deben hacer muy pocas transferencias de datos y mucho procesamiento. El proceso de migración sísmica 2D en profundidad de *Kirchhoff* no cumple con la primer característica.
- La metodología empleada durante esta investigación se puede aplicar para cualquier problema de codiseño que se desee abordar con la plataforma de trabajo ML507.
- Una manera en la que los FPGAs podrían llegar a incursionar fuertemente en la computación de alto rendimiento sería mediante una plataforma de trabajo en la que se utilizará un procesador de propósito general externo, un FPGA de la familia LXT o HXT (gama alta) y una conexión de alta velocidad entre estos dos dispositivos (PCI express que sea superior a 4Gbps). Debido a que de esta manera se solucionarían los dos principales inconvenientes encontrados durante este trabajo (área y velocidad de transferencia de datos).

- Con la plataforma de codiseño desarrollada durante este proyecto de maestría, se pudo identificar una forma practica de lograr la interacción entre la teoría de procesadores, arquitectura, digitales y sistemas embebidos para tratar de solucionar un problema específico.

5.5.2 Trabajo futuro

- Con la alternativa de trabajo presentada en este proyecto de maestría y con las limitantes descubiertas a la hora de la transferencia de datos, se podría hacer un estudio sobre los demás procesos de interés para el ICP de SU, con el fin de identificar cuales de ellos se podrían acelerar.
- Partiendo de la información recopilada en esta investigación, se podría implementar una plataforma de codiseño utilizando un procesador de propósito general (externo con sistema operativo linux y con puerto PCI express) y el sistema de desarrollo ML507. De esta forma aprovechando la velocidad del puerto PCI express se podría eliminar la limitación en transferencia de datos. Adicionalmente como se cuenta con un procesador externo, el sistema de desarrollo ML507 ahora contaría con el 100% de sus recursos lógicos, lo que permitiría mapear más procesos dentro del FPGA e incluso usar unidades de punto flotante de precisión doble.

Bibliografía

[1] Oilfield glossary. `http://www.glossary.oilfield.slb.com/Display.cfm?Term=migration`.

[2] Scientific electronic library online. `http://www.scielo.org.ve/img/fbpe/rfiucv/v22n4/art06fig3.jpg`.

[3] U. Albertin, J. Kapoor, R. Randall, and M. Smith. La era de las imágenes en escala de profundidad. *Oilfield Review*, pages 1–16, julio 2002.

[4] HENGCHANG DAIBRITISH GEOLOGICAL SURVEY MURCHISON HOUSE. Optimizing the performance of parallel prestack kirchhoff time migration on a beowulf cluster. *EAGE 64th Conference and Exhibition - Florence, Italy*, pages 1–5, May 2002.

[5] Ramírez Ana Salamanca william. Diseño e implementación de un sistema embebido dinámicamente reconfigurable sobre un sistema de desarrollo ml507 configurado como un nodo de un cluster de procesadores de propósito general. Master's thesis, Universidad Industrial de Santander, September 2010.

[6] Abreo Sergio y Ramírez Ana. Viabilidad de acelerar la migración sísmica 2d usando un procesador específico implementado sobre un fpga. *Ingeniería e investigación*, 30:64–70, Abril 2010.

[7] Jon F. Claerbout. *Basic Earth Imaging.* Stanford, San Francisco, USA, 2008.

[8] C. He, M. Lu, and C. Sun. Accelerating seismic migration using fpga-based coprocessor platform. In *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pages 207–216, April 2004.

[9] Center for wave phenomena - colorado school of mines. `http://www.cwp.mines.edu/`.

[10] Clive Temperton. A new set of minimum-add small-n rotated dft modules. *J. Comput. Phys.*, 75:190–198, March 1988.

[11] Clive Temperton. Implementation of a self-sorting in-place prime factor fft algorithm. *Journal of Computational Physics*, 58(3):283 – 299, 1985.

[12] Fajardo Carlos. Ramón Jorge. Apropiación tecnológica del diseño de embedded system implementado sobre fpgas y cpld´s. Master's thesis, Universidad Industrial de Santander, December 2010.

[13] David Harris and Sarah Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[14] Modular reliable design. `http://sites.google.com/site/cursofpgasuis/archivos`.

[15] Clock skew and clock domains. `http://sites.google.com/site/cursofpgasuis/diapositivas`.

[16] Debian clusters for education and research: The missing manual. `http://www.debianclusters.org/index.php/Ganglia:_Installation`.

[17] gappmon: Monitoração de aplicações paralelas. `http://http://www.veiga.eti.br/gappmon/index-pt.html`.

[18] Lab 3 - powerpc processor. adding custom ip to an embedded system lab:. `http://www.it.uom.gr/teaching/embedded/material/tutorial%20adding%20my_ip.pdf`.

[19] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers, 3rd Edition*. O'Reilly Media, Inc., 2005.

Parte I

Apéndices

Apéndice **A**

Configuración de Cshot.

A.1 Configuración de Cshot1

La función *Cshot1* se configura a partir de un archivo llamado *param1* el cual se puede ver en el Script A.1. Para interpretar de una forma más clara lo que hace este Script, vamos a revisarlo renglón por renglón.

- Podemos apreciar que lo primero que realiza el script es llamar a un archivo llamado **Modelo**(ver script A.2) el cual contiene las coordenadas de las capas de la estructura del modelo geológico; para este caso se creo un modelo de tres capas.
- Segundo renglón: Define la cantidad de capas del modelo que se desea crear; nuevamente tres.
- Tercer renglón: llama al archivo **coloresmodelo**(ver scriptA.3) el cual define que los receptores serán de color negro, las fuentes de color azul oscuro, los rayos de color rojo, etc..
- Cuarto renglón: Se escoge lo que se quiere dibujar. Las opciones son el modelo, el pozo, ambos ó ninguno.
- Quinto renglón: Llama el archivo **pozofalso**(ver scriptA.4) el cual tiene las coordenadas del pozo.
- Sexto renglón: Define como se realizará la adquisición, si es superficial o con amarre a pozo.
- Séptimo renglón: Llama al archivo **geometria**(ver scriptA.5) el cual define las coordenadas de la posición y profundidad de las fuentes y los receptores.
- Octavo renglón: Define si se desean dibujar las fuentes, los receptores, ambos o ninguno.
- Noveno renglón: Determina que información de toda la que genera el módulo cshot1 se desea guardar. Las opciones son guardar los rayos, el tiempo y el archivo que pide cshot2.
- Décimo renglón: Permite definir el nombre con el que empiezan los archivos que se guardaran después de ejecutar el comando cshot1.
- Renglones once y doce: Configuran los parámetros de los ángulos de búsqueda del trazador de rayos.
- Renglón trece: Define la velocidad de cada capa (contando la superficial).
- Renglones catorce al diecinueve: Permiten escoger con cuales ondas se quiere trabajar (directas o reflexión, primarias o segundos arribos).

```
1  Modelo          ;Nombre del archivo que tiene el modelo
2  3               ;Número de interfaces del modelo
3  coloresmodelo   ;Archivo de los colores del modelo
4  m               ;Dibuja(m=modelo, w=pozo, q=nada)
5  pozofalso       ;coordenadas de pozo
6  s               ;Modo de adquisición s=superficie y d=pozo.
7  geometria       ;geometria de receptores
8  sg              ;Dibujar fuente, receptor ó quitar(sgq).
9  rlt             ;Resultados a guardar en archivo (r-rayos, l-tiempo, t-cshot2)
10 demo            ;Nombre del archivo de salida
11 -80. 80.        ;Rango de ángulos de busqueda
12 0.1             ;Incremento de ángulo de busqueda
13 2000.0 3000.0 4000.0 5000.0     ;velocidades
14 n       ;Onda directa?(y or n)
15         ;interfaces con refracciones
16 y       ;Primarias? (y or n)
17 1
18 2
19 3
```

Script A.1: Param1

```
1   -100.          0.              ;upper surface
2  10000.          0.              ...
3      1.      -99999.             ;end of upper surface
4   -100.        1000.             ;interface 1
5      0.        1050
6    500.        1100.             ...
7   1000.        1300.             ...
8   1800.         950.             ...
9   2500.        1100.
10  3400.         950.             ...
11  4000.        1000.             ...
12 10000.         800.             ...
13     1.      -99999.             ;end of interface 1
14  -100.        1900.             ;interface 2
15     0.        1850
16   500.        1800.             ...
17  1000.        1950.             ...
18  1800.        1900.             ...
19  2500.        1700.
20  3400.        1750.             ...
21  4000.        1810.             ...
22 10000.        1900.
23     1.      -99999.             ;end of interface 2
24  -100         2500.     ;interface 3
25  1000.        2500.
26  2500.        2000.
27  4000.        2000.
28 10000.        2000.     ;end of interface 3
29     1.      -99999.
```

Script A.2: Modelo

```
0          receivers
4          sources
6          well color
3          caustic rays
2          rays
4          interfaces


key:  (CWP's xgraph colors)
0      black
1      white
2      red
3      green
4      dark blue
5      light blue
6      violet
7      yellow
```

Script A.3: Colores Modelo

```
7.                          :x-coord. del tope del pozo
7.  2.                      :x,z
5.  7.                      :x,z
1.   -999999.               :end of well definition


Notes
-----

(1) When the upper surface is curved you may not know the elevation
at the x-coordinate of the top of the  well. The program calculates
this for you -
hence you only provide the x-coordinate where the well meets the upper
surface.

(2) The well should not be too curvy, and must be single valued in z (ie it
can't turn back towards the upper surface).
```

Script A.4: Pozo falso

El script de geometría A.5 quizás no es tan claro como los anteriores, es por ello que a continuación se explicará de forma detallada cada renglón :

- El primer renglón es el encargado de definir cual va a ser la estación de referencia y cual será su coordenada en el eje x. Para este caso particular la estación de referencia será la número 1 y estará ubicada en la coordenada x=50. Esta estación de referencia es importante para construir toda la geometría de la adquisición, debido a que todos los receptores y los disparos se ubicaran tomando como referencia a dicha estación.

- El segundo renglón es el encargado de definir la distancia entre los geófonos y la ubicación en profundidad. Para este caso particular los geófonos estarán ubicadas cada 10 metros y a una profundidad de 0 (en la superficie).
- A partir del tercer renglón se definen las geometrías para todos los disparos. Es decir se puede definir la ubicación de los receptores para cada uno de los disparos que se van a analizar. Volviendo al script A.5 vemos que el tercer renglón tiene 6 parámetros, los cuales tienen el siguiente significado en la adquisición física real:

  El primer parámetro define el nombre del primer geófono para esta adquisición.

  El segundo parámetro define el nombre del último geófono antes del gap.

  El tercer parámetro define el nombre del primer geófono después del gap.

  El cuarto parámetro define el nombre del último geófono de esta adquisición.

  El quinto parámetro define la coordenada en el eje x del disparo actual y el sexto define la profundidad de este disparo.

  El gap lo podemos definir como un lugar en el cual no hubo geófono.
- A partir del cuarto renglón se pueden definir las geometrías para las adquisiciones restantes (desde el disparo 2 hasta n), de la misma forma que en el tercer renglón, si es que la geometría de las adquisiciones va cambiando. Para nuestro caso particular, como deseamos que la geometría de la adquisición se mantenga para los 79 disparos restantes, entonces escribimos en el cuarto renglón 80, que representa los 80 disparos haga lo mismo desplazando todas las estaciones 50 metros por disparo hacia la derecha.

```
1 1          50.                          :reference station number and x-coord.
2 10.         0.                           :station spacing and receiver depth
3 1   2   2   2        100.   0.            :shot 1 - r1 r2 r3 r4 s sdepth
4 80 50
```

Script A.5: geometria

A.2 Configuración de Cshot2

La función *Cshot2* se configura a partir de un archivo llamado *param2* el cual se puede ver en el script A.6. Para poder interpretar de una forma más clara el contenido de este script, vamos a revisarlo renglón por renglón.

- Primer renglón: Especifica la opción para organizar los datos de la adquisición. Se pueden agrupar por disparos (s) ó por receptores (r).
- Segundo renglón: Define cuales grabaciones por disparo (cada disparo tiene la misma cantidad de trazas) se incluirán en el grupo con el que se va a trabajar.
- Tercer renglón: Este renglón depende del primero. Si en el primero se escogió **s** entonces en este renglón se define la primer y última traza con las que se van a trabajar dentro de todos los grupos seleccionados en dos. Por ejemplo si se desean trabajar con las trazas desde 30-50 grabadas por cada disparo simplemente se

coloca en este renglón 30 50. Ahora si en el primer renglón se escogió **r** entonces en este lugar se escogen el primer y último receptor sobre los que se quieren trabajar. El número de trazas adquiridas por cada receptor depende del número de disparos para los cuales los receptores estuvieron activos.

- Desde los renglones cuatro al nueve el script A.6 es lo suficientemente claro.

```
1 s                :job option(s,r)
2 1 81       :first, last shot for sort
3 1 1              :first, last trace OR first last receiver
4 10. 25. 35. 50. :frequency espectrum of wavelet
5 .050             :wavelet length (secs)
6 0.004              :sample rate (secs)
7 4.       :record length (secs)
8 demoshot         :input filename
9 trazas           :output filename
```

Script A.6: Param2

Apéndice B

Código fuente de la función SUKDMIG2D.

B.1 Función SUKDMIG2D

```
1
2  /* Copyright (c) Colorado School of Mines, 2008.*/
3  /* All rights reserved.                       */
4
5  /* SUKDMIG2D: $Revision: 1.24 $ ; $Date: 2006/11/07 22:58:42 $  */
6
7  #include "su.h"
8  #include "segy.h"
9
10 /************************ self documentation **********************/
11 char *sdoc[] = {
12 "                                                                    ",
13 "SUKDMIG2D - Kirchhoff Depth Migration of 2D poststack/prestack data    ",
14 "                                                                    ",
15 "    sukdmig2d  infile=  outfile=  ttfile=   [parameters]             ",
16 "                                                                    ",
17 " Required parameters:                                               ",
18 " infile=stdin          file for input seismic traces                ",
19 " outfile=stdout        file for common offset migration output      ",
20 " ttfile=               file for input traveltime tables             ",
21 "                                                                    ",
22 " ...  The following 9 parameters describe traveltime tables:        ",
23 " fzt=                  first depth sample in traveltime table       ",
24 " nzt=                  number of depth samples in traveltime table  ",
25 " dzt=                  depth interval in traveltime table           ",
26 " fxt=                  first lateral sample in traveltime table     ",
27 " nxt=                  number of lateral samples in traveltime table ",
28 " dxt=                  lateral interval in traveltime table         ",
29 " fs=                   x-coordinate of first source                 ",
30 " ns=                   number of sources                            ",
31 " ds=                   x-coordinate increment of sources            ",
32 "                                                                    ",
33 " Optional Parameters:                                               ",
34 " dt= or from header (dt)       time sampling interval of input data  ",
```

```
35  " ft= or from header (ft)       first time sample of input data         ",
36  " dxm= or from header (d2)      sampling interval of midpoints          ",
37  " fzo=fzt                   z-coordinate of first point in output trace ",
38  " dzo=0.2*dzt           vertical spacing of output trace                ",
39  " nzo=5*(nzt-1)+1       number of points in output trace                ",
40  " fxo=fxt                   x-coordinate of first output trace          ",
41  " dxo=0.5*dxt           horizontal spacing of output trace              ",
42  " nxo=2*(nxt-1)+1       number of output traces                         ",
43  " off0=0                        first offest in output                  ",
44  " doff=99999            offset increment in output                      ",
45  " noff=1                number of offsets in output                     ",
46  " absoff=0      flag for using absolute offsets of input traces         ",
47  "               =0 means use offset=gx-sx                               ",
48  "               =1 means use abs(gx-sx)                                 ",
49  " limoff=0      flag for only using input traces that fall within the range",
50  "               of defined output offset bins (off0,doff,noff)          ",
51  "               =0 means use all input traces                           ",
52  "               =1 means limit traces used by offset                    ",
53  " fmax=0.25/dt          frequency-highcut for input traces              ",
54  " offmax=99999          maximum absolute offset allowed in migration    ",
55  " aperx=nxt*dxt/2       migration lateral aperature                     ",
56  " angmax=60             migration angle aperature from vertical         ",
57  " v0=1500(m/s)          reference velocity value at surface            ",
58  " dvz=0.0               reference velocity vertical gradient            ",
59  "                                                                       ",
60  " ls=1                  flag for line source                            ",
61  " jpfile=stderr         job print file name                             ",
62  "                                                                       ",
63  " mtr=100               print verbal information at every mtr traces    ",
64  " ntr=100000            maximum number of input traces to be migrated   ",
65  " npv=0                 flag of computing quantities for velocity analysis",
66  " rscale=1000.0         scaling for roundoff error suppression          ",
67  "                                                                       ",
68  "   ...if npv>0 specify the following three files:                      ",
69  " tvfile=tvfile         input file of traveltime variation tables       ",
70  "                       tv[ns][nxt][nzt]                                ",
71  " csfile=csfile         input file of cosine tables cs[ns][nxt][nzt]    ",
72  " outfile1=outfile1     file containning additional migration output    ",
73  "                       with extra amplitude                            ",
74  "                                                                       ",
75  " Notes:                                                                ",
76  " 1. Traveltime tables were generated by program rayt2d (or other ones) ",
77  "    on relatively coarse grids, with dimension ns*nxt*nzt. In the      ",
78  "    migration process, traveltimes are interpolated into shot/gephone  ",
79  "    positions and output grids.                                        ",
80  " 2. Input seismic traces must be SU format and can be any type of      ",
81  "    gathers (common shot, common offset, common CDP, and so on).       ",
82  " 3. Migrated traces are output in CDP gathers if velocity analysis     ",
83  "    is required, with dimension nxo*noff*nzo.                          ",
84  " 4. If the offset value of an input trace is not in the offset array   ",
85  "    of output, the nearest one in the array is chosen.                 ",
86  " 5. Memory requirement for this program is about                       ",
87  "       [ns*nxt*nzt+noff*nxo*nzo+4*nr*nzt+5*nxt*nzt+npa*(2*ns*nxt*nzt   ",
```

```
"       +noff*nxo*nzo+4*nxt*nzt)]*4 bytes                                ",
"    where nr = 1+min(nxt*dxt,0.5*offmax+aperx)/dxo.                     ",
" 6. Amplitudes are computed using the reference velocity profile, v(z),",
"    specified by the parameters v0= and dvz=.                           ",
" 7. Input traces must specify source and receiver positions via the header",
"    fields tr.sx and tr.gx. Offset is computed automatically.           ",
" 8. if limoff=0, input traces from outside the range defined by off0, doff, ",
"    noff, will get migrated into the extremal offset bins/planes.  E.g. if ",
"    absoff=0 and limoff=0, all traces with gx<sx will get migrated into the ",
"    off0 bin.",
"                                                                        ",
NULL};
/*
 * Author:  Zhenyue Liu, 03/01/95,  Colorado School of Mines
 * Modifcations:
 *    Gary Billings, Talisman Energy, Sept 2005:  added absoff, limoff.
 *
 * Trace header fields accessed: ns, dt, delrt, d2
 * Trace header fields modified: sx, gx
 */

/**************** end self doc ********************************/
  void resit(int nx,float fx,float dx,int nz,int nr,float dr,
        float **tb,float **t,float x0);
  void interpx(int nxt,float fxt,float dxt,int nx,float fx,float dx,
        int nzt,float **tt,float **t);
  void sum2(int nx,int nz,float a1,float a2,float **t1,float **t2,float **t);
  void timeb(int nr,int nz,float dr,float dz,float fz,float a,
        float v0,float **t,float **p,float **sig,float **ang);
  void mig2d(float *trace,int nt,float ft,float dt,
        float sx,float gx,float **mig,float aperx,
        int nx,float fx,float dx,float nz,float fz,float dz,
        int ls,int mtmax,float dxm,float fmax,float angmax,
        float **tb,float **pb,float **cs0b,float **angb,int nr,
        float **tsum,int nzt,float fzt,float dzt,int nxt,float fxt,float dxt,
        int npv,float **cssum,float **tvsum,float **mig1);

/* define */
#define RSCALE_KDMIG 1000.0

/* segy trace */
segy tr, tro;

int
main (int argc, char **argv)
{
        int nt;         /* number of time samples in input data         */
        int nzt;        /* number of z-values in traveltime table       */
        int nxt;        /* number of x-values in traveltime table       */
        int nzo;        /* number of z-values in output data            */
        int nxo;        /* number of x-values in output data            */
        int ns;         /* number of sources                            */
        int noff;       /* number of offsets in output data             */
```

```
141         int nr;
142         int is,io,ixo,izo; /* counters */
143         int ls,ntr,jtr,ktr,mtr,npv,mtmax;
144         int   offset,absoff,limoff;
145         off_t nseek;
146         float   ft,fzt,fxt,fzo,fxo,fs,off0,dt,dzt,dxt,dzo,dxo,ds,doff,dxm,
147           ext,ezt,ezo,exo,es,s,scal;
148         float v0,dvz,fmax,angmax,offmax,rmax,aperx,sx=0,gx=0;
149         float ***mig=NULL,***ttab=NULL,**tb=NULL,**pb=NULL;
150         float **cs0b=NULL,**angb=NULL,**tsum=NULL,**tt=NULL;
151         float **tvsum=NULL,***mig1=NULL,***cs=NULL,***tv=NULL,**cssum=NULL;
152
153         float rscale;                   /* scaling factor for roundoff */
154
155         char *infile="stdin",*outfile="stdout",*ttfile,*jpfile,*tvfile,
156              *csfile,*outfile1;
157         FILE *infp,*outfp,*ttfp,*jpfp,*tvfp=NULL,*out1fp=NULL,*csfp=NULL;
158
159
160         /* hook up getpar to handle the parameters */
161         initargs(argc, argv);
162         requestdoc(1);
163
164         /* open input and output files  */ /* sección que revisa si se especificaron los tres
165         parámetros requeridos infile, ttfile y outfile */
166         if( !getparstring("infile",&infile)) {
167                 infp = stdin;
168         } else
169                 if ((infp=fopen(infile,"r"))==NULL)
170                         err("cannot open infile=%s\n",infile);
171         if( !getparstring("outfile",&outfile)) {
172                 outfp = stdout;
173         } else
174                 outfp = fopen(outfile,"w");
175         efseeko(infp,(off_t) 0,SEEK_CUR);
176         efseeko(outfp,(off_t) 0,SEEK_CUR);
177         if( !getparstring("ttfile",&ttfile))
178                 err("must specify ttfile!\n");
179         if ((ttfp=fopen(ttfile,"r"))==NULL)
180                 err("cannot open ttfile=%s\n",ttfile);
181         if( !getparstring("jpfile",&jpfile)) {
182                 jpfp = stderr;
183         } else
184                 jpfp = fopen(jpfile,"w");
185
186         /* get information from the first header */ /* revisa si se modificaron algunos de los
187          parámetros opcionales */
188         if (!fgettr(infp,&tr)) err("can't get first trace");
189         nt = tr.ns;
190         if (!getparfloat("dt",&dt)) dt = ((double) tr.dt)/1000000.0;
191         if (dt<0.0000001) err("dt must be positive!\n");
192         if (!getparfloat("ft",&ft)) ft = tr.delrt/1000.0;
193         if (!getparfloat("dxm",&dxm)) dxm = tr.d2;
```

```
194         if  (dxm<0.0000001) err("dxm must be positive!\n");
195
196         /* get traveltime tabel parameters */  /* Lee los parámetros de las tablas de los
197         tiempos de vuelo */
198         if (!getparint("nxt",&nxt)) err("must specify nxt!\n");
199         if (!getparfloat("fxt",&fxt)) err("must specify fxt!\n");
200         if (!getparfloat("dxt",&dxt)) err("must specify dxt!\n");
201         if (!getparint("nzt",&nzt)) err("must specify nzt!\n");
202         if (!getparfloat("fzt",&fzt)) err("must specify fzt!\n");
203         if (!getparfloat("dzt",&dzt)) err("must specify dzt!\n");
204         if (!getparint("ns",&ns)) err("must specify ns!\n");
205         if (!getparfloat("fs",&fs)) err("must specify fs!\n");
206         if (!getparfloat("ds",&ds)) err("must specify ds!\n");
207         if (!getparfloat("rscale",&rscale))   rscale = RSCALE_KDMIG;
208
209         ext = NINT(rscale*(fxt+(nxt-1)*dxt))/rscale;/* Distancia lateral de la tabla de
210         tiempos de vuelo */
211         ezt = NINT(rscale*(fzt+(nzt-1)*dzt))/rscale;/* Distancia vertical en la tabla de
212         tiempos de vuelo */
213         es = NINT(rscale*(fs+(ns-1)*ds))/rscale;/* Distancia cubierta por las fuentes durante
214          la adquisición */
215
216         /* optional parameters  */ /* Revisa si se modificaron algunos de los parámetros
217         opcionales */
218         if (!getparint("nxo",&nxo)) nxo = (nxt-1)*2+1;
219         if (!getparfloat("fxo",&fxo)) fxo = fxt;
220         if (!getparfloat("dxo",&dxo)) dxo = dxt*0.5;
221         if (!getparint("nzo",&nzo)) nzo = (nzt-1)*5+1;
222         if (!getparfloat("fzo",&fzo)) fzo = fzt;
223         if (!getparfloat("dzo",&dzo)) dzo = dzt*0.2;
224         exo = NINT(rscale*(fxo+(nxo-1)*dxo))/rscale;/* Distancia horizontal del sismograma de
225          salida */
226         ezo = NINT(rscale*(fzo+(nzo-1)*dzo))/rscale;/* Distancia vertical del sismograma de
227         salida */
228
229         fprintf(jpfp," fxt=%f fxo=%f \n",fxt,fxo);
230         fprintf(jpfp," ext=%f exo=%f \n",ext,exo);
231         fprintf(jpfp," fzt=%f fzo=%f \n",fzt,fzo);
232         fprintf(jpfp," ezt=%f ezo=%f \n",ezt,ezo);
233         fprintf(jpfp," es=%f \n",es);
234         fprintf(jpfp," \n");
235         if(fxt>fxo || ext<exo || fzt>fzo || ezt<ezo) {
236 warn("This condition must NOT be satisfied: fxt>fxo || ext<exo || fzt>fzo || ezt<ezo");
237                 warn("fxt=%f fxo=%f ext=%f exo=%f fzt=%f fzo=%f ezt=%f ezo=%f",
238                         fxt,fxo,ext,exo,fzt,fzo,ezt,ezo);
239                 err(" migration output range is out of traveltime table!\n");
240           }
241
242         if (!getparfloat("v0",&v0)) v0 = 1500; /* Revisa si fué asignado algun parámetro de
243         los opcionales de lo contrario asigna valores por defecto*/
244         if (!getparfloat("dvz",&dvz)) dvz = 0;
245         if (!getparfloat("angmax",&angmax)) angmax = 60.;
246         if  (angmax<0.00001) err("angmax must be positive!\n");
```

```
247         mtmax = 2*dxm*sin(angmax*PI/180.)/(v0*dt);
248         if(mtmax<1) mtmax = 1;
249         if (!getparfloat("aperx",&aperx)) aperx = 0.5*nxt*dxt;
250         if (!getparfloat("offmax",&offmax)) offmax = 3000;
251         if (!getparfloat("fmax",&fmax)) fmax = 0.25/dt;
252         if (!getparint("noff",&noff))    noff = 1;
253         if (!getparfloat("off0",&off0)) off0 = 0.;
254         if (!getparfloat("doff",&doff)) doff = 99999;
255
256         if (!getparint("ls",&ls))        ls = 1;
257         if (!getparint("absoff",&absoff))    absoff=0;
258         if (!getparint("limoff",&limoff))    limoff=0;
259         if (!getparint("ntr",&ntr))      ntr = 100000;
260         if (!getparint("mtr",&mtr))      mtr = 100;
261         if (!getparint("npv",&npv))      npv = 0;
262         if(npv){
263                 if( !getparstring("tvfile",&tvfile))
264                         err("must specify tvfile!\n");
265                 tvfp = fopen(tvfile,"r");
266                 if( !getparstring("csfile",&csfile))
267                         err("must specify csfile!\n");
268                 csfp = fopen(csfile,"r");
269                 if( !getparstring("outfile1",&outfile1))
270                         outfile1="outfile1";
271                 out1fp = fopen(outfile1,"w");
272         }
273
274         fprintf(jpfp,"\n");
275         fprintf(jpfp," Migration parameters\n");
276         fprintf(jpfp," ================\n");
277         fprintf(jpfp," infile=%s \n",infile);
278         fprintf(jpfp," outfile=%s \n",outfile);
279         fprintf(jpfp," ttfile=%s \n",ttfile);
280         fprintf(jpfp," \n");
281         fprintf(jpfp," nzt=%d fzt=%g dzt=%g\n",nzt,fzt,dzt);
282         fprintf(jpfp," nxt=%d fxt=%g dxt=%g\n",nxt,fxt,dxt);
283         fprintf(jpfp," ns=%d fs=%g ds=%g\n",ns,fs,ds);
284         fprintf(jpfp," \n");
285         fprintf(jpfp," nzo=%d fzo=%g dzo=%g\n",nzo,fzo,dzo);
286         fprintf(jpfp," nxo=%d fxo=%g dxo=%g\n",nxo,fxo,dxo);
287         fprintf(jpfp," \n");
288
289         /* compute reference traveltime and slowness  */
290         rmax = MAX(es-fxt,ext-fs);/*longitud máxima en x*/
291         rmax = MIN(rmax,0.5*offmax+aperx);/*mínima entre esta longitud y la mitad del offset
292         máximo más la apertura lateral de la migración*/
293         nr = 2+(int)(rmax/dxo);/*parte entera de esta distancia dividida entre el espaciado
294         horizontal de la traza de salida*/
295         tb = ealloc2float(nzt,nr);
296         pb = ealloc2float(nzt,nr);
297         cs0b = ealloc2float(nzt,nr);
298         angb = ealloc2float(nzt,nr);
299         timeb(nr,nzt,dxo,dzt,fzt,dvz,v0,tb,pb,cs0b,angb);/*Calcula 4 matrices que son slowness,
```

```
300          tiempo, cos(theta) y theta.*/
301
302         fprintf(jpfp," nt=%d ft=%g dt=%g \n",nt,ft,dt);
303         fprintf(jpfp," dxm=%g fmax=%g\n",dxm,fmax);
304         fprintf(jpfp," noff=%d off0=%g doff=%g\n",noff,off0,doff);
305         fprintf(jpfp," v0=%g dvz=%g \n",v0,dvz);
306         fprintf(jpfp," aperx=%g offmax=%g angmax=%g\n",aperx,offmax,angmax);
307         fprintf(jpfp," ntr=%d mtr=%d ls=%d npv=%d\n",ntr,mtr,ls,npv);
308         fprintf(jpfp," absoff=%d limoff=%d\n",absoff,limoff);
309         if(npv)
310           fprintf(jpfp," tvfile=%s csfile=%s outfile1=%s\n",
311                 tvfile,csfile,outfile1);
312         fprintf(jpfp," ================\n");
313         fflush(jpfp);
314
315         /* allocate space */
316         mig = ealloc3float(nzo,nxo,noff);
317         ttab = ealloc3float(nzt,nxt,ns);
318         tt = ealloc2float(nzt,nxt);
319         tsum = ealloc2float(nzt,nxt);
320         if(npv){
321                 tv = ealloc3float(nzt,nxt,ns);
322                 tvsum = ealloc2float(nzt,nxt);
323                 cs = ealloc3float(nzt,nxt,ns);
324                 cssum = ealloc2float(nzt,nxt);
325         }
326         if(!npv)
327                 mig1 = ealloc3float(1,1,noff);
328         else
329                 mig1 = ealloc3float(nzo,nxo,noff);
330
331
332         memset((void *) mig[0][0],0,noff*nxo*nzo*sizeof(float));
333          if(npv)
334                 memset((void *) mig1[0][0], 0,
335                         noff*nxo*nzo*sizeof(float));
336
337         fprintf(jpfp," input traveltime tables \n");
338
339         /* compute traveltime residual  */
340         for(is=0; is<ns; ++is){
341                 nseek = (off_t) nxt*nzt*is;
342                 efseeko(ttfp,nseek*((off_t) sizeof(float)),SEEK_SET);/*Revisa que el archivo ttfp
343                 (tablas de tiempos de vuelo) este correcto*/
344                 fread(ttab[is][0],sizeof(float),nxt*nzt,ttfp);/*Guarda en ttab un bloque de nxt*nzt
345                 de flotantes traido de ttfp que contiene las tablas de los tiempos de vuelo */
346                 s = fs+is*ds;/*Parte de la primera fuente en el eje x y avanza por todas las
347                 posiciones donde hubo una fuente*/
348                 resit(nxt,fxt,dxt,nzt,nr,dxo,tb,ttab[is],s);
349                 if(npv) {
350         efseeko(tvfp, nseek*((off_t) sizeof(float)),SEEK_SET);/*Revisa que el archivo tvfp
351         (tablas de variación de tiempos de vuelo) este correcto*/
352                         fread(tv[is][0],sizeof(float),nxt*nzt,tvfp);/*Guarda en tv un bloque de
```

```
353                         nxt*nzt de flotantes traido de tvfp*/
354                         efseeko(csfp, nseek*((off_t) sizeof(float)),SEEK_SET);/*Revisa que el
355                         archivo csfp (tablas de coseno) este correcto*/
356                         fread(cs[is][0],sizeof(float),nxt*nzt,csfp);/*Guarda en cs un bloque de
357                         nxt*nzt de flotantes traido de csfp*/
358                 }
359
360         }
361
362         fprintf(jpfp," start migration ... \n");
363         fprintf(jpfp," \n");
364         fflush(jpfp);
365
366
367         jtr = 1;
368         ktr = 0;
369
370           fprintf(jpfp," fs=%g es=%g offmax=%g\n",fs,es,offmax);
371
372         do {
373                 /* determine offset index       */
374             float as,res;
375
376                 if (tr.scalco) { /* if tr.scalco is set, apply value */
377                         if (tr.scalco>0) {
378                                 sx = tr.sx*tr.scalco;
379                                 gx = tr.gx*tr.scalco;
380                         } else { /* if tr.scalco is negative divide */
381                                 sx = tr.sx/ABS(tr.scalco);
382                                 gx = tr.gx/ABS(tr.scalco);
383                         }
384
385                 } else {
386                                 sx = tr.sx;
387                                 gx = tr.gx;
388                 }
389
390                 /* GWB 2005.09.22: */
391                 /* io = (int)((gx-sx-off0)/doff+0.5); */
392                 offset=gx-sx;
393                 if( absoff && offset<0 )offset=-offset;
394                 io = (int)((offset-off0)/doff+0.5);
395                 if( limoff && (io<0 || io>=noff) ) continue;
396                 /* end of GWB 2005.09.22 */
397
398             if(io<0) io = 0;
399             if(io>=noff) io = noff-1;
400
401             if(MIN(sx,gx)>=fs && MAX(sx,gx)<=es &&
402                  MAX(gx-sx,sx-gx)<=offmax ){
403                 /*      migrate this trace       */
404
405                     /* fprintf(jpfp," Good! Condition NOT satisfied\n"); */
```

```
406
407                 as = (sx-fs)/ds;
408                 is = (int)as;
409                 if(is==ns-1) is=ns-2;
410                 res = as-is;
411                 if(res<=0.01) res = 0.0;
412                 if(res>=0.99) res = 1.0;
413                 sum2(nxt,nzt,1-res,res,ttab[is],ttab[is+1],tsum);
414                 if(npv)  {
415                         sum2(nxt,nzt,1-res,res,tv[is],tv[is+1],tvsum);
416                         sum2(nxt,nzt,1-res,res,cs[is],cs[is+1],cssum);
417                 }
418
419                 as = (gx-fs)/ds;
420                 is = (int)as;
421                 if(is==ns-1) is=ns-2;
422                 res = as-is;
423                 if(res<=0.01) res = 0.0;
424                 if(res>=0.99) res = 1.0;
425                 sum2(nxt,nzt,1-res,res,ttab[is],ttab[is+1],tt);
426                 sum2(nxt,nzt,1,1,tt,tsum,tsum);
427                 if(npv)  {
428                         sum2(nxt,nzt,1-res,res,tv[is],tv[is+1],tt);
429                         sum2(nxt,nzt,1,1,tt,tvsum,tvsum);
430                         sum2(nxt,nzt,1-res,res,cs[is],cs[is+1],tt);
431                         sum2(nxt,nzt,1,1,tt,cssum,cssum);
432                 }
433
434                 mig2d(tr.data,nt,ft,dt,sx,gx,mig[io],aperx,
435                   nxo,fxo,dxo,nzo,fzo,dzo,
436                   ls,mtmax,dxm,fmax,angmax,
437                   tb,pb,cs0b,angb,nr,tsum,nzt,fzt,dzt,nxt,fxt,dxt,
438                   npv,cssum,tvsum,mig1[io]);
439
440                   ktr++;
441                   if((jtr-1)%mtr ==0 ){
442                         fprintf(jpfp," migrated trace %d\n",jtr);
443                         fflush(jpfp);
444                 }
445             }
446             jtr++;
447         } while (fgettr(infp,&tr) && jtr<ntr);
448
449         fprintf(jpfp," migrated %d traces in total\n",ktr);
450
451         memset((void *) &tro, 0, sizeof(segy));
452         tro.ns = nzo;
453         tro.d1 = dzo;
454         tro.dt = 1000*(int)(1000*dt+0.5);
455         tro.delrt = 0.0;
456         tro.f1 = fzo;
457         tro.f2 = fxo;
458         tro.d2 = dxo;
```

```
459         tro.trid = 200;
460
461         scal = 4/sqrt(PI)*dxm/v0;
462         for(ixo=0; ixo<nxo; ixo++) {
463                 for(io=0; io<noff; io++)  {
464                         memcpy((void *) tro.data,
465                                 (const void *) mig[io][ixo], nzo*sizeof(float));
466                         tro.offset = off0+io*doff;
467                         tro.tracr = 1+ixo;
468                         tro.tracl = 1+io+noff*ixo;
469                         tro.cdp = fxo+ixo*dxo;
470                         tro.cdpt = 1+io;
471
472                         for(izo=0; izo<nzo; ++izo)
473                                 tro.data[izo] *=scal;
474
475                         /* write out */
476                         fputtr(outfp,&tro);
477
478                    if(npv){
479                         memcpy((void *) tro.data,
480                                 (const void *) mig1[io][ixo],nzo*sizeof(float));
481                         for(izo=0; izo<nzo; ++izo)
482                                 tro.data[izo] *=scal;
483                         /* write out */
484                         fputtr(out1fp,&tro);
485                     }
486                 }
487         }
488
489         fprintf(jpfp," \n");
490         fprintf(jpfp," output done\n");
491         fflush(jpfp);
492
493         efclose(jpfp);
494         efclose(outfp);
495
496
497         free2float(tsum);/* En esta sección se liberan las matrices 2d y 3d que habían sido
498         apartadas para calcular valores previos */
499         free2float(tt);
500         free2float(pb);
501         free2float(tb);
502         free2float(cs0b);
503         free2float(angb);
504         free3float(ttab);
505         free3float(mig);
506         free3float(mig1);
507         if(npv){
508                 free3float(tv);
509                 free3float(cs);
510                 free2float(tvsum);
511                 free2float(cssum);
```

```
512         }
513         return(CWP_Exit());
514 }
515
516 /* residual traveltime calculation based on reference time */
517   void resi(int nx,float fx,float dx,int nz,int nr,float dr,
518                 float **tb,float **t,float x0)
519 {
520         int ix,iz,jr;
521         float xi,ar,sr,sr0;
522
523         for(ix=0; ix<nx; ++ix){
524                 xi = fx+ix*dx-x0;
525                 ar = abs(xi)/dr;
526                 jr = (int)ar;
527                 sr = ar-jr;
528                 sr0 = 1.0-sr;
529                 if(jr>nr-2) jr = nr-2;
530                 for(iz=0; iz<nz; ++iz)
531                         t[ix][iz] -= sr0*tb[jr][iz]+sr*tb[jr+1][iz];
532         }
533 }
534
535 /* lateral interpolation        */
536
537 /* sum of two tables    */
538   void sum2(int nx,int nz,float a1,float a2,float **t1,float **t2,float **t)
539 {
540         int ix,iz;
541
542         for(ix=0; ix<nx; ++ix)
543                 for(iz=0; iz<nz; ++iz)
544                         t[ix][iz] = a1*t1[ix][iz]+a2*t2[ix][iz];
545 }
546
547 /* compute  reference traveltime and slowness   */
548         void timeb(int nr,int nz,float dr,float dz,float fz,float a,
549         float v0,float **t,float **p,float **cs0,float **ang)
550 {
551         int  ir,iz;
552         float r,z,v,rc,oa,temp,rou,zc;
553
554
555         if( a==0.0) {
556                 for(ir=0,r=0;ir<nr;++ir,r+=dr)
557                         for(iz=0,z=fz;iz<nz;++iz,z+=dz){
558                                 rou = sqrt(r*r+z*z);
559                                 if(rou<dz) rou = dz;
560                                 t[ir][iz] = rou/v0;
561                                 p[ir][iz] = r/(rou*v0);
562                                 cs0[ir][iz] = z/rou;
563                                 ang[ir][iz] = asin(r/rou);
564                         }
```

```
565         } else {
566                 oa = 1.0/a;     zc = v0*oa;
567                 for(ir=0,r=0;ir<nr;++ir,r+=dr)
568                         for(iz=0,z=fz+zc;iz<nz;++iz,z+=dz){
569                                 rou = sqrt(r*r+z*z);
570                                 v = v0+a*(z-zc);
571                                 if(ir==0){
572                                         t[ir][iz] = log(v/v0)*oa;
573                                         p[ir][iz] = 0.0;
574                                         ang[ir][iz] = 0.0;
575                                         cs0[ir][iz] = 1.0;
576                                 } else {
577                                         rc = (r*r+z*z-zc*zc)/(2.0*r);
578                                         rou = sqrt(zc*zc+rc*rc);
579                                         t[ir][iz] = log((v*(rou+rc))
580                                                 /(v0*(rou+rc-r)))*oa;
581                                         p[ir][iz] = sqrt(rou*rou-rc*rc)
582                                                 /(rou*v0);
583                                         temp = v*p[ir][iz];
584                                         if(temp>1.0) temp = 1.0;
585                                         ang[ir][iz] = asin(temp);
586                                         cs0[ir][iz] = rc/rou;
587                                 }
588                         }
589         }
590 }
591
592 void filt(float *trace,int nt,float dt,float fmax,int ls,int m,float *trf);
593
594   void mig2d(float *trace,int nt,float ft,float dt,
595         float sx,float gx,float **mig,float aperx,
596         int nx,float fx,float dx,float nz,float fz,float dz,
597         int ls,int mtmax,float dxm,float fmax,float angmax,
598         float **tb,float **pb,float **cs0b,float **angb,int nr,
599         float **tsum,int nzt,float fzt,float dzt,int nxt,float fxt,float dxt,
600         int npv,float **cssum,float **tvsum,float **mig1)
601 /*****************************************************************************
602 Migrate one trace
603 *****************************************************************************
604 Input:
605 *trace           one seismic trace
606 nt              number of time samples in seismic trace
607 ft              first time sample of seismic trace
608 dt              time sampleing interval in seismic trace
609 sx,gx           lateral coordinates of source and geophone
610 aperx           lateral aperature in migration
611 nx,fx,dx,nz,fz,dz       dimension parameters of migration region
612 ls              =1 for line source; =0 for point source
613 mtmax           number of time samples in triangle filter
614 dxm             midpoint sampling interval
615 fmax            frequency-highcut for input trace
616 angmax          migration angle aperature from vertical
617 tb,pb,cs0b,angb          reference traveltime, lateral slowness, cosine of
```

```
618                 incident angle, and emergent angle
619 nr              number of lateral samples in reference quantities
620 tsum            sum of residual traveltimes from shot and receiver
621 nxt,fxt,dxt,nzt,fzt,dzt         dimension parameters of traveltime table
622 npv=0           flag of computing quantities for velocity analysis
623 cssume          sum of cosine of emergence angles from shot and recerver
624 tvsum           sum of  traveltime variations from shot and recerver
625
626 Output:
627 mig             migrated section
628 mig1            additional migrated section for velocity analysis if npv>0
629 ***************************************************************************/
630 {
631         int nxf,nxe,nxtf,nxte,ix,iz,iz0,izt0,nzp,jrs,jrg,jz,jt,mt,jx;
632         float xm,x,dis,rxz,ar,srs,srg,srs0,srg0,sigp,z0,rdz,ampd,res0,
633                 angs,angg,cs0s,cs0g,ax,ax0,pmin,
634                 odt=1.0/dt,pd,az,sz,sz0,at,td,res,temp;
635         float **tmt,**ampt,**ampti,**ampt1=NULL,*tm,*amp,*ampi,*amp1=NULL,
636                 *tzt,*trf,*zpt;
637
638         tmt = alloc2float(nzt,nxt);
639         ampt = alloc2float(nzt,nxt);
640         ampti = alloc2float(nzt,nxt);
641         tm = alloc1float(nzt);
642         tzt = alloc1float(nzt);
643         amp = alloc1float(nzt);
644         ampi = alloc1float(nzt);
645         zpt = alloc1float(nxt);
646         trf = alloc1float(nt+2*mtmax);
647         if(npv) {
648                 ampt1 = alloc2float(nzt,nxt);
649                 amp1 = alloc1float(nzt);
650         }
651
652         z0 = (fz-fzt)/dzt;
653         rdz = dz/dzt;
654         pmin = 1.0/(2.0*dxm*fmax);
655
656         filt(trace,nt,dt,fmax,ls,mtmax,trf);
657
658         xm = 0.5*(sx+gx);
659         rxz = (angmax==90)?0.0:1.0/tan(angmax*PI/180.);
660         nxtf = (xm-aperx-fxt)/dxt;
661         if(nxtf<0) nxtf = 0;
662         nxte = (xm+aperx-fxt)/dxt+1;
663         if(nxte>=nxt) nxte = nxt-1;
664
665         /* compute amplitudes and filter length */
666         for(ix=nxtf; ix<=nxte; ++ix){
667                 x = fxt+ix*dxt;
668                 dis = (xm>=x)?xm-x:x-xm;
669                 izt0 = ((dis-dxt)*rxz-fzt)/dzt-1;
670                 if(izt0<0) izt0 = 0;
```

```
671                 if(izt0>=nzt) izt0 = nzt-1;
672
673                 ar = (sx>=x)?(sx-x)/dx:(x-sx)/dx;
674                 jrs = (int)ar;
675                 if(jrs>nr-2) jrs = nr-2;
676                 srs = ar-jrs;
677                 srs0 = 1.0-srs;
678                 ar = (gx>=x)?(gx-x)/dx:(x-gx)/dx;
679                 jrg = (int)ar;
680                 if(jrg>nr-2) jrg = nr-2;
681                 srg = ar-jrg;
682                 srg0 = 1.0-srg;
683                 sigp = ((sx-x)*(gx-x)>0)?1.0:-1.0;
684                 zpt[ix] = fzt+(nzt-1)*dzt;
685
686                 for(iz=izt0; iz<nzt; ++iz){
687                         angs = srs0*angb[jrs][iz]+srs*angb[jrs+1][iz];
688                         angg = srg0*angb[jrg][iz]+srg*angb[jrg+1][iz];
689                         cs0s = srs0*cs0b[jrs][iz]+srs*cs0b[jrs+1][iz];
690                         cs0g = srg0*cs0b[jrg][iz]+srg*cs0b[jrg+1][iz];
691                         ampd = (cs0s+cs0g)*cos(0.5*(angs-angg));
692                         if(ampd<0.0) ampd = -ampd;
693                         ampt[ix][iz] = ampd;
694
695                         pd = srs0*pb[jrs][iz]+srs*pb[jrs+1][iz]+sigp
696                               *(srg0*pb[jrg][iz]+srg*pb[jrg+1][iz]);
697                         if(pd<0.0) pd = -pd;
698                         temp = pd*dxm*odt;
699                         if(temp<1) temp = 1.0;
700                         if(temp>mtmax) temp = mtmax;
701                         ampti[ix][iz] = ampd/(temp*temp);
702                         tmt[ix][iz] = temp;
703                         if(pd<pmin && zpt[ix]>fzt+(nzt-1.1)*dzt)
704                                 zpt[ix] = fzt+iz*dzt;
705
706                     if(npv){
707                         if(cssum[ix][iz]<1.0)
708                               ampt1[ix][iz] = 0;
709                         else
710                               ampt1[ix][iz] = tvsum[ix][iz]/cssum[ix][iz];
711                     }
712                 }
713         }
714
715         nxf = (xm-aperx-fx)/dx+0.5;
716         if(nxf<0) nxf = 0;
717         nxe = (xm+aperx-fx)/dx+0.5;
718         if(nxe>=nx) nxe = nx-1;
719
720         /* interpolate amplitudes and filter length along lateral       */
721         for(ix=nxf; ix<=nxe; ++ix){
722                 x = fx+ix*dx;
723                 dis = (xm>=x)?xm-x:x-xm;
```

```
724                 izt0 = (dis*rxz-fzt)/dzt;
725                 if(izt0<0) izt0 = 0;
726                 if(izt0>=nzt) izt0 = nzt-1;
727                 iz0 = (dis*rxz-fz)/dz;
728                 if(iz0<0) iz0 = 0;
729                 if(iz0>=nz) iz0 = nz-1;
730
731                 ax = (x-fxt)/dxt;
732                 jx = (int)ax;
733                 ax = ax-jx;
734                 if(ax<=0.01) ax = 0.;
735                 if(ax>=0.99) ax = 1.0;
736                 ax0 = 1.0-ax;
737                 if(jx>nxte-1) jx = nxte-1;
738                 if(jx<nxtf) jx = nxtf;
739
740                 ar = (sx>=x)?(sx-x)/dx:(x-sx)/dx;
741                 jrs = (int)ar;
742                 if(jrs>nr-2) jrs = nr-2;
743                 srs = ar-jrs;
744                 srs0 = 1.0-srs;
745                 ar = (gx>=x)?(gx-x)/dx:(x-gx)/dx;
746                 jrg = (int)ar;
747                 if(jrg>nr-2) jrg = nr-2;
748                 srg = ar-jrg;
749                 srg0 = 1.0-srg;
750
751                 for(iz=izt0; iz<nzt; ++iz){
752                     tzt[iz] = ax0*tsum[jx][iz]+ax*tsum[jx+1][iz]
753                                 +srs0*tb[jrs][iz]+srs*tb[jrs+1][iz]
754                                 +srg0*tb[jrg][iz]+srg*tb[jrg+1][iz];
755
756                     amp[iz] = ax0*ampt[jx][iz]+ax*ampt[jx+1][iz];
757                     ampi[iz] = ax0*ampti[jx][iz]+ax*ampti[jx+1][iz];
758                     tm[iz] = ax0*tmt[jx][iz]+ax*tmt[jx+1][iz];
759
760                     if(npv)
761                         amp1[iz] = ax0*ampt1[jx][iz]+ax*ampt1[jx+1][iz];
762
763                 }
764
765                 nzp = (ax0*zpt[jx]+ax*zpt[jx+1]-fz)/dz+1.5;
766                 if(nzp<iz0) nzp = iz0;
767                 if(nzp>nz) nzp = nz;
768
769                 /* interpolate along depth if operater aliasing         */
770                 for(iz=iz0; iz<nzp; ++iz) {
771                         az = z0+iz*rdz;
772                         jz = (int)az;
773                         if(jz>=nzt-1) jz = nzt-2;
774                         sz = az-jz;
775                         sz0 = 1.0-sz;
776                         td = sz0*tzt[jz]+sz*tzt[jz+1];
```

```
777                         at = (td-ft)*odt+mtmax;
778                         jt = (int)at;
779                         if(jt > mtmax && jt < nt+mtmax-1){
780                             ampd = sz0*ampi[jz]+sz*ampi[jz+1];
781                             mt = (int)(0.5+sz0*tm[jz]+sz*tm[jz+1]);
782                             res = at-jt;
783                             res0 = 1.0-res;
784                             temp = (res0*(-trf[jt-mt]+2.0*trf[jt]-trf[jt+mt])
785                                 +res*(-trf[jt-mt+1]+2.0*trf[jt+1]
786                                 -trf[jt+mt+1]))*ampd;
787                             mig[ix][iz] += temp;
788 
789                             if(npv)
790                                 mig1[ix][iz]  += temp
791                                         *(sz0*amp1[jz]+sz*amp1[jz+1]);
792                         }
793                 }
794 
795                 /* interpolate along depth if not operater aliasing     */
796                 for(iz=nzp; iz<nz; ++iz) {
797                         az = z0+iz*rdz;
798                         jz = (int)az;
799                         if(jz>=nzt-1) jz = nzt-2;
800                         sz = az-jz;
801                         sz0 = 1.0-sz;
802                         td = sz0*tzt[jz]+sz*tzt[jz+1];
803                         at = (td-ft)*odt;
804                         jt = (int)at;
805                         if(jt > 0 && jt < nt-1){
806                             ampd = sz0*amp[jz]+sz*amp[jz+1];
807                             res = at-jt;
808                             res0 = 1.0-res;
809                             temp = (res0*trace[jt]+res*trace[jt+1])*ampd;
810                             mig[ix][iz] += temp;
811                             if(npv)
812                                 mig1[ix][iz]  += temp
813                                         *(sz0*amp1[jz]+sz*amp1[jz+1]);
814                         }
815                 }
816 
817         }
818 
819         free2float(ampt);
820         free2float(ampti);
821         free2float(tmt);
822         free1float(amp);
823         free1float(ampi);
824         free1float(zpt);
825         free1float(tm);
826         free1float(tzt);
827         free1float(trf);
828         if(npv) {
829                 free1float(amp1);
```

```
830                     free2float(ampt1);
831             }
832     }
833
834     void filt(float *trace,int nt,float dt,float fmax,int ls,int m,float *trf)
835     /* Low-pass filter, integration and phase shift for input data
836        input:
837         trace(nt)    single seismic trace
838        fmax high cut frequency
839         ls           ls=1, line source; ls=0, point source
840       output:
841         trace(nt)    filtered and phase-shifted seismic trace
842         tracei(nt)   filtered, integrated and phase-shifted seismic trace
843      */
844     {
845             static int nfft=0, itaper, nw, nwf;
846             static float *taper, *amp, *ampi, dw;
847             int  it, iw, itemp;
848             float temp, ftaper, const2, *rt;
849             complex *ct;
850
851             fmax *= 2.0*PI; /* fmax=fmax*2*pi => Wmax */
852             ftaper = 0.1*fmax;/* frecuencia de corte =10% de la frecuencia máxima */
853             const2 = sqrt(2.0);
854
855             if(nfft==0) {
856                     /* Set up FFT parameters */
857                     nfft = npfaro(nt+m, 2 * (nt+m));
858                     if (nfft >= SU_NFLTS || nfft >= 720720)
859                             err("Padded nt=%d -- too big", nfft);
860
861                     nw = nfft/2 + 1;
862                     dw = 2.0*PI/(nfft*dt);/*(2*pi/N)*(fs)*/
863
864                     itaper = 0.5+ftaper/dw;
865                     taper = ealloc1float(2*itaper+1);
866                     for(iw=-itaper; iw<=itaper; ++iw){
867                             temp = (float)iw/(1.0+itaper);
868                             taper[iw+itaper] = (1-temp)*(1-temp)*(temp+2)/4;
869                     }
870
871                     nwf = 0.5+fmax/dw;
872                     if(nwf>nw-itaper-1) nwf = nw-itaper-1;
873                     amp = ealloc1float(nwf+itaper+1);
874                     ampi = ealloc1float(nwf+itaper+1);
875                     amp[0] = ampi[0] = 0.;
876                     for(iw=1; iw<=nwf+itaper; ++iw){
877                             amp[iw] = sqrt(dw*iw)/nfft;
878                             ampi[iw] = 0.5/(1-cos(iw*dw*dt));
879                     }
880             }
881
882               /* Allocate fft arrays */
```

```
883             rt    = ealloc1float(nfft);
884             ct    = ealloc1complex(nw);
885
886             memcpy(rt, trace, nt*FSIZE); /* copia los primeros nt*FSIZE valores de la traza de
887             entrada al vector rt*/
888             memset((void *) (rt + nt), (int) '\0', (nfft-nt)*FSIZE);/*copia en los primeros
889             (nfft-nt)*FSIZE lugares de la posición apuntada por (rt+nt) el dato convertido a
890             entero '\0'*/
891             pfarc(1, nfft, rt, ct);/*Ejecuta la FFT en 1D de real a complejo usando el método
892             de los factores primos*/
893
894           for(iw=nwf-itaper;iw<=nwf+itaper;++iw){
895                   itemp = iw-(nwf-itaper);
896                   ct[iw].r = taper[itemp]*ct[iw].r;
897                   ct[iw].i = taper[itemp]*ct[iw].i;
898           }
899           for(iw=nwf+itaper+1;iw<nw;++iw){
900                   ct[iw].r = 0.;
901                   ct[iw].i = 0.;
902           }
903
904                   if(!ls){/*niega el valor de ls */
905                   for(iw=0; iw<=nwf+itaper; ++iw){
906                           /* phase shifts PI/4    */
907                           temp = (ct[iw].r-ct[iw].i)*amp[iw]*const2;
908                           ct[iw].i = (ct[iw].r+ct[iw].i)*amp[iw]*const2;
909                           ct[iw].r = temp;
910                       }
911           } else {
912                   for(iw=0; iw<=nwf+itaper; ++iw){
913                           ct[iw].i = ct[iw].i*amp[iw];
914                           ct[iw].r = ct[iw].r*amp[iw];
915                   }
916           }
917             pfacr(-1, nfft, ct, rt);
918
919             /* Load traces back in */
920           for (it=0; it<nt; ++it) trace[it] = rt[it];
921
922             /* Integrate traces   */
923           for(iw=0; iw<=nwf+itaper; ++iw){
924                   ct[iw].i = ct[iw].i*ampi[iw];
925                   ct[iw].r = ct[iw].r*ampi[iw];
926           }
927             pfacr(-1, nfft, ct, rt);
928             for (it=0; it<m; ++it)  trf[it] = rt[nfft-m+it];
929             for (it=0; it<nt+m; ++it)  trf[it+m] = rt[it];
930
931           free1float(rt);
932           free1complex(ct);
933   }
```

Script B.1: Código fuente de la función sukdmig2d.c

Apéndice C

Código fuente de la función pfarc.

C.1 Función PFARC

```
1
2  void pfarc (int isign, int n, float rz[], complex cz[])
3  /**************************************************************************
4  Prime factor fft:  real to complex transform
5  **************************************************************************
6  Input:
7  isign        sign of isign is the sign of exponent in fourier kernel
8  n            length of transform; must be even (see notes below)
9  rz           array[n] of real values (may be equivalenced to cz)
10
11 Output:
12 cz           array[n/2+1] of complex values (may be equivalenced to rz)
13 **************************************************************************
14 Notes:
15 Because pfarc uses pfacc to do most of the work, n must be even
16 and n/2 must be a valid length for pfacc.  The simplest way to
17 obtain a valid n is via n = npfar(nmin).  A more optimal n can be
18 obtained with npfaro.
19 **************************************************************************
20 References:
21 Inspired by, but differ significantly from Press et al, 1988, NR in C, p. 417.
22
23 Also, see notes and references for function pfacc.
24 **************************************************************************
25 Author:  Dave Hale, Colorado School of Mines, 06/13/89
26 **************************************************************************/
27 {
28     int i,ir,ii,jr,ji,no2;
29     float *z,tempr,tempi,sumr,sumi,difr,difi;
30     double wr,wi,wpr,wpi,wtemp,theta;
31
32     /* copy input to output while scaling */
33     z = (float*)cz;/*Convierte el puntero cz de complejo a flotante para que z lo pueda apuntar*/
34     for (i=0; i<n; i++)
```

```
35          z[i] = 0.5*rz[i];/*Guarda en cz una versión escalada a la mitad de rz*/
36
37      /* do complex to complex transform */
38      pfacc(isign,n/2,cz);
39
40      /* fix dc and nyquist */
41      z[n] = 2.0*(z[0]-z[1]);
42      z[0] = 2.0*(z[0]+z[1]);
43      z[n+1] = 0.0;
44      z[1] = 0.0;
45
46      /* initialize cosine-sine recurrence */
47      theta = 2.0*PI/(double)n;
48      if (isign<0) theta = -theta;
49      wtemp = sin(0.5*theta);
50      wpr = -2.0*wtemp*wtemp;
51      wpi = sin(theta);
52      wr = 1.0+wpr;
53      wi = wpi;
54
55      /* twiddle */
56      no2 = n/2;
57      for (ir=2,ii=3,jr=n-2,ji=n-1; ir<=no2; ir+=2,ii+=2,jr-=2,ji-=2) {
58          sumr = z[ir]+z[jr];
59          sumi = z[ii]+z[ji];
60          difr = z[ir]-z[jr];
61          difi = z[ii]-z[ji];
62          tempr = wi*difr+wr*sumi;
63          tempi = wi*sumi-wr*difr;
64          z[ir] = sumr+tempr;
65          z[ii] = difi+tempi;
66          z[jr] = sumr-tempr;
67          z[ji] = tempi-difi;
68          wtemp = wr;
69          wr += wr*wpr-wi*wpi;
70          wi += wi*wpr+wtemp*wpi;
71      }
72  }
```

Script C.1: Código fuente de la función pfarc.c

Apéndice D

Código fuente de la función pfacr.

D.1 Función PFACR

```
1
2  void pfacr (int isign, int n, complex cz[], float rz[])
3  /**************************************************************************
4  Prime factor fft:  complex to real transform
5  **************************************************************************
6  Input:
7  isign        sign of isign is the sign of exponent in fourier kernel
8  n            length of transform (see notes below)
9  cz           array[n/2+1] of complex values (may be equivalenced to rz)
10
11 Output:
12 rz           array[n] of real values (may be equivalenced to cz)
13 **************************************************************************
14 Notes:
15 Because pfacr uses pfacc to do most of the work, n must be even
16 and n/2 must be a valid length for pfacc.  The simplest way to
17 obtain a valid n is via n = npfar(nmin).  A more optimal n can be
18 obtained with npfaro.
19 **************************************************************************
20 References:
21 Inspired by, but differ significantly from Press et al, 1988, NR in C, p. 417.
22
23 Also, see notes and references for function pfacc.
24 **************************************************************************
25 Author:  Dave Hale, Colorado School of Mines, 06/13/89
26 **************************************************************************/
27 {
28     int i,ir,ii,jr,ji,no2;
29     float *z,tempr,tempi,sumr,sumi,difr,difi;
30     double wr,wi,wpr,wpi,wtemp,theta;
31
32     /* copy input to output and fix dc and nyquist */
33     z = (float*)cz;
34     for (i=2; i<n; i++)
```

```
35          rz[i] = z[i];
36      rz[1] = z[0]-z[n];
37      rz[0] = z[0]+z[n];
38      z = rz;
39
40      /* initialize cosine-sine recurrence */
41      theta = 2.0*PI/(double)n;
42      if (isign>0) theta = -theta;
43      wtemp = sin(0.5*theta);
44      wpr = -2.0*wtemp*wtemp;
45      wpi = sin(theta);
46      wr = 1.0+wpr;
47      wi = wpi;
48
49      /* twiddle */
50      no2 = n/2;
51      for (ir=2,ii=3,jr=n-2,ji=n-1; ir<=no2; ir+=2,ii+=2,jr-=2,ji-=2) {
52          sumr = z[ir]+z[jr];
53          sumi = z[ii]+z[ji];
54          difr = z[ir]-z[jr];
55          difi = z[ii]-z[ji];
56          tempr = wi*difr-wr*sumi;
57          tempi = wi*sumi+wr*difr;
58          z[ir] = sumr+tempr;
59          z[ii] = difi+tempi;
60          z[jr] = sumr-tempr;
61          z[ji] = tempi-difi;
62          wtemp = wr;
63          wr += wr*wpr-wi*wpi;
64          wi += wi*wpr+wtemp*wpi;
65      }
66
67      /* do complex to complex transform */
68      pfacc(isign,n/2,(complex*)z);
69  }
```

Script D.1: Código fuente de la función pfacr.c

Apéndice E

Código fuente de la función pfacc.

E.1 Función PFACC

```
void pfacc (int isign, int n, complex cz[])
/*************************************************************************
Prime factor fft:  complex to complex transform, in place
*************************************************************************
Input:
isign           sign of isign is the sign of exponent in fourier kernel
n               length of transform (see notes below)
z               array[n] of complex numbers to be transformed in place

Output:
z               array[n] of complex numbers transformed
*************************************************************************
Notes:
n must be factorable into mutually prime factors taken
from the set {2,3,4,5,7,8,9,11,13,16}.  in other words,
        n = 2**p * 3**q * 5**r * 7**s * 11**t * 13**u
where
        0 <= p <= 4,  0 <= q <= 2,  0 <= r,s,t,u <= 1
is required for pfa to yield meaningful results.  this
restriction implies that n is restricted to the range
        1 <= n <= 720720 (= 5*7*9*11*13*16)
*************************************************************************
References:
Temperton, C., 1985, Implementation of a self-sorting
in-place prime factor fft algorithm:  Journal of
Computational Physics, v. 58, p. 283-299.

Temperton, C., 1988, A new set of minimum-add rotated
rotated dft modules: Journal of Computational Physics,
v. 75, p. 190-198.
*************************************************************************
Author:  Dave Hale, Colorado School of Mines, 04/27/89
*************************************************************************/
{
```

```
35          static int kfax[] = { 16,13,11,9,8,7,5,4,3,2 };
36          register float *z=(float*)cz;
37          register int j00,j01,j2,j3,j4,j5,j6,j7,j8,j9,j10,j11,j12,j13,j14,j15,jt;
38          int nleft,jfax,ifac,jfac,jinc,jmax,ndiv,m,mm=0,mu=0,l;
39          float t1r,t1i,t2r,t2i,t3r,t3i,t4r,t4i,t5r,t5i,
40                  t6r,t6i,t7r,t7i,t8r,t8i,t9r,t9i,t10r,t10i,
41                  t11r,t11i,t12r,t12i,t13r,t13i,t14r,t14i,t15r,t15i,
42                  t16r,t16i,t17r,t17i,t18r,t18i,t19r,t19i,t20r,t20i,
43                  t21r,t21i,t22r,t22i,t23r,t23i,t24r,t24i,t25r,t25i,
44                  t26r,t26i,t27r,t27i,t28r,t28i,t29r,t29i,t30r,t30i,
45                  t31r,t31i,t32r,t32i,t33r,t33i,t34r,t34i,t35r,t35i,
46                  t36r,t36i,t37r,t37i,t38r,t38i,t39r,t39i,t40r,t40i,
47                  t41r,t41i,t42r,t42i,
48                  y1r,y1i,y2r,y2i,y3r,y3i,y4r,y4i,y5r,y5i,
49                  y6r,y6i,y7r,y7i,y8r,y8i,y9r,y9i,y10r,y10i,
50                  y11r,y11i,y12r,y12i,y13r,y13i,y14r,y14i,y15r,y15i,
51                  c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
52
53          /* keep track of n left after dividing by factors */
54          nleft = n;
55
56          /* begin loop over possible factors (from biggest to smallest) */
57          for (jfax=0; jfax<NFAX; jfax++) {
58
59                  /* skip if not a mutually prime factor of n */
60          ifac = kfax[jfax];/*en estas tres lineas se empieza a descomponer n en sus*/
61          ndiv = nleft/ifac;/*valores primos para despues a partir de cada uno de los*/
62          if (ndiv*ifac!=nleft) continue;/*mismos determinar los siguientes parámetros*/
63
64                  /* update n left and determine n divided by factor */
65          nleft = ndiv;
66          m = n/ifac;
67
68                  /* determine rotation factor mu and stride mm */
69          for (jfac=1; jfac<=ifac; jfac++) {
70                          mu = jfac;
71                          mm = jfac*m;
72                          if (mm%ifac==1) break;
73                  }
74
75                  /* adjust rotation factor for sign of transform */
76          if (isign<0) mu = ifac-mu;
77
78                  /* compute stride, limit, and pointers */
79          jinc = 2*mm;
80                  jmax = 2*n;
81          j00 = 0;
82          j01 = j00+jinc;
83
84                  /* if factor is 2 */
85          if (ifac==2) {
86                          for (l=0; l<m; l++) {
87                                  t1r = z[j00]-z[j01];
```

```
88                                      t1i = z[j00+1]-z[j01+1];
89                                      z[j00] = z[j00]+z[j01];
90                                      z[j00+1] = z[j00+1]+z[j01+1];
91                                      z[j01] = t1r;
92                                      z[j01+1] = t1i;
93                                      jt = j01+2;
94                                      j01 = j00+2;
95                                      j00 = jt;
96                              }
97                              continue;
98                      }
99              j2 = j01+jinc;
100             if (j2>=jmax) j2 = j2-jmax;
101
102                     /* if factor is 3 */
103             if (ifac==3) {
104                             if (mu==1)
105                                     c1 = P866;
106                             else
107                                     c1 = -P866;
108                             for (l=0; l<m; l++) {
109                                     t1r = z[j01]+z[j2];
110                                     t1i = z[j01+1]+z[j2+1];
111                                     y1r = z[j00]-0.5*t1r;
112                                     y1i = z[j00+1]-0.5*t1i;
113                                     y2r = c1*(z[j01]-z[j2]);
114                                     y2i = c1*(z[j01+1]-z[j2+1]);
115                                     z[j00] = z[j00]+t1r;
116                                     z[j00+1] = z[j00+1]+t1i;
117                                     z[j01] = y1r-y2i;
118                                     z[j01+1] = y1i+y2r;
119                                     z[j2] = y1r+y2i;
120                                     z[j2+1] = y1i-y2r;
121                                     jt = j2+2;
122                                     j2 = j01+2;
123                                     j01 = j00+2;
124                                     j00 = jt;
125                             }
126                             continue;
127                     }
128                     j3 = j2+jinc;
129                     if (j3>=jmax) j3 = j3-jmax;
130
131                     /* if factor is 4 */
132                     if (ifac==4) {
133                             if (mu==1)
134                                     c1 = 1.0;
135                             else
136                                     c1 = -1.0;
137                             for (l=0; l<m; l++) {
138                                     t1r = z[j00]+z[j2];
139                                     t1i = z[j00+1]+z[j2+1];
140                                     t2r = z[j01]+z[j3];
```

```
141                             t2i = z[j01+1]+z[j3+1];
142                             y1r = z[j00]-z[j2];
143                             y1i = z[j00+1]-z[j2+1];
144                             y3r = c1*(z[j01]-z[j3]);
145                             y3i = c1*(z[j01+1]-z[j3+1]);
146                             z[j00] = t1r+t2r;
147                             z[j00+1] = t1i+t2i;
148                             z[j01] = y1r-y3i;
149                             z[j01+1] = y1i+y3r;
150                             z[j2] = t1r-t2r;
151                             z[j2+1] = t1i-t2i;
152                             z[j3] = y1r+y3i;
153                             z[j3+1] = y1i-y3r;
154                             jt = j3+2;
155                             j3 = j2+2;
156                             j2 = j01+2;
157                             j01 = j00+2;
158                             j00 = jt;
159                     }
160                     continue;
161             }
162             j4 = j3+jinc;
163             if (j4>=jmax) j4 = j4-jmax;
164
165             /* if factor is 5 */
166             if (ifac==5) {
167                     if (mu==1) {
168                             c1 = P559;
169                             c2 = P951;
170                             c3 = P587;
171                     } else if (mu==2) {
172                             c1 = -P559;
173                             c2 = P587;
174                             c3 = -P951;
175                     } else if (mu==3) {
176                             c1 = -P559;
177                             c2 = -P587;
178                             c3 = P951;
179                     } else {
180                             c1 = P559;
181                             c2 = -P951;
182                             c3 = -P587;
183                     }
184                     for (l=0; l<m; l++) {
185                             t1r = z[j01]+z[j4];
186                             t1i = z[j01+1]+z[j4+1];
187                             t2r = z[j2]+z[j3];
188                             t2i = z[j2+1]+z[j3+1];
189                             t3r = z[j01]-z[j4];
190                             t3i = z[j01+1]-z[j4+1];
191                             t4r = z[j2]-z[j3];
192                             t4i = z[j2+1]-z[j3+1];
193                             t5r = t1r+t2r;
```

```
194                             t5i = t1i+t2i;
195                             t6r = c1*(t1r-t2r);
196                             t6i = c1*(t1i-t2i);
197                             t7r = z[j00]-0.25*t5r;
198                             t7i = z[j00+1]-0.25*t5i;
199                             y1r = t7r+t6r;
200                             y1i = t7i+t6i;
201                             y2r = t7r-t6r;
202                             y2i = t7i-t6i;
203                             y3r = c3*t3r-c2*t4r;
204                             y3i = c3*t3i-c2*t4i;
205                             y4r = c2*t3r+c3*t4r;
206                             y4i = c2*t3i+c3*t4i;
207                             z[j00] = z[j00]+t5r;
208                             z[j00+1] = z[j00+1]+t5i;
209                             z[j01] = y1r-y4i;
210                             z[j01+1] = y1i+y4r;
211                             z[j2] = y2r-y3i;
212                             z[j2+1] = y2i+y3r;
213                             z[j3] = y2r+y3i;
214                             z[j3+1] = y2i-y3r;
215                             z[j4] = y1r+y4i;
216                             z[j4+1] = y1i-y4r;
217                             jt = j4+2;
218                             j4 = j3+2;
219                             j3 = j2+2;
220                             j2 = j01+2;
221                             j01 = j00+2;
222                             j00 = jt;
223                     }
224                     continue;
225             }
226             j5 = j4+jinc;
227             if (j5>=jmax) j5 = j5-jmax;
228             j6 = j5+jinc;
229             if (j6>=jmax) j6 = j6-jmax;
230
231             /* if factor is 7 */
232             if (ifac==7) {
233                     if (mu==1) {
234                             c1 = P623;
235                             c2 = -P222;
236                             c3 = -P900;
237                             c4 = P781;
238                             c5 = P974;
239                             c6 = P433;
240                     } else if (mu==2) {
241                             c1 = -P222;
242                             c2 = -P900;
243                             c3 = P623;
244                             c4 = P974;
245                             c5 = -P433;
246                             c6 = -P781;
```

```
247                     } else if (mu==3) {
248                             c1 = -P900;
249                             c2 = P623;
250                             c3 = -P222;
251                             c4 = P433;
252                             c5 = -P781;
253                             c6 = P974;
254                     } else if (mu==4) {
255                             c1 = -P900;
256                             c2 = P623;
257                             c3 = -P222;
258                             c4 = -P433;
259                             c5 = P781;
260                             c6 = -P974;
261                     } else if (mu==5) {
262                             c1 = -P222;
263                             c2 = -P900;
264                             c3 = P623;
265                             c4 = -P974;
266                             c5 = P433;
267                             c6 = P781;
268                     } else {
269                             c1 = P623;
270                             c2 = -P222;
271                             c3 = -P900;
272                             c4 = -P781;
273                             c5 = -P974;
274                             c6 = -P433;
275                     }
276                     for (l=0; l<m; l++) {
277                             t1r = z[j01]+z[j6];
278                             t1i = z[j01+1]+z[j6+1];
279                             t2r = z[j2]+z[j5];
280                             t2i = z[j2+1]+z[j5+1];
281                             t3r = z[j3]+z[j4];
282                             t3i = z[j3+1]+z[j4+1];
283                             t4r = z[j01]-z[j6];
284                             t4i = z[j01+1]-z[j6+1];
285                             t5r = z[j2]-z[j5];
286                             t5i = z[j2+1]-z[j5+1];
287                             t6r = z[j3]-z[j4];
288                             t6i = z[j3+1]-z[j4+1];
289                             t7r = z[j00]-0.5*t3r;
290                             t7i = z[j00+1]-0.5*t3i;
291                             t8r = t1r-t3r;
292                             t8i = t1i-t3i;
293                             t9r = t2r-t3r;
294                             t9i = t2i-t3i;
295                             y1r = t7r+c1*t8r+c2*t9r;
296                             y1i = t7i+c1*t8i+c2*t9i;
297                             y2r = t7r+c2*t8r+c3*t9r;
298                             y2i = t7i+c2*t8i+c3*t9i;
299                             y3r = t7r+c3*t8r+c1*t9r;
```

```
300                             y3i = t7i+c3*t8i+c1*t9i;
301                             y4r = c6*t4r-c4*t5r+c5*t6r;
302                             y4i = c6*t4i-c4*t5i+c5*t6i;
303                             y5r = c5*t4r-c6*t5r-c4*t6r;
304                             y5i = c5*t4i-c6*t5i-c4*t6i;
305                             y6r = c4*t4r+c5*t5r+c6*t6r;
306                             y6i = c4*t4i+c5*t5i+c6*t6i;
307                             z[j00] = z[j00]+t1r+t2r+t3r;
308                             z[j00+1] = z[j00+1]+t1i+t2i+t3i;
309                             z[j01] = y1r-y6i;
310                             z[j01+1] = y1i+y6r;
311                             z[j2] = y2r-y5i;
312                             z[j2+1] = y2i+y5r;
313                             z[j3] = y3r-y4i;
314                             z[j3+1] = y3i+y4r;
315                             z[j4] = y3r+y4i;
316                             z[j4+1] = y3i-y4r;
317                             z[j5] = y2r+y5i;
318                             z[j5+1] = y2i-y5r;
319                             z[j6] = y1r+y6i;
320                             z[j6+1] = y1i-y6r;
321                             jt = j6+2;
322                             j6 = j5+2;
323                             j5 = j4+2;
324                             j4 = j3+2;
325                             j3 = j2+2;
326                             j2 = j01+2;
327                             j01 = j00+2;
328                             j00 = jt;
329                     }
330                     continue;
331             }
332             j7 = j6+jinc;
333             if (j7>=jmax) j7 = j7-jmax;
334
335             /* if factor is 8 */
336             if (ifac==8) {
337                     if (mu==1) {
338                             c1 = 1.0;
339                             c2 = P707;
340                     } else if (mu==3) {
341                             c1 = -1.0;
342                             c2 = -P707;
343                     } else if (mu==5) {
344                             c1 = 1.0;
345                             c2 = -P707;
346                     } else {
347                             c1 = -1.0;
348                             c2 = P707;
349                     }
350                     c3 = c1*c2;
351                     for (l=0; l<m; l++) {
352                             t1r = z[j00]+z[j4];
```

```
353                     t1i = z[j00+1]+z[j4+1];
354                     t2r = z[j00]-z[j4];
355                     t2i = z[j00+1]-z[j4+1];
356                     t3r = z[j01]+z[j5];
357                     t3i = z[j01+1]+z[j5+1];
358                     t4r = z[j01]-z[j5];
359                     t4i = z[j01+1]-z[j5+1];
360                     t5r = z[j2]+z[j6];
361                     t5i = z[j2+1]+z[j6+1];
362                     t6r = c1*(z[j2]-z[j6]);
363                     t6i = c1*(z[j2+1]-z[j6+1]);
364                     t7r = z[j3]+z[j7];
365                     t7i = z[j3+1]+z[j7+1];
366                     t8r = z[j3]-z[j7];
367                     t8i = z[j3+1]-z[j7+1];
368                     t9r = t1r+t5r;
369                     t9i = t1i+t5i;
370                     t10r = t3r+t7r;
371                     t10i = t3i+t7i;
372                     t11r = c2*(t4r-t8r);
373                     t11i = c2*(t4i-t8i);
374                     t12r = c3*(t4r+t8r);
375                     t12i = c3*(t4i+t8i);
376                     y1r = t2r+t11r;
377                     y1i = t2i+t11i;
378                     y2r = t1r-t5r;
379                     y2i = t1i-t5i;
380                     y3r = t2r-t11r;
381                     y3i = t2i-t11i;
382                     y5r = t12r-t6r;
383                     y5i = t12i-t6i;
384                     y6r = c1*(t3r-t7r);
385                     y6i = c1*(t3i-t7i);
386                     y7r = t12r+t6r;
387                     y7i = t12i+t6i;
388                     z[j00] = t9r+t10r;
389                     z[j00+1] = t9i+t10i;
390                     z[j01] = y1r-y7i;
391                     z[j01+1] = y1i+y7r;
392                     z[j2] = y2r-y6i;
393                     z[j2+1] = y2i+y6r;
394                     z[j3] = y3r-y5i;
395                     z[j3+1] = y3i+y5r;
396                     z[j4] = t9r-t10r;
397                     z[j4+1] = t9i-t10i;
398                     z[j5] = y3r+y5i;
399                     z[j5+1] = y3i-y5r;
400                     z[j6] = y2r+y6i;
401                     z[j6+1] = y2i-y6r;
402                     z[j7] = y1r+y7i;
403                     z[j7+1] = y1i-y7r;
404                     jt = j7+2;
405                     j7 = j6+2;
```

```
406                             j6 = j5+2;
407                             j5 = j4+2;
408                             j4 = j3+2;
409                             j3 = j2+2;
410                             j2 = j01+2;
411                             j01 = j00+2;
412                             j00 = jt;
413                     }
414                     continue;
415             }
416             j8 = j7+jinc;
417             if (j8>=jmax) j8 = j8-jmax;
418
419             /* if factor is 9 */
420             if (ifac==9) {
421                     if (mu==1) {
422                             c1 = P866;
423                             c2 = P766;
424                             c3 = P642;
425                             c4 = P173;
426                             c5 = P984;
427                     } else if (mu==2) {
428                             c1 = -P866;
429                             c2 = P173;
430                             c3 = P984;
431                             c4 = -P939;
432                             c5 = P342;
433                     } else if (mu==4) {
434                             c1 = P866;
435                             c2 = -P939;
436                             c3 = P342;
437                             c4 = P766;
438                             c5 = -P642;
439                     } else if (mu==5) {
440                             c1 = -P866;
441                             c2 = -P939;
442                             c3 = -P342;
443                             c4 = P766;
444                             c5 = P642;
445                     } else if (mu==7) {
446                             c1 = P866;
447                             c2 = P173;
448                             c3 = -P984;
449                             c4 = -P939;
450                             c5 = -P342;
451                     } else {
452                             c1 = -P866;
453                             c2 = P766;
454                             c3 = -P642;
455                             c4 = P173;
456                             c5 = -P984;
457                     }
458                     c6 = c1*c2;
```

```
459                     c7 = c1*c3;
460                     c8 = c1*c4;
461                     c9 = c1*c5;
462                     for (l=0; l<m; l++) {
463                             t1r = z[j3]+z[j6];
464                             t1i = z[j3+1]+z[j6+1];
465                             t2r = z[j00]-0.5*t1r;
466                             t2i = z[j00+1]-0.5*t1i;
467                             t3r = c1*(z[j3]-z[j6]);
468                             t3i = c1*(z[j3+1]-z[j6+1]);
469                             t4r = z[j00]+t1r;
470                             t4i = z[j00+1]+t1i;
471                             t5r = z[j4]+z[j7];
472                             t5i = z[j4+1]+z[j7+1];
473                             t6r = z[j01]-0.5*t5r;
474                             t6i = z[j01+1]-0.5*t5i;
475                             t7r = z[j4]-z[j7];
476                             t7i = z[j4+1]-z[j7+1];
477                             t8r = z[j01]+t5r;
478                             t8i = z[j01+1]+t5i;
479                             t9r = z[j2]+z[j5];
480                             t9i = z[j2+1]+z[j5+1];
481                             t10r = z[j8]-0.5*t9r;
482                             t10i = z[j8+1]-0.5*t9i;
483                             t11r = z[j2]-z[j5];
484                             t11i = z[j2+1]-z[j5+1];
485                             t12r = z[j8]+t9r;
486                             t12i = z[j8+1]+t9i;
487                             t13r = t8r+t12r;
488                             t13i = t8i+t12i;
489                             t14r = t6r+t10r;
490                             t14i = t6i+t10i;
491                             t15r = t6r-t10r;
492                             t15i = t6i-t10i;
493                             t16r = t7r+t11r;
494                             t16i = t7i+t11i;
495                             t17r = t7r-t11r;
496                             t17i = t7i-t11i;
497                             t18r = c2*t14r-c7*t17r;
498                             t18i = c2*t14i-c7*t17i;
499                             t19r = c4*t14r+c9*t17r;
500                             t19i = c4*t14i+c9*t17i;
501                             t20r = c3*t15r+c6*t16r;
502                             t20i = c3*t15i+c6*t16i;
503                             t21r = c5*t15r-c8*t16r;
504                             t21i = c5*t15i-c8*t16i;
505                             t22r = t18r+t19r;
506                             t22i = t18i+t19i;
507                             t23r = t20r-t21r;
508                             t23i = t20i-t21i;
509                             y1r = t2r+t18r;
510                             y1i = t2i+t18i;
511                             y2r = t2r+t19r;
```

```
512                             y2i = t2i+t19i;
513                             y3r = t4r-0.5*t13r;
514                             y3i = t4i-0.5*t13i;
515                             y4r = t2r-t22r;
516                             y4i = t2i-t22i;
517                             y5r = t3r-t23r;
518                             y5i = t3i-t23i;
519                             y6r = c1*(t8r-t12r);
520                             y6i = c1*(t8i-t12i);
521                             y7r = t21r-t3r;
522                             y7i = t21i-t3i;
523                             y8r = t3r+t20r;
524                             y8i = t3i+t20i;
525                             z[j00] = t4r+t13r;
526                             z[j00+1] = t4i+t13i;
527                             z[j01] = y1r-y8i;
528                             z[j01+1] = y1i+y8r;
529                             z[j2] = y2r-y7i;
530                             z[j2+1] = y2i+y7r;
531                             z[j3] = y3r-y6i;
532                             z[j3+1] = y3i+y6r;
533                             z[j4] = y4r-y5i;
534                             z[j4+1] = y4i+y5r;
535                             z[j5] = y4r+y5i;
536                             z[j5+1] = y4i-y5r;
537                             z[j6] = y3r+y6i;
538                             z[j6+1] = y3i-y6r;
539                             z[j7] = y2r+y7i;
540                             z[j7+1] = y2i-y7r;
541                             z[j8] = y1r+y8i;
542                             z[j8+1] = y1i-y8r;
543                             jt = j8+2;
544                             j8 = j7+2;
545                             j7 = j6+2;
546                             j6 = j5+2;
547                             j5 = j4+2;
548                             j4 = j3+2;
549                             j3 = j2+2;
550                             j2 = j01+2;
551                             j01 = j00+2;
552                             j00 = jt;
553                     }
554                     continue;
555             }
556             j9 = j8+jinc;
557             if (j9>=jmax) j9 = j9-jmax;
558             j10 = j9+jinc;
559             if (j10>=jmax) j10 = j10-jmax;
560
561             /* if factor is 11 */
562             if (ifac==11) {
563                     if (mu==1) {
564                             c1 = P841;
```

```
565                     c2 = P415;
566                     c3 = -P142;
567                     c4 = -P654;
568                     c5 = -P959;
569                     c6 = P540;
570                     c7 = P909;
571                     c8 = P989;
572                     c9 = P755;
573                     c10 = P281;
574                 } else if (mu==2) {
575                     c1 = P415;
576                     c2 = -P654;
577                     c3 = -P959;
578                     c4 = -P142;
579                     c5 = P841;
580                     c6 = P909;
581                     c7 = P755;
582                     c8 = -P281;
583                     c9 = -P989;
584                     c10 = -P540;
585                 } else if (mu==3) {
586                     c1 = -P142;
587                     c2 = -P959;
588                     c3 = P415;
589                     c4 = P841;
590                     c5 = -P654;
591                     c6 = P989;
592                     c7 = -P281;
593                     c8 = -P909;
594                     c9 = P540;
595                     c10 = P755;
596                 } else if (mu==4) {
597                     c1 = -P654;
598                     c2 = -P142;
599                     c3 = P841;
600                     c4 = -P959;
601                     c5 = P415;
602                     c6 = P755;
603                     c7 = -P989;
604                     c8 = P540;
605                     c9 = P281;
606                     c10 = -P909;
607                 } else if (mu==5) {
608                     c1 = -P959;
609                     c2 = P841;
610                     c3 = -P654;
611                     c4 = P415;
612                     c5 = -P142;
613                     c6 = P281;
614                     c7 = -P540;
615                     c8 = P755;
616                     c9 = -P909;
617                     c10 = P989;
```

```
618                     } else if (mu==6) {
619                             c1 = -P959;
620                             c2 = P841;
621                             c3 = -P654;
622                             c4 = P415;
623                             c5 = -P142;
624                             c6 = -P281;
625                             c7 = P540;
626                             c8 = -P755;
627                             c9 = P909;
628                             c10 = -P989;
629                     } else if (mu==7) {
630                             c1 = -P654;
631                             c2 = -P142;
632                             c3 = P841;
633                             c4 = -P959;
634                             c5 = P415;
635                             c6 = -P755;
636                             c7 = P989;
637                             c8 = -P540;
638                             c9 = -P281;
639                             c10 = P909;
640                     } else if (mu==8) {
641                             c1 = -P142;
642                             c2 = -P959;
643                             c3 = P415;
644                             c4 = P841;
645                             c5 = -P654;
646                             c6 = -P989;
647                             c7 = P281;
648                             c8 = P909;
649                             c9 = -P540;
650                             c10 = -P755;
651                     } else if (mu==9) {
652                             c1 = P415;
653                             c2 = -P654;
654                             c3 = -P959;
655                             c4 = -P142;
656                             c5 = P841;
657                             c6 = -P909;
658                             c7 = -P755;
659                             c8 = P281;
660                             c9 = P989;
661                             c10 = P540;
662                     } else {
663                             c1 = P841;
664                             c2 = P415;
665                             c3 = -P142;
666                             c4 = -P654;
667                             c5 = -P959;
668                             c6 = -P540;
669                             c7 = -P909;
670                             c8 = -P989;
```

```
671                             c9 = -P755;
672                             c10 = -P281;
673                     }
674                     for (l=0; l<m; l++) {
675                             t1r = z[j01]+z[j10];
676                             t1i = z[j01+1]+z[j10+1];
677                             t2r = z[j2]+z[j9];
678                             t2i = z[j2+1]+z[j9+1];
679                             t3r = z[j3]+z[j8];
680                             t3i = z[j3+1]+z[j8+1];
681                             t4r = z[j4]+z[j7];
682                             t4i = z[j4+1]+z[j7+1];
683                             t5r = z[j5]+z[j6];
684                             t5i = z[j5+1]+z[j6+1];
685                             t6r = z[j01]-z[j10];
686                             t6i = z[j01+1]-z[j10+1];
687                             t7r = z[j2]-z[j9];
688                             t7i = z[j2+1]-z[j9+1];
689                             t8r = z[j3]-z[j8];
690                             t8i = z[j3+1]-z[j8+1];
691                             t9r = z[j4]-z[j7];
692                             t9i = z[j4+1]-z[j7+1];
693                             t10r = z[j5]-z[j6];
694                             t10i = z[j5+1]-z[j6+1];
695                             t11r = z[j00]-0.5*t5r;
696                             t11i = z[j00+1]-0.5*t5i;
697                             t12r = t1r-t5r;
698                             t12i = t1i-t5i;
699                             t13r = t2r-t5r;
700                             t13i = t2i-t5i;
701                             t14r = t3r-t5r;
702                             t14i = t3i-t5i;
703                             t15r = t4r-t5r;
704                             t15i = t4i-t5i;
705                             y1r = t11r+c1*t12r+c2*t13r+c3*t14r+c4*t15r;
706                             y1i = t11i+c1*t12i+c2*t13i+c3*t14i+c4*t15i;
707                             y2r = t11r+c2*t12r+c4*t13r+c5*t14r+c3*t15r;
708                             y2i = t11i+c2*t12i+c4*t13i+c5*t14i+c3*t15i;
709                             y3r = t11r+c3*t12r+c5*t13r+c2*t14r+c1*t15r;
710                             y3i = t11i+c3*t12i+c5*t13i+c2*t14i+c1*t15i;
711                             y4r = t11r+c4*t12r+c3*t13r+c1*t14r+c5*t15r;
712                             y4i = t11i+c4*t12i+c3*t13i+c1*t14i+c5*t15i;
713                             y5r = t11r+c5*t12r+c1*t13r+c4*t14r+c2*t15r;
714                             y5i = t11i+c5*t12i+c1*t13i+c4*t14i+c2*t15i;
715                             y6r = c10*t6r-c6*t7r+c9*t8r-c7*t9r+c8*t10r;
716                             y6i = c10*t6i-c6*t7i+c9*t8i-c7*t9i+c8*t10i;
717                             y7r = c9*t6r-c8*t7r+c6*t8r+c10*t9r-c7*t10r;
718                             y7i = c9*t6i-c8*t7i+c6*t8i+c10*t9i-c7*t10i;
719                             y8r = c8*t6r-c10*t7r-c7*t8r+c6*t9r+c9*t10r;
720                             y8i = c8*t6i-c10*t7i-c7*t8i+c6*t9i+c9*t10i;
721                             y9r = c7*t6r+c9*t7r-c10*t8r-c8*t9r-c6*t10r;
722                             y9i = c7*t6i+c9*t7i-c10*t8i-c8*t9i-c6*t10i;
723                             y10r = c6*t6r+c7*t7r+c8*t8r+c9*t9r+c10*t10r;
```

```
724                             y10i = c6*t6i+c7*t7i+c8*t8i+c9*t9i+c10*t10i;
725                             z[j00] = z[j00]+t1r+t2r+t3r+t4r+t5r;
726                             z[j00+1] = z[j00+1]+t1i+t2i+t3i+t4i+t5i;
727                             z[j01] = y1r-y10i;
728                             z[j01+1] = y1i+y10r;
729                             z[j2] = y2r-y9i;
730                             z[j2+1] = y2i+y9r;
731                             z[j3] = y3r-y8i;
732                             z[j3+1] = y3i+y8r;
733                             z[j4] = y4r-y7i;
734                             z[j4+1] = y4i+y7r;
735                             z[j5] = y5r-y6i;
736                             z[j5+1] = y5i+y6r;
737                             z[j6] = y5r+y6i;
738                             z[j6+1] = y5i-y6r;
739                             z[j7] = y4r+y7i;
740                             z[j7+1] = y4i-y7r;
741                             z[j8] = y3r+y8i;
742                             z[j8+1] = y3i-y8r;
743                             z[j9] = y2r+y9i;
744                             z[j9+1] = y2i-y9r;
745                             z[j10] = y1r+y10i;
746                             z[j10+1] = y1i-y10r;
747                             jt = j10+2;
748                             j10 = j9+2;
749                             j9 = j8+2;
750                             j8 = j7+2;
751                             j7 = j6+2;
752                             j6 = j5+2;
753                             j5 = j4+2;
754                             j4 = j3+2;
755                             j3 = j2+2;
756                             j2 = j01+2;
757                             j01 = j00+2;
758                             j00 = jt;
759                     }
760                     continue;
761             }
762             j11 = j10+jinc;
763             if (j11>=jmax) j11 = j11-jmax;
764             j12 = j11+jinc;
765             if (j12>=jmax) j12 = j12-jmax;
766
767             /* if factor is 13 */
768             if (ifac==13) {
769                     if (mu==1) {
770                             c1 = P885;
771                             c2 = P568;
772                             c3 = P120;
773                             c4 = -P354;
774                             c5 = -P748;
775                             c6 = -P970;
776                             c7 = P464;
```

```
777                     c8 = P822;
778                     c9 = P992;
779                     c10 = P935;
780                     c11 = P663;
781                     c12 = P239;
782                 } else if (mu==2) {
783                     c1 = P568;
784                     c2 = -P354;
785                     c3 = -P970;
786                     c4 = -P748;
787                     c5 = P120;
788                     c6 = P885;
789                     c7 = P822;
790                     c8 = P935;
791                     c9 = P239;
792                     c10 = -P663;
793                     c11 = -P992;
794                     c12 = -P464;
795                 } else if (mu==3) {
796                     c1 = P120;
797                     c2 = -P970;
798                     c3 = -P354;
799                     c4 = P885;
800                     c5 = P568;
801                     c6 = -P748;
802                     c7 = P992;
803                     c8 = P239;
804                     c9 = -P935;
805                     c10 = -P464;
806                     c11 = P822;
807                     c12 = P663;
808                 } else if (mu==4) {
809                     c1 = -P354;
810                     c2 = -P748;
811                     c3 = P885;
812                     c4 = P120;
813                     c5 = -P970;
814                     c6 = P568;
815                     c7 = P935;
816                     c8 = -P663;
817                     c9 = -P464;
818                     c10 = P992;
819                     c11 = -P239;
820                     c12 = -P822;
821                 } else if (mu==5) {
822                     c1 = -P748;
823                     c2 = P120;
824                     c3 = P568;
825                     c4 = -P970;
826                     c5 = P885;
827                     c6 = -P354;
828                     c7 = P663;
829                     c8 = -P992;
```

```
830                     c9 = P822;
831                     c10 = -P239;
832                     c11 = -P464;
833                     c12 = P935;
834                 } else if (mu==6) {
835                     c1 = -P970;
836                     c2 = P885;
837                     c3 = -P748;
838                     c4 = P568;
839                     c5 = -P354;
840                     c6 = P120;
841                     c7 = P239;
842                     c8 = -P464;
843                     c9 = P663;
844                     c10 = -P822;
845                     c11 = P935;
846                     c12 = -P992;
847                 } else if (mu==7) {
848                     c1 = -P970;
849                     c2 = P885;
850                     c3 = -P748;
851                     c4 = P568;
852                     c5 = -P354;
853                     c6 = P120;
854                     c7 = -P239;
855                     c8 = P464;
856                     c9 = -P663;
857                     c10 = P822;
858                     c11 = -P935;
859                     c12 = P992;
860                 } else if (mu==8) {
861                     c1 = -P748;
862                     c2 = P120;
863                     c3 = P568;
864                     c4 = -P970;
865                     c5 = P885;
866                     c6 = -P354;
867                     c7 = -P663;
868                     c8 = P992;
869                     c9 = -P822;
870                     c10 = P239;
871                     c11 = P464;
872                     c12 = -P935;
873                 } else if (mu==9) {
874                     c1 = -P354;
875                     c2 = -P748;
876                     c3 = P885;
877                     c4 = P120;
878                     c5 = -P970;
879                     c6 = P568;
880                     c7 = -P935;
881                     c8 = P663;
882                     c9 = P464;
```

```
883                         c10 = -P992;
884                         c11 = P239;
885                         c12 = P822;
886                 } else if (mu==10) {
887                         c1 = P120;
888                         c2 = -P970;
889                         c3 = -P354;
890                         c4 = P885;
891                         c5 = P568;
892                         c6 = -P748;
893                         c7 = -P992;
894                         c8 = -P239;
895                         c9 = P935;
896                         c10 = P464;
897                         c11 = -P822;
898                         c12 = -P663;
899                 } else if (mu==11) {
900                         c1 = P568;
901                         c2 = -P354;
902                         c3 = -P970;
903                         c4 = -P748;
904                         c5 = P120;
905                         c6 = P885;
906                         c7 = -P822;
907                         c8 = -P935;
908                         c9 = -P239;
909                         c10 = P663;
910                         c11 = P992;
911                         c12 = P464;
912                 } else {
913                         c1 = P885;
914                         c2 = P568;
915                         c3 = P120;
916                         c4 = -P354;
917                         c5 = -P748;
918                         c6 = -P970;
919                         c7 = -P464;
920                         c8 = -P822;
921                         c9 = -P992;
922                         c10 = -P935;
923                         c11 = -P663;
924                         c12 = -P239;
925                 }
926                 for (l=0; l<m; l++) {
927                         t1r = z[j01]+z[j12];
928                         t1i = z[j01+1]+z[j12+1];
929                         t2r = z[j2]+z[j11];
930                         t2i = z[j2+1]+z[j11+1];
931                         t3r = z[j3]+z[j10];
932                         t3i = z[j3+1]+z[j10+1];
933                         t4r = z[j4]+z[j9];
934                         t4i = z[j4+1]+z[j9+1];
935                         t5r = z[j5]+z[j8];
```

```
936                 t5i = z[j5+1]+z[j8+1];
937                 t6r = z[j6]+z[j7];
938                 t6i = z[j6+1]+z[j7+1];
939                 t7r = z[j01]-z[j12];
940                 t7i = z[j01+1]-z[j12+1];
941                 t8r = z[j2]-z[j11];
942                 t8i = z[j2+1]-z[j11+1];
943                 t9r = z[j3]-z[j10];
944                 t9i = z[j3+1]-z[j10+1];
945                 t10r = z[j4]-z[j9];
946                 t10i = z[j4+1]-z[j9+1];
947                 t11r = z[j5]-z[j8];
948                 t11i = z[j5+1]-z[j8+1];
949                 t12r = z[j6]-z[j7];
950                 t12i = z[j6+1]-z[j7+1];
951                 t13r = z[j00]-0.5*t6r;
952                 t13i = z[j00+1]-0.5*t6i;
953                 t14r = t1r-t6r;
954                 t14i = t1i-t6i;
955                 t15r = t2r-t6r;
956                 t15i = t2i-t6i;
957                 t16r = t3r-t6r;
958                 t16i = t3i-t6i;
959                 t17r = t4r-t6r;
960                 t17i = t4i-t6i;
961                 t18r = t5r-t6r;
962                 t18i = t5i-t6i;
963                 y1r = t13r+c1*t14r+c2*t15r+c3*t16r+c4*t17r+c5*t18r;
964                 y1i = t13i+c1*t14i+c2*t15i+c3*t16i+c4*t17i+c5*t18i;
965                 y2r = t13r+c2*t14r+c4*t15r+c6*t16r+c5*t17r+c3*t18r;
966                 y2i = t13i+c2*t14i+c4*t15i+c6*t16i+c5*t17i+c3*t18i;
967                 y3r = t13r+c3*t14r+c6*t15r+c4*t16r+c1*t17r+c2*t18r;
968                 y3i = t13i+c3*t14i+c6*t15i+c4*t16i+c1*t17i+c2*t18i;
969                 y4r = t13r+c4*t14r+c5*t15r+c1*t16r+c3*t17r+c6*t18r;
970                 y4i = t13i+c4*t14i+c5*t15i+c1*t16i+c3*t17i+c6*t18i;
971                 y5r = t13r+c5*t14r+c3*t15r+c2*t16r+c6*t17r+c1*t18r;
972                 y5i = t13i+c5*t14i+c3*t15i+c2*t16i+c6*t17i+c1*t18i;
973                 y6r = t13r+c6*t14r+c1*t15r+c5*t16r+c2*t17r+c4*t18r;
974                 y6i = t13i+c6*t14i+c1*t15i+c5*t16i+c2*t17i+c4*t18i;
975                 y7r = c12*t7r-c7*t8r+c11*t9r-c8*t10r+c10*t11r-c9*t12r;
976                 y7i = c12*t7i-c7*t8i+c11*t9i-c8*t10i+c10*t11i-c9*t12i;
977                 y8r = c11*t7r-c9*t8r+c8*t9r-c12*t10r-c7*t11r+c10*t12r;
978                 y8i = c11*t7i-c9*t8i+c8*t9i-c12*t10i-c7*t11i+c10*t12i;
979                 y9r = c10*t7r-c11*t8r-c7*t9r+c9*t10r-c12*t11r-c8*t12r;
980                 y9i = c10*t7i-c11*t8i-c7*t9i+c9*t10i-c12*t11i-c8*t12i;
981                 y10r = c9*t7r+c12*t8r-c10*t9r-c7*t10r+c8*t11r+c11*t12r;
982                 y10i = c9*t7i+c12*t8i-c10*t9i-c7*t10i+c8*t11i+c11*t12i;
983                 y11r = c8*t7r+c10*t8r+c12*t9r-c11*t10r-c9*t11r-c7*t12r;
984                 y11i = c8*t7i+c10*t8i+c12*t9i-c11*t10i-c9*t11i-c7*t12i;
985                 y12r = c7*t7r+c8*t8r+c9*t9r+c10*t10r+c11*t11r+c12*t12r;
986                 y12i = c7*t7i+c8*t8i+c9*t9i+c10*t10i+c11*t11i+c12*t12i;
987                 z[j00] = z[j00]+t1r+t2r+t3r+t4r+t5r+t6r;
988                 z[j00+1] = z[j00+1]+t1i+t2i+t3i+t4i+t5i+t6i;
```

```
989                                     z[j01] = y1r-y12i;
990                                     z[j01+1] = y1i+y12r;
991                                     z[j2] = y2r-y11i;
992                                     z[j2+1] = y2i+y11r;
993                                     z[j3] = y3r-y10i;
994                                     z[j3+1] = y3i+y10r;
995                                     z[j4] = y4r-y9i;
996                                     z[j4+1] = y4i+y9r;
997                                     z[j5] = y5r-y8i;
998                                     z[j5+1] = y5i+y8r;
999                                     z[j6] = y6r-y7i;
1000                                    z[j6+1] = y6i+y7r;
1001                                    z[j7] = y6r+y7i;
1002                                    z[j7+1] = y6i-y7r;
1003                                    z[j8] = y5r+y8i;
1004                                    z[j8+1] = y5i-y8r;
1005                                    z[j9] = y4r+y9i;
1006                                    z[j9+1] = y4i-y9r;
1007                                    z[j10] = y3r+y10i;
1008                                    z[j10+1] = y3i-y10r;
1009                                    z[j11] = y2r+y11i;
1010                                    z[j11+1] = y2i-y11r;
1011                                    z[j12] = y1r+y12i;
1012                                    z[j12+1] = y1i-y12r;
1013                                    jt = j12+2;
1014                                    j12 = j11+2;
1015                                    j11 = j10+2;
1016                                    j10 = j9+2;
1017                                    j9 = j8+2;
1018                                    j8 = j7+2;
1019                                    j7 = j6+2;
1020                                    j6 = j5+2;
1021                                    j5 = j4+2;
1022                                    j4 = j3+2;
1023                                    j3 = j2+2;
1024                                    j2 = j01+2;
1025                                    j01 = j00+2;
1026                                    j00 = jt;
1027                            }
1028                            continue;
1029                    }
1030                    j13 = j12+jinc;
1031                    if (j13>=jmax) j13 = j13-jmax;
1032                    j14 = j13+jinc;
1033                    if (j14>=jmax) j14 = j14-jmax;
1034                    j15 = j14+jinc;
1035                    if (j15>=jmax) j15 = j15-jmax;
1036
1037                    /* if factor is 16 */
1038                    if (ifac==16) {
1039                            if (mu==1) {
1040                                    c1 = 1.0;
1041                                    c2 = P923;
```

```
1042                    c3 = P382;
1043                    c4 = P707;
1044            } else if (mu==3) {
1045                    c1 = -1.0;
1046                    c2 = P382;
1047                    c3 = P923;
1048                    c4 = -P707;
1049            } else if (mu==5) {
1050                    c1 = 1.0;
1051                    c2 = -P382;
1052                    c3 = P923;
1053                    c4 = -P707;
1054            } else if (mu==7) {
1055                    c1 = -1.0;
1056                    c2 = -P923;
1057                    c3 = P382;
1058                    c4 = P707;
1059            } else if (mu==9) {
1060                    c1 = 1.0;
1061                    c2 = -P923;
1062                    c3 = -P382;
1063                    c4 = P707;
1064            } else if (mu==11) {
1065                    c1 = -1.0;
1066                    c2 = -P382;
1067                    c3 = -P923;
1068                    c4 = -P707;
1069            } else if (mu==13) {
1070                    c1 = 1.0;
1071                    c2 = P382;
1072                    c3 = -P923;
1073                    c4 = -P707;
1074            } else {
1075                    c1 = -1.0;
1076                    c2 = P923;
1077                    c3 = -P382;
1078                    c4 = P707;
1079            }
1080            c5 = c1*c4;
1081            c6 = c1*c3;
1082            c7 = c1*c2;
1083            for (l=0; l<m; l++) {
1084                    t1r = z[j00]+z[j8];
1085                    t1i = z[j00+1]+z[j8+1];
1086                    t2r = z[j4]+z[j12];
1087                    t2i = z[j4+1]+z[j12+1];
1088                    t3r = z[j00]-z[j8];
1089                    t3i = z[j00+1]-z[j8+1];
1090                    t4r = c1*(z[j4]-z[j12]);
1091                    t4i = c1*(z[j4+1]-z[j12+1]);
1092                    t5r = t1r+t2r;
1093                    t5i = t1i+t2i;
1094                    t6r = t1r-t2r;
```

```
1095                    t6i = t1i-t2i;
1096                    t7r = z[j01]+z[j9];
1097                    t7i = z[j01+1]+z[j9+1];
1098                    t8r = z[j5]+z[j13];
1099                    t8i = z[j5+1]+z[j13+1];
1100                    t9r = z[j01]-z[j9];
1101                    t9i = z[j01+1]-z[j9+1];
1102                    t10r = z[j5]-z[j13];
1103                    t10i = z[j5+1]-z[j13+1];
1104                    t11r = t7r+t8r;
1105                    t11i = t7i+t8i;
1106                    t12r = t7r-t8r;
1107                    t12i = t7i-t8i;
1108                    t13r = z[j2]+z[j10];
1109                    t13i = z[j2+1]+z[j10+1];
1110                    t14r = z[j6]+z[j14];
1111                    t14i = z[j6+1]+z[j14+1];
1112                    t15r = z[j2]-z[j10];
1113                    t15i = z[j2+1]-z[j10+1];
1114                    t16r = z[j6]-z[j14];
1115                    t16i = z[j6+1]-z[j14+1];
1116                    t17r = t13r+t14r;
1117                    t17i = t13i+t14i;
1118                    t18r = c4*(t15r-t16r);
1119                    t18i = c4*(t15i-t16i);
1120                    t19r = c5*(t15r+t16r);
1121                    t19i = c5*(t15i+t16i);
1122                    t20r = c1*(t13r-t14r);
1123                    t20i = c1*(t13i-t14i);
1124                    t21r = z[j3]+z[j11];
1125                    t21i = z[j3+1]+z[j11+1];
1126                    t22r = z[j7]+z[j15];
1127                    t22i = z[j7+1]+z[j15+1];
1128                    t23r = z[j3]-z[j11];
1129                    t23i = z[j3+1]-z[j11+1];
1130                    t24r = z[j7]-z[j15];
1131                    t24i = z[j7+1]-z[j15+1];
1132                    t25r = t21r+t22r;
1133                    t25i = t21i+t22i;
1134                    t26r = t21r-t22r;
1135                    t26i = t21i-t22i;
1136                    t27r = t9r+t24r;
1137                    t27i = t9i+t24i;
1138                    t28r = t10r+t23r;
1139                    t28i = t10i+t23i;
1140                    t29r = t9r-t24r;
1141                    t29i = t9i-t24i;
1142                    t30r = t10r-t23r;
1143                    t30i = t10i-t23i;
1144                    t31r = t5r+t17r;
1145                    t31i = t5i+t17i;
1146                    t32r = t11r+t25r;
1147                    t32i = t11i+t25i;
```

```
1148                    t33r = t3r+t18r;
1149                    t33i = t3i+t18i;
1150                    t34r = c2*t29r-c6*t30r;
1151                    t34i = c2*t29i-c6*t30i;
1152                    t35r = t3r-t18r;
1153                    t35i = t3i-t18i;
1154                    t36r = c7*t27r-c3*t28r;
1155                    t36i = c7*t27i-c3*t28i;
1156                    t37r = t4r+t19r;
1157                    t37i = t4i+t19i;
1158                    t38r = c3*t27r+c7*t28r;
1159                    t38i = c3*t27i+c7*t28i;
1160                    t39r = t4r-t19r;
1161                    t39i = t4i-t19i;
1162                    t40r = c6*t29r+c2*t30r;
1163                    t40i = c6*t29i+c2*t30i;
1164                    t41r = c4*(t12r-t26r);
1165                    t41i = c4*(t12i-t26i);
1166                    t42r = c5*(t12r+t26r);
1167                    t42i = c5*(t12i+t26i);
1168                    y1r = t33r+t34r;
1169                    y1i = t33i+t34i;
1170                    y2r = t6r+t41r;
1171                    y2i = t6i+t41i;
1172                    y3r = t35r+t40r;
1173                    y3i = t35i+t40i;
1174                    y4r = t5r-t17r;
1175                    y4i = t5i-t17i;
1176                    y5r = t35r-t40r;
1177                    y5i = t35i-t40i;
1178                    y6r = t6r-t41r;
1179                    y6i = t6i-t41i;
1180                    y7r = t33r-t34r;
1181                    y7i = t33i-t34i;
1182                    y9r = t38r-t37r;
1183                    y9i = t38i-t37i;
1184                    y10r = t42r-t20r;
1185                    y10i = t42i-t20i;
1186                    y11r = t36r+t39r;
1187                    y11i = t36i+t39i;
1188                    y12r = c1*(t11r-t25r);
1189                    y12i = c1*(t11i-t25i);
1190                    y13r = t36r-t39r;
1191                    y13i = t36i-t39i;
1192                    y14r = t42r+t20r;
1193                    y14i = t42i+t20i;
1194                    y15r = t38r+t37r;
1195                    y15i = t38i+t37i;
1196                    z[j00] = t31r+t32r;
1197                    z[j00+1] = t31i+t32i;
1198                    z[j01] = y1r-y15i;
1199                    z[j01+1] = y1i+y15r;
1200                    z[j2] = y2r-y14i;
```

```
1201                                    z[j2+1] = y2i+y14r;
1202                                    z[j3] = y3r-y13i;
1203                                    z[j3+1] = y3i+y13r;
1204                                    z[j4] = y4r-y12i;
1205                                    z[j4+1] = y4i+y12r;
1206                                    z[j5] = y5r-y11i;
1207                                    z[j5+1] = y5i+y11r;
1208                                    z[j6] = y6r-y10i;
1209                                    z[j6+1] = y6i+y10r;
1210                                    z[j7] = y7r-y9i;
1211                                    z[j7+1] = y7i+y9r;
1212                                    z[j8] = t31r-t32r;
1213                                    z[j8+1] = t31i-t32i;
1214                                    z[j9] = y7r+y9i;
1215                                    z[j9+1] = y7i-y9r;
1216                                    z[j10] = y6r+y10i;
1217                                    z[j10+1] = y6i-y10r;
1218                                    z[j11] = y5r+y11i;
1219                                    z[j11+1] = y5i-y11r;
1220                                    z[j12] = y4r+y12i;
1221                                    z[j12+1] = y4i-y12r;
1222                                    z[j13] = y3r+y13i;
1223                                    z[j13+1] = y3i-y13r;
1224                                    z[j14] = y2r+y14i;
1225                                    z[j14+1] = y2i-y14r;
1226                                    z[j15] = y1r+y15i;
1227                                    z[j15+1] = y1i-y15r;
1228                                    jt = j15+2;
1229                                    j15 = j14+2;
1230                                    j14 = j13+2;
1231                                    j13 = j12+2;
1232                                    j12 = j11+2;
1233                                    j11 = j10+2;
1234                                    j10 = j9+2;
1235                                    j9 = j8+2;
1236                                    j8 = j7+2;
1237                                    j7 = j6+2;
1238                                    j6 = j5+2;
1239                                    j5 = j4+2;
1240                                    j4 = j3+2;
1241                                    j3 = j2+2;
1242                                    j2 = j01+2;
1243                                    j01 = j00+2;
1244                                    j00 = jt;
1245                                }
1246                                continue;
1247                            }
1248                        }
1249                    }
```

Script E.1: Código fuente de la función pfacc.c

Apéndice F

Datapath final del procesador específico para el factor de 7.

F.1 Datapath final del procesador específico

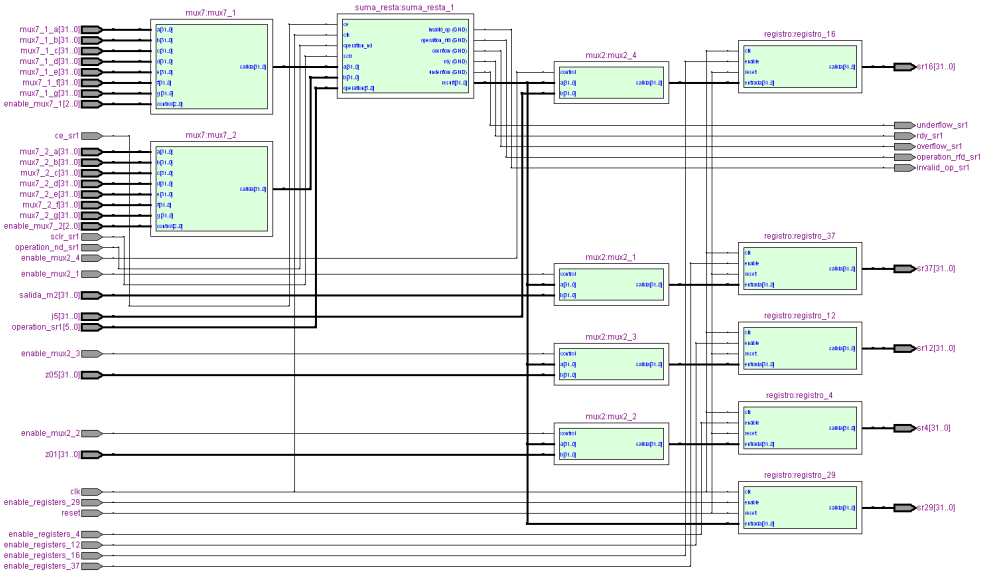


Figura F.1: *Datapath* de la sección de suma-resta SR1.

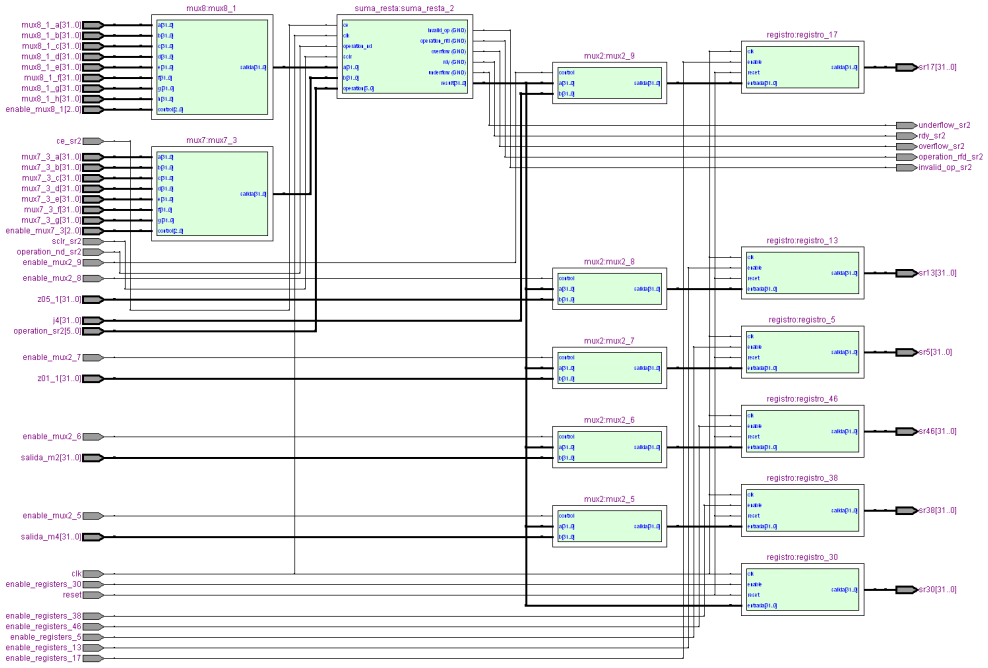


Figura F.2: *Datapath* de la sección de suma-resta SR2.

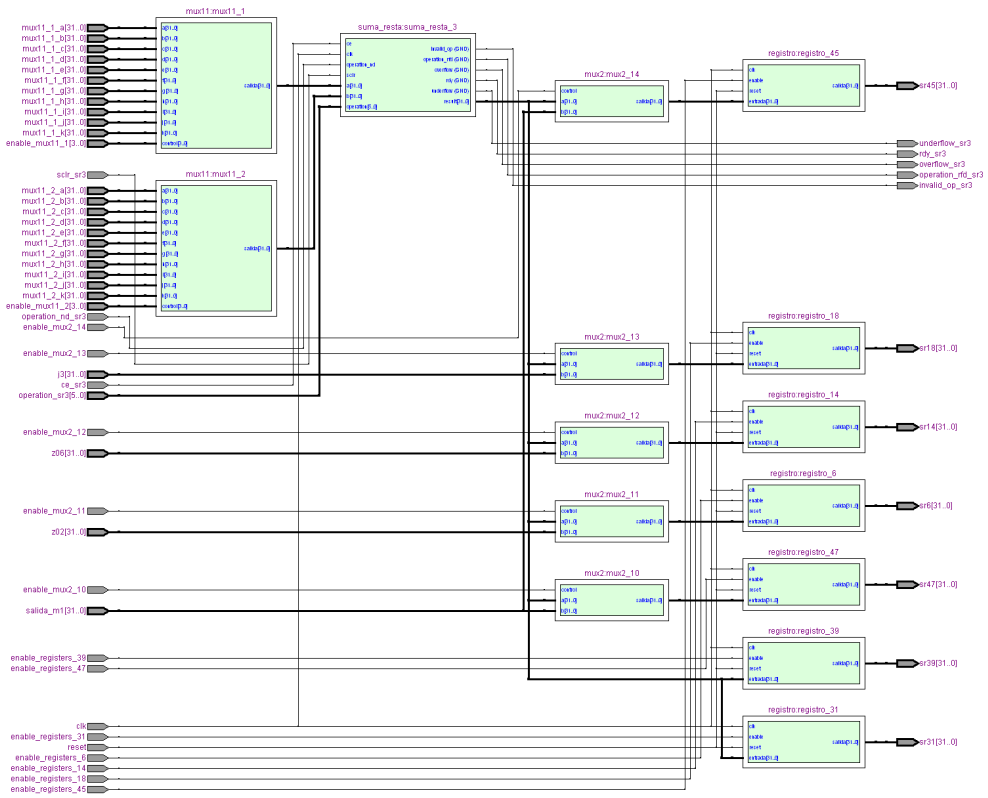


Figura F.3: *Datapath* de la sección de suma-resta SR3.

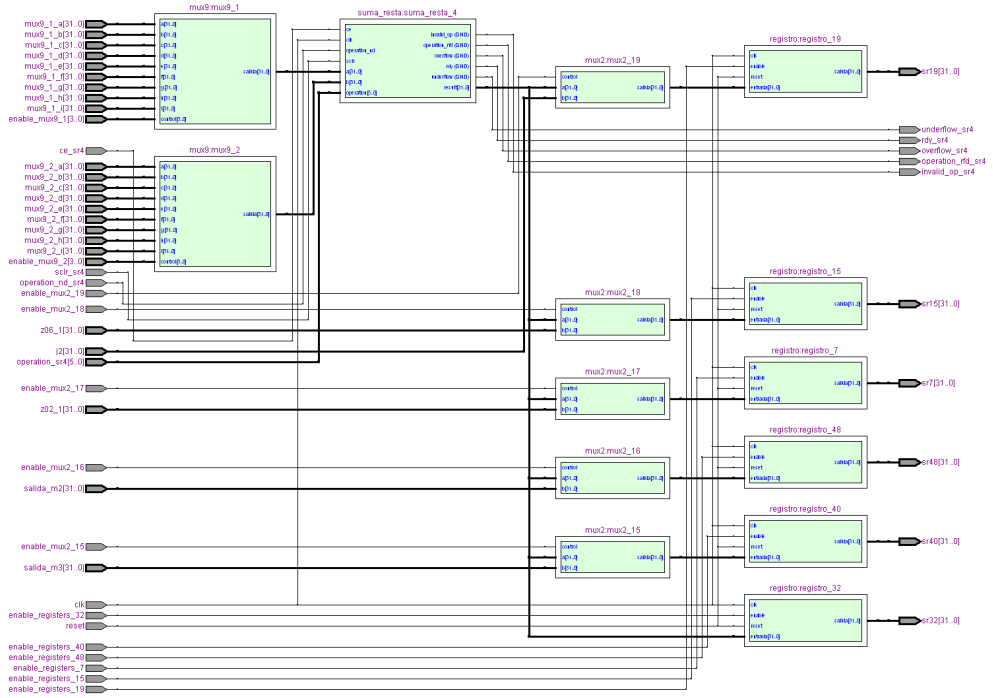


Figura F.4: *Datapath* de la sección de suma-resta SR4.

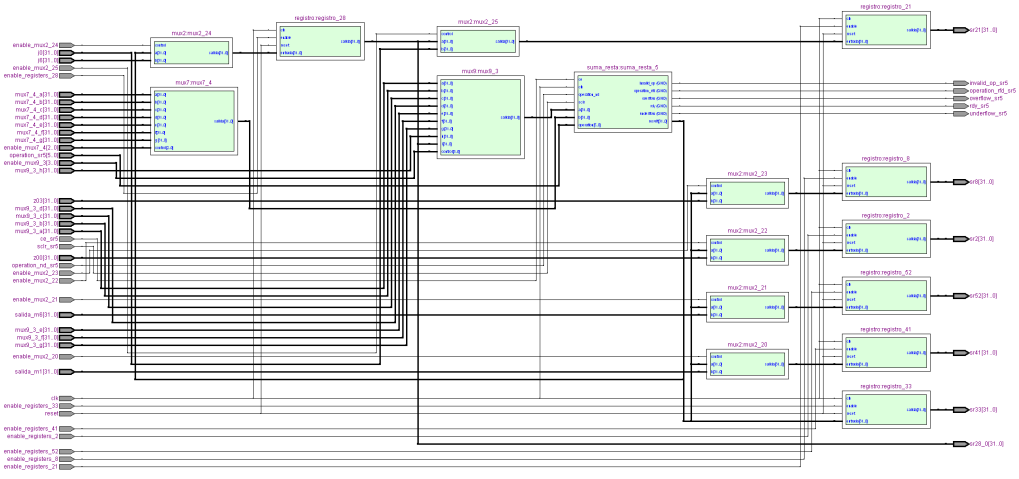


Figura F.5: *Datapath* de la sección de suma-resta SR5.

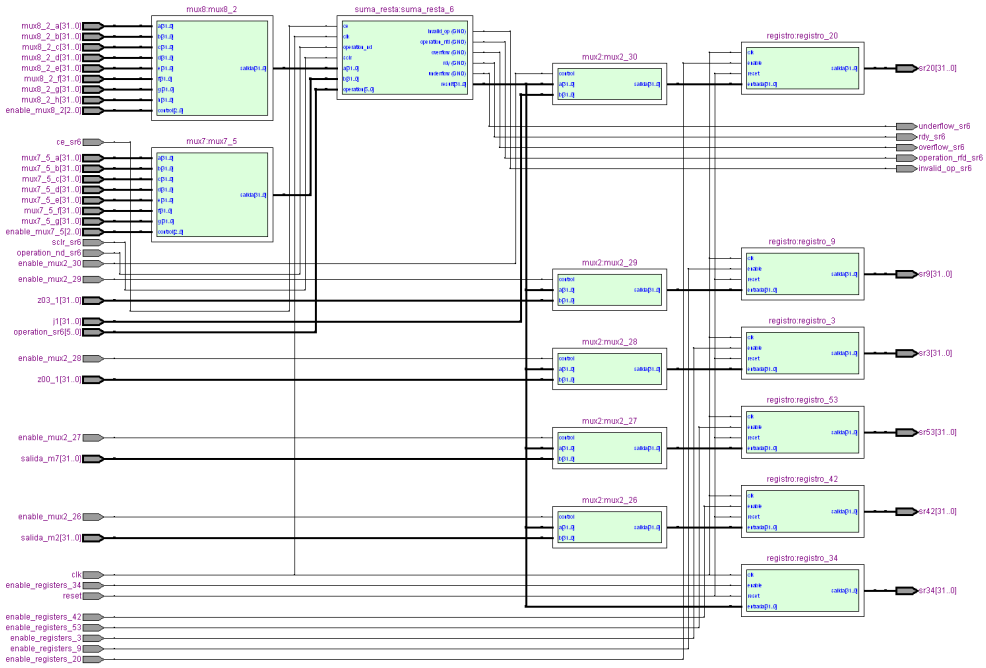


Figura F.6: *Datapath* de la sección de suma-resta SR6.

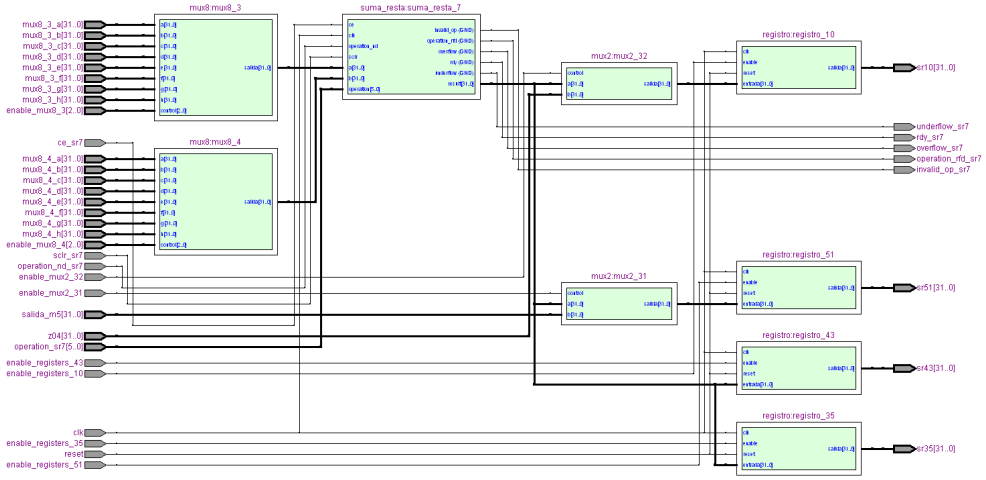


Figura F.7: *Datapath* de la sección de suma-resta SR7.

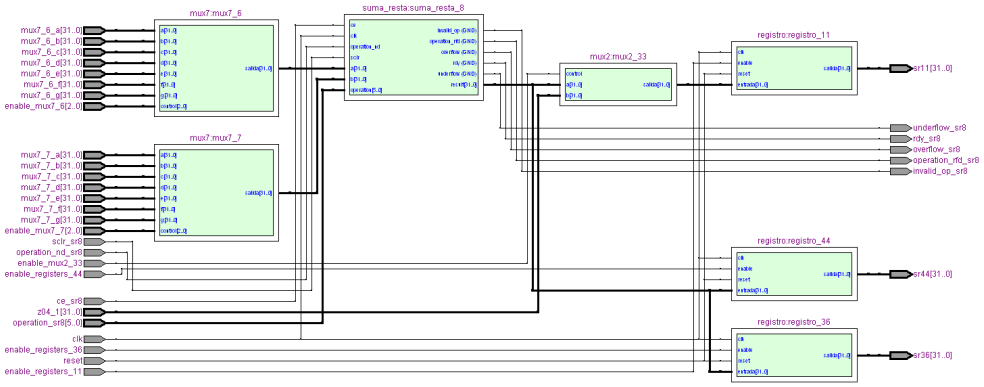


Figura F.8: *Datapath* de la sección de suma-resta SR8.

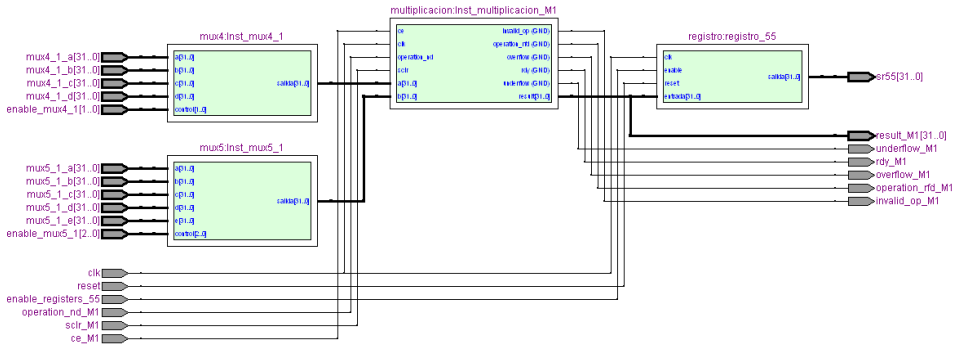


Figura F.9: *Datapath* de la sección de multiplicación M1.

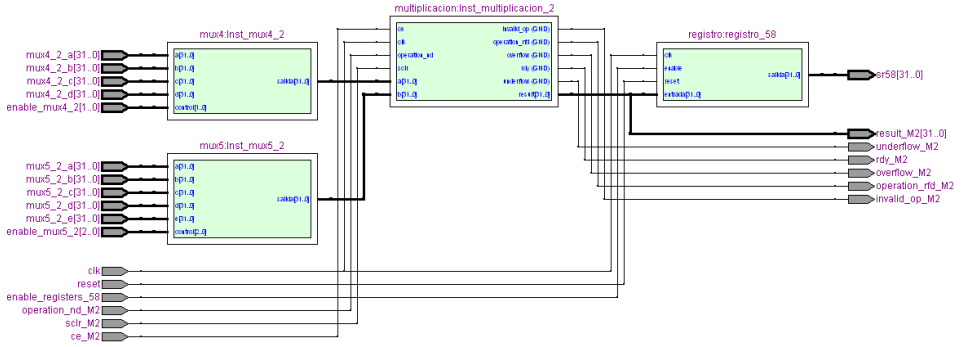


Figura F.10: *Datapath* de la sección de multiplicación M2.

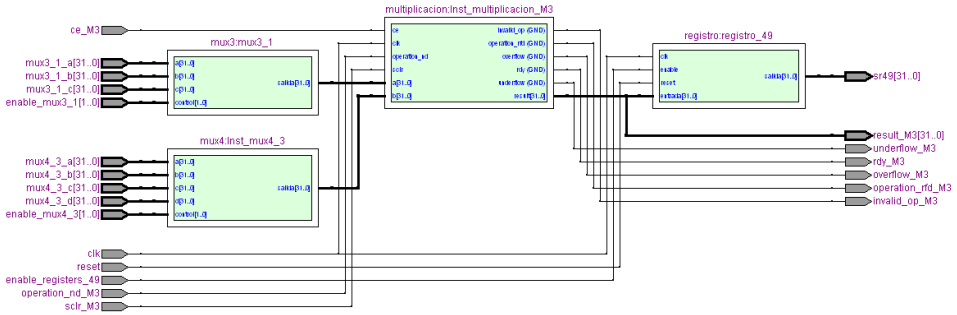


Figura F.11: *Datapath* de la sección de multiplicación M3.

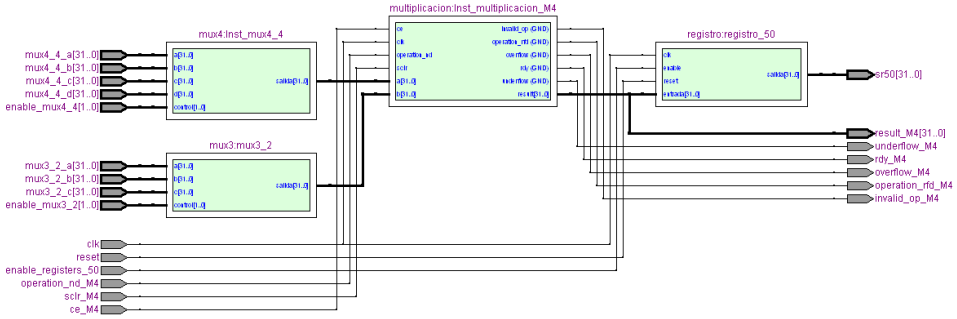


Figura F.12: *Datapath* de la sección de multiplicación M4.

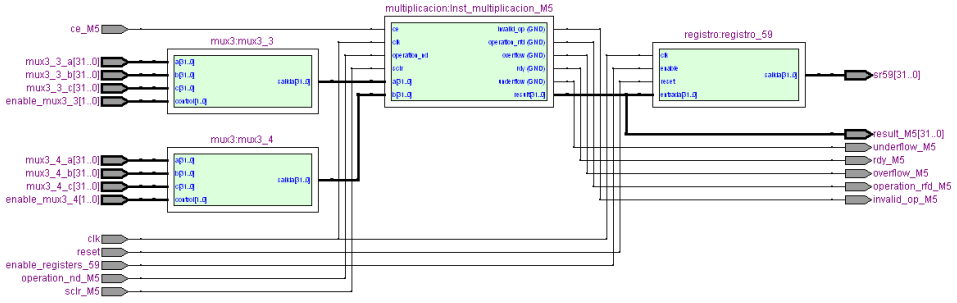


Figura F.13: *Datapath* de la sección de multiplicación M5.

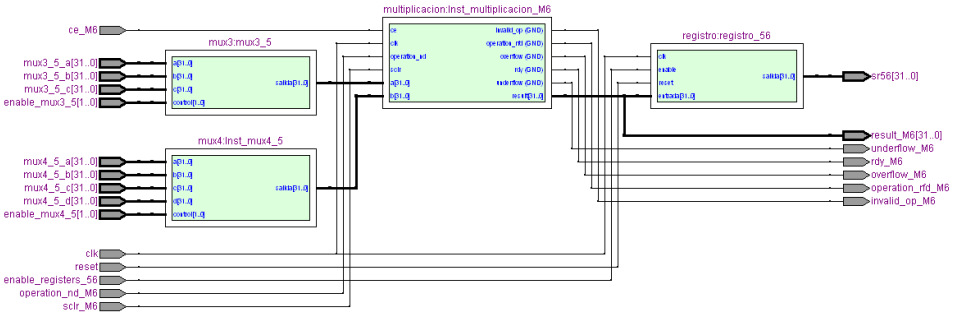


Figura F.14: *Datapath* de la sección de multiplicación M6.

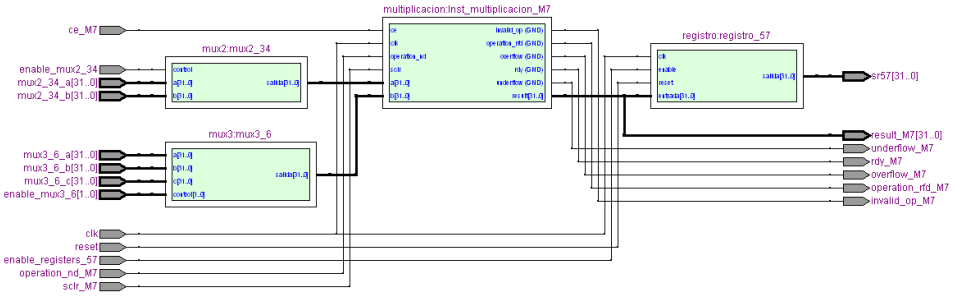


Figura F.15: *Datapath* de la sección de multiplicación M7.

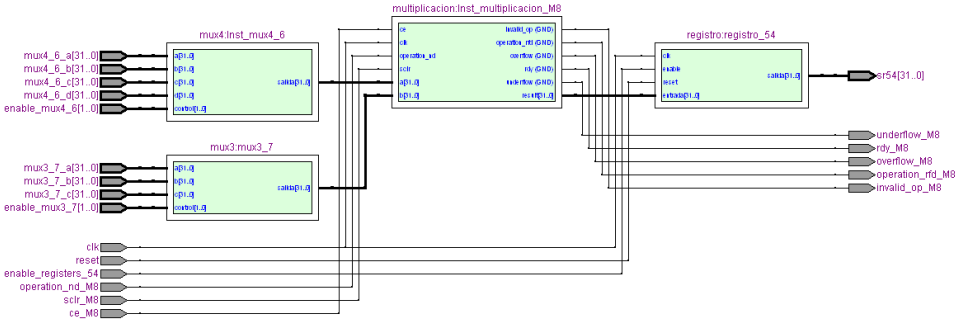


Figura F.16: *Datapath* de la sección de multiplicación M8.

Apéndice G

Descripción en VHDL del procesador específico para el factor de 7.

G.1 Sección principal.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:24:38 08/11/2010
6  -- Design Name:
7  -- Module Name:    principal - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity principal is
31     Port (
```

```
32 mu : in  STD_LOGIC_VECTOR (3 downto 0);
33 inicio : in  STD_LOGIC;
34 clk,reset : in  STD_LOGIC;
35 underflow_sr1,underflow_sr2,underflow_sr3,underflow_sr4,underflow_sr5,underflow_sr6,
36 underflow_sr7,underflow_sr8,underflow_M1,underflow_M2,underflow_M3,underflow_M4,underflow_M5,
37 underflow_M6,underflow_M7,underflow_M8 : out std_logic;
38 overflow_sr1,overflow_sr2,overflow_sr3,overflow_sr4,overflow_sr5,overflow_sr6,overflow_sr7,
39 overflow_sr8,overflow_M1,overflow_M2,overflow_M3,overflow_M4,overflow_M5,overflow_M6,
40 overflow_M7,overflow_M8 : out std_logic;
41 invalid_op_sr1,invalid_op_sr2,invalid_op_sr3,invalid_op_sr4,invalid_op_sr5,invalid_op_sr6,
42 invalid_op_sr7,invalid_op_sr8,invalid_op_M1,invalid_op_M2,invalid_op_M3,invalid_op_M4,
43 invalid_op_M5,invalid_op_M6,invalid_op_M7,invalid_op_M8 : out std_logic;
44 rdy_M5,rdy_M6,rdy_M7,rdy_M8 : out std_logic;
45 m : in  STD_LOGIC_VECTOR (19 downto 0);
46 ocupado : out  STD_LOGIC;
47 clk_driver: IN std_logic;
48 ram_enable_driver : IN std_logic;
49 write_enable_driver : IN std_logic;
50 input_data_driver : IN std_logic_vector(31 downto 0);
51 ram_output_driver : OUT std_logic_vector(31 downto 0);
52 address_driver : IN std_logic_vector(3 downto 0);
53 bloque_ram_selection : IN std_logic_vector(3 downto 0);
54 banderas_11,banderas_22 : out std_logic_vector(15 downto 0);--señal de prueba
55 j0 : in  STD_LOGIC_VECTOR (31 downto 0);
56 j1 : in  STD_LOGIC_VECTOR (31 downto 0);
57 j2 : in  STD_LOGIC_VECTOR (31 downto 0);
58 j3 : in  STD_LOGIC_VECTOR (31 downto 0);
59 j4 : in  STD_LOGIC_VECTOR (31 downto 0);
60 j5 : in  STD_LOGIC_VECTOR (31 downto 0);
61 j6 : in  STD_LOGIC_VECTOR (31 downto 0);
62 j0_o : out  STD_LOGIC_VECTOR (31 downto 0);
63 j1_o : out  STD_LOGIC_VECTOR (31 downto 0);
64 j2_o : out  STD_LOGIC_VECTOR (31 downto 0);
65 j3_o : out  STD_LOGIC_VECTOR (31 downto 0);
66 j4_o : out  STD_LOGIC_VECTOR (31 downto 0);
67 j5_o : out  STD_LOGIC_VECTOR (31 downto 0);
68 j6_o : out  STD_LOGIC_VECTOR (31 downto 0));
69
70 end principal;
71
72 architecture Behavioral of principal is
73
74 COMPONENT async_trap_and_reset3
75     GENERIC(retardo : integer range 0 to 7 :=4);
76         PORT(
77                 async_sig : IN std_logic;
78                 outclk : IN std_logic;
79                 auto_reset : IN std_logic;
80                 reset : IN std_logic;
81                 out_sync_sig : OUT std_logic
82                 );
83         END COMPONENT;
84
```

```
85  COMPONENT registro
86          PORT(
87                  clk : IN std_logic;
88                  reset : IN std_logic;
89                  enable : IN std_logic;
90                  entrada : IN std_logic_vector(31 downto 0);
91                  salida : OUT std_logic_vector(31 downto 0)
92                  );
93          END COMPONENT;
94
95  COMPONENT Seccion_SR1
96          PORT(
97                  clk : IN std_logic;
98                  reset : IN std_logic;
99                  mux7_1_a : IN std_logic_vector(31 downto 0);
100                 mux7_1_b : IN std_logic_vector(31 downto 0);
101                 mux7_1_c : IN std_logic_vector(31 downto 0);
102                 mux7_1_d : IN std_logic_vector(31 downto 0);
103                 mux7_1_e : IN std_logic_vector(31 downto 0);
104                 mux7_1_f : IN std_logic_vector(31 downto 0);
105                 mux7_1_g : IN std_logic_vector(31 downto 0);
106                 mux7_2_a : IN std_logic_vector(31 downto 0);
107                 mux7_2_b : IN std_logic_vector(31 downto 0);
108                 mux7_2_c : IN std_logic_vector(31 downto 0);
109                 mux7_2_d : IN std_logic_vector(31 downto 0);
110                 mux7_2_e : IN std_logic_vector(31 downto 0);
111                 mux7_2_f : IN std_logic_vector(31 downto 0);
112                 mux7_2_g : IN std_logic_vector(31 downto 0);
113                 salida_m2 : IN std_logic_vector(31 downto 0);
114                 z01 : IN std_logic_vector(31 downto 0);
115                 z05 : IN std_logic_vector(31 downto 0);
116                 j5 : IN std_logic_vector(31 downto 0);
117                 enable_registers_4 : IN std_logic;
118                 enable_registers_12 : IN std_logic;
119                 enable_registers_16 : IN std_logic;
120                 enable_registers_29 : IN std_logic;
121                 enable_registers_37 : IN std_logic;
122                 enable_mux7_1 : IN std_logic_vector(2 downto 0);
123                 enable_mux7_2 : IN std_logic_vector(2 downto 0);
124                 enable_mux2_1 : IN std_logic;
125                 enable_mux2_2 : IN std_logic;
126                 enable_mux2_3 : IN std_logic;
127                 enable_mux2_4 : IN std_logic;
128                 operation_sr1 : IN std_logic_vector(5 downto 0);
129                 operation_nd_sr1 : IN std_logic;
130                 sclr_sr1 : IN std_logic;
131                 ce_sr1 : IN std_logic;
132                 sr29 : OUT std_logic_vector(31 downto 0);
133                 sr37 : OUT std_logic_vector(31 downto 0);
134                 sr4 : OUT std_logic_vector(31 downto 0);
135                 sr12 : OUT std_logic_vector(31 downto 0);
136                 sr16 : OUT std_logic_vector(31 downto 0);
137                 operation_rfd_sr1 : OUT std_logic;
```

```
138                     underflow_sr1 : OUT std_logic;
139                     overflow_sr1 : OUT std_logic;
140                     invalid_op_sr1 : OUT std_logic;
141                     rdy_sr1 : OUT std_logic
142                     );
143             END COMPONENT;
144
145     COMPONENT Seccion_SR2
146             PORT(
147                     clk : IN std_logic;
148                     reset : IN std_logic;
149                     mux8_1_a : IN std_logic_vector(31 downto 0);
150                     mux8_1_b : IN std_logic_vector(31 downto 0);
151                     mux8_1_c : IN std_logic_vector(31 downto 0);
152                     mux8_1_d : IN std_logic_vector(31 downto 0);
153                     mux8_1_e : IN std_logic_vector(31 downto 0);
154                     mux8_1_f : IN std_logic_vector(31 downto 0);
155                     mux8_1_g : IN std_logic_vector(31 downto 0);
156                     mux8_1_h : IN std_logic_vector(31 downto 0);
157                     mux7_3_a : IN std_logic_vector(31 downto 0);
158                     mux7_3_b : IN std_logic_vector(31 downto 0);
159                     mux7_3_c : IN std_logic_vector(31 downto 0);
160                     mux7_3_d : IN std_logic_vector(31 downto 0);
161                     mux7_3_e : IN std_logic_vector(31 downto 0);
162                     mux7_3_f : IN std_logic_vector(31 downto 0);
163                     mux7_3_g : IN std_logic_vector(31 downto 0);
164                     salida_m4 : IN std_logic_vector(31 downto 0);
165                     salida_m2 : IN std_logic_vector(31 downto 0);
166                     z01_1 : IN std_logic_vector(31 downto 0);
167                     z05_1 : IN std_logic_vector(31 downto 0);
168                     j4 : IN std_logic_vector(31 downto 0);
169                     enable_registers_5  : IN std_logic;
170                     enable_registers_13 : IN std_logic;
171                     enable_registers_17 : IN std_logic;
172                     enable_registers_30 : IN std_logic;
173                     enable_registers_38 : IN std_logic;
174                     enable_registers_46 : IN std_logic;
175                     enable_mux8_1 : IN std_logic_vector(2 downto 0);
176                     enable_mux7_3 : IN std_logic_vector(2 downto 0);
177                     enable_mux2_5 : IN std_logic;
178                     enable_mux2_6 : IN std_logic;
179                     enable_mux2_7 : IN std_logic;
180                     enable_mux2_8 : IN std_logic;
181                     enable_mux2_9 : IN std_logic;
182                     operation_sr2 : IN std_logic_vector(5 downto 0);
183                     operation_nd_sr2 : IN std_logic;
184                     sclr_sr2 : IN std_logic;
185                     ce_sr2 : IN std_logic;
186                     sr30 : OUT std_logic_vector(31 downto 0);
187                     sr38 : OUT std_logic_vector(31 downto 0);
188                     sr46 : OUT std_logic_vector(31 downto 0);
189                     sr5 : OUT std_logic_vector(31 downto 0);
190                     sr13 : OUT std_logic_vector(31 downto 0);
```

```
191                 sr17 : OUT std_logic_vector(31 downto 0);
192                 operation_rfd_sr2 : OUT std_logic;
193                 underflow_sr2 : OUT std_logic;
194                 overflow_sr2 : OUT std_logic;
195                 invalid_op_sr2 : OUT std_logic;
196                 rdy_sr2 : OUT std_logic
197                 );
198         END COMPONENT;
199
200 COMPONENT Seccion_SR3
201         PORT(
202                 clk : IN std_logic;
203                 reset : IN std_logic;
204                 mux11_1_a : IN std_logic_vector(31 downto 0);
205                 mux11_1_b : IN std_logic_vector(31 downto 0);
206                 mux11_1_c : IN std_logic_vector(31 downto 0);
207                 mux11_1_d : IN std_logic_vector(31 downto 0);
208                 mux11_1_e : IN std_logic_vector(31 downto 0);
209                 mux11_1_f : IN std_logic_vector(31 downto 0);
210                 mux11_1_g : IN std_logic_vector(31 downto 0);
211                 mux11_1_h : IN std_logic_vector(31 downto 0);
212                 mux11_1_i : IN std_logic_vector(31 downto 0);
213                 mux11_1_j : IN std_logic_vector(31 downto 0);
214                 mux11_1_k : IN std_logic_vector(31 downto 0);
215                 mux11_2_a : IN std_logic_vector(31 downto 0);
216                 mux11_2_b : IN std_logic_vector(31 downto 0);
217                 mux11_2_c : IN std_logic_vector(31 downto 0);
218                 mux11_2_d : IN std_logic_vector(31 downto 0);
219                 mux11_2_e : IN std_logic_vector(31 downto 0);
220                 mux11_2_f : IN std_logic_vector(31 downto 0);
221                 mux11_2_g : IN std_logic_vector(31 downto 0);
222                 mux11_2_h : IN std_logic_vector(31 downto 0);
223                 mux11_2_i : IN std_logic_vector(31 downto 0);
224                 mux11_2_j : IN std_logic_vector(31 downto 0);
225                 mux11_2_k : IN std_logic_vector(31 downto 0);
226                 salida_m1 : IN std_logic_vector(31 downto 0);
227                 z02 : IN std_logic_vector(31 downto 0);
228                 z06 : IN std_logic_vector(31 downto 0);
229                 j3 : IN std_logic_vector(31 downto 0);
230                 enable_registers_6  : IN std_logic;
231                 enable_registers_14  : IN std_logic;
232                 enable_registers_18  : IN std_logic;
233                 enable_registers_31  : IN std_logic;
234                 enable_registers_39  : IN std_logic;
235                 enable_registers_47  : IN std_logic;
236                 enable_registers_45  : IN std_logic;
237                 enable_mux11_1 : IN std_logic_vector(3 downto 0);
238                 enable_mux11_2 : IN std_logic_vector(3 downto 0);
239                 enable_mux2_10 : IN std_logic;
240                 enable_mux2_11 : IN std_logic;
241                 enable_mux2_12 : IN std_logic;
242                 enable_mux2_13 : IN std_logic;
243                 enable_mux2_14 : IN std_logic;
```

```
244                 operation_sr3 : IN std_logic_vector(5 downto 0);
245                 operation_nd_sr3 : IN std_logic;
246                 sclr_sr3 : IN std_logic;
247                 ce_sr3 : IN std_logic;
248                 sr31 : OUT std_logic_vector(31 downto 0);
249                 sr39 : OUT std_logic_vector(31 downto 0);
250                 sr47 : OUT std_logic_vector(31 downto 0);
251                 sr6 : OUT std_logic_vector(31 downto 0);
252                 sr14 : OUT std_logic_vector(31 downto 0);
253                 sr18 : OUT std_logic_vector(31 downto 0);
254                 sr45 : OUT std_logic_vector(31 downto 0);
255                 operation_rfd_sr3 : OUT std_logic;
256                 underflow_sr3 : OUT std_logic;
257                 overflow_sr3 : OUT std_logic;
258                 invalid_op_sr3 : OUT std_logic;
259                 rdy_sr3 : OUT std_logic
260                 );
261         END COMPONENT;
262
263 COMPONENT Seccion_SR4
264         PORT(
265                 clk : IN std_logic;
266                 reset : IN std_logic;
267                 mux9_1_a : IN std_logic_vector(31 downto 0);
268                 mux9_1_b : IN std_logic_vector(31 downto 0);
269                 mux9_1_c : IN std_logic_vector(31 downto 0);
270                 mux9_1_d : IN std_logic_vector(31 downto 0);
271                 mux9_1_e : IN std_logic_vector(31 downto 0);
272                 mux9_1_f : IN std_logic_vector(31 downto 0);
273                 mux9_1_g : IN std_logic_vector(31 downto 0);
274                 mux9_1_h : IN std_logic_vector(31 downto 0);
275                 mux9_1_i : IN std_logic_vector(31 downto 0);
276                 mux9_2_a : IN std_logic_vector(31 downto 0);
277                 mux9_2_b : IN std_logic_vector(31 downto 0);
278                 mux9_2_c : IN std_logic_vector(31 downto 0);
279                 mux9_2_d : IN std_logic_vector(31 downto 0);
280                 mux9_2_e : IN std_logic_vector(31 downto 0);
281                 mux9_2_f : IN std_logic_vector(31 downto 0);
282                 mux9_2_g : IN std_logic_vector(31 downto 0);
283                 mux9_2_h : IN std_logic_vector(31 downto 0);
284                 mux9_2_i : IN std_logic_vector(31 downto 0);
285                 salida_m2 : IN std_logic_vector(31 downto 0);
286                 salida_m3 : IN std_logic_vector(31 downto 0);
287                 z02_1 : IN std_logic_vector(31 downto 0);
288                 z06_1 : IN std_logic_vector(31 downto 0);
289                 j2 : IN std_logic_vector(31 downto 0);
290                 enable_registers_7  : IN std_logic;
291                 enable_registers_15  : IN std_logic;
292                 enable_registers_19  : IN std_logic;
293                 enable_registers_32  : IN std_logic;
294                 enable_registers_40  : IN std_logic;
295                 enable_registers_48  : IN std_logic;
296                 enable_mux9_1 : IN std_logic_vector(3 downto 0);
```

```
297                     enable_mux9_2 : IN std_logic_vector(3 downto 0);
298                     enable_mux2_15 : IN std_logic;
299                     enable_mux2_16 : IN std_logic;
300                     enable_mux2_17 : IN std_logic;
301                     enable_mux2_18 : IN std_logic;
302                     enable_mux2_19 : IN std_logic;
303                     operation_sr4 : IN std_logic_vector(5 downto 0);
304                     operation_nd_sr4 : IN std_logic;
305                     sclr_sr4 : IN std_logic;
306                     ce_sr4 : IN std_logic;
307                     sr40 : OUT std_logic_vector(31 downto 0);
308                     sr48 : OUT std_logic_vector(31 downto 0);
309                     sr7 : OUT std_logic_vector(31 downto 0);
310                     sr15 : OUT std_logic_vector(31 downto 0);
311                     sr19 : OUT std_logic_vector(31 downto 0);
312                     sr32 : OUT std_logic_vector(31 downto 0);
313                     operation_rfd_sr4 : OUT std_logic;
314                     underflow_sr4 : OUT std_logic;
315                     overflow_sr4 : OUT std_logic;
316                     invalid_op_sr4 : OUT std_logic;
317                     rdy_sr4 : OUT std_logic
318                     );
319             END COMPONENT;
320
321     COMPONENT Seccion_SR5
322             PORT(
323                     clk : IN std_logic;
324                     reset : IN std_logic;
325                     mux9_3_a : IN std_logic_vector(31 downto 0);
326                     mux9_3_b : IN std_logic_vector(31 downto 0);
327                     mux9_3_c : IN std_logic_vector(31 downto 0);
328                     mux9_3_d : IN std_logic_vector(31 downto 0);
329                     mux9_3_e : IN std_logic_vector(31 downto 0);
330                     mux9_3_f : IN std_logic_vector(31 downto 0);
331                     mux9_3_g : IN std_logic_vector(31 downto 0);
332                     mux9_3_h : IN std_logic_vector(31 downto 0);
333                     mux7_4_a : IN std_logic_vector(31 downto 0);
334                     mux7_4_b : IN std_logic_vector(31 downto 0);
335                     mux7_4_c : IN std_logic_vector(31 downto 0);
336                     mux7_4_d : IN std_logic_vector(31 downto 0);
337                     mux7_4_e : IN std_logic_vector(31 downto 0);
338                     mux7_4_f : IN std_logic_vector(31 downto 0);
339                     mux7_4_g : IN std_logic_vector(31 downto 0);
340                     salida_m1 : IN std_logic_vector(31 downto 0);
341                     salida_m6 : IN std_logic_vector(31 downto 0);
342                     z00 : IN std_logic_vector(31 downto 0);
343                     z03 : IN std_logic_vector(31 downto 0);
344                     j6 : IN std_logic_vector(31 downto 0);
345                     j0 : IN std_logic_vector(31 downto 0);
346                     enable_registers_2  : IN std_logic;
347                     enable_registers_8  : IN std_logic;
348                     enable_registers_21  : IN std_logic;
349                     enable_registers_28  : IN std_logic;
```

```
350                     enable_registers_33  : IN std_logic;
351                     enable_registers_41  : IN std_logic;
352                     enable_registers_52  : IN std_logic;
353                     enable_mux9_3 : IN std_logic_vector(3 downto 0);
354                     enable_mux7_4 : IN std_logic_vector(2 downto 0);
355                     enable_mux2_20 : IN std_logic;
356                     enable_mux2_21 : IN std_logic;
357                     enable_mux2_22 : IN std_logic;
358                     enable_mux2_23 : IN std_logic;
359                     enable_mux2_24 : IN std_logic;
360                     enable_mux2_25 : IN std_logic;
361                     operation_sr5 : IN std_logic_vector(5 downto 0);
362                     operation_nd_sr5 : IN std_logic;
363                     sclr_sr5 : IN std_logic;
364                     ce_sr5 : IN std_logic;
365                     sr33 : OUT std_logic_vector(31 downto 0);
366                     sr41 : OUT std_logic_vector(31 downto 0);
367                     sr52 : OUT std_logic_vector(31 downto 0);
368                     sr2 : OUT std_logic_vector(31 downto 0);
369                     sr8 : OUT std_logic_vector(31 downto 0);
370                     sr21 : OUT std_logic_vector(31 downto 0);
371                     sr28_0 : out std_logic_vector(31 downto 0);
372                     operation_rfd_sr5 : OUT std_logic;
373                     underflow_sr5 : OUT std_logic;
374                     overflow_sr5 : OUT std_logic;
375                     invalid_op_sr5 : OUT std_logic;
376                     rdy_sr5 : OUT std_logic
377                     );
378             END COMPONENT;
379
380     COMPONENT Seccion_SR6
381             PORT(
382                     clk : IN std_logic;
383                     reset : IN std_logic;
384                     mux8_2_a : IN std_logic_vector(31 downto 0);
385                     mux8_2_b : IN std_logic_vector(31 downto 0);
386                     mux8_2_c : IN std_logic_vector(31 downto 0);
387                     mux8_2_d : IN std_logic_vector(31 downto 0);
388                     mux8_2_e : IN std_logic_vector(31 downto 0);
389                     mux8_2_f : IN std_logic_vector(31 downto 0);
390                     mux8_2_g : IN std_logic_vector(31 downto 0);
391                     mux8_2_h : IN std_logic_vector(31 downto 0);
392                     mux7_5_a : IN std_logic_vector(31 downto 0);
393                     mux7_5_b : IN std_logic_vector(31 downto 0);
394                     mux7_5_c : IN std_logic_vector(31 downto 0);
395                     mux7_5_d : IN std_logic_vector(31 downto 0);
396                     mux7_5_e : IN std_logic_vector(31 downto 0);
397                     mux7_5_f : IN std_logic_vector(31 downto 0);
398                     mux7_5_g : IN std_logic_vector(31 downto 0);
399                     salida_m2 : IN std_logic_vector(31 downto 0);
400                     salida_m7 : IN std_logic_vector(31 downto 0);
401                     z00_1 : IN std_logic_vector(31 downto 0);
402                     z03_1 : IN std_logic_vector(31 downto 0);
```

```
403                     j1 : IN std_logic_vector(31 downto 0);
404                     enable_registers_3  : IN std_logic;
405                     enable_registers_9  : IN std_logic;
406                     enable_registers_20  : IN std_logic;
407                     enable_registers_34  : IN std_logic;
408                     enable_registers_42  : IN std_logic;
409                     enable_registers_53  : IN std_logic;
410                     enable_mux8_2 : IN std_logic_vector(2 downto 0);
411                     enable_mux7_5 : IN std_logic_vector(2 downto 0);
412                     enable_mux2_26 : IN std_logic;
413                     enable_mux2_27 : IN std_logic;
414                     enable_mux2_28 : IN std_logic;
415                     enable_mux2_29 : IN std_logic;
416                     enable_mux2_30 : IN std_logic;
417                     operation_sr6 : IN std_logic_vector(5 downto 0);
418                     operation_nd_sr6 : IN std_logic;
419                     sclr_sr6 : IN std_logic;
420                     ce_sr6 : IN std_logic;
421                     sr34 : OUT std_logic_vector(31 downto 0);
422                     sr42 : OUT std_logic_vector(31 downto 0);
423                     sr53 : OUT std_logic_vector(31 downto 0);
424                     sr3 : OUT std_logic_vector(31 downto 0);
425                     sr9 : OUT std_logic_vector(31 downto 0);
426                     sr20 : OUT std_logic_vector(31 downto 0);
427                     operation_rfd_sr6 : OUT std_logic;
428                     underflow_sr6 : OUT std_logic;
429                     overflow_sr6 : OUT std_logic;
430                     invalid_op_sr6 : OUT std_logic;
431                     rdy_sr6 : OUT std_logic
432                     );
433             END COMPONENT;
434
435     COMPONENT Seccion_SR7
436             PORT(
437                     clk : IN std_logic;
438                     reset : IN std_logic;
439                     mux8_3_a : IN std_logic_vector(31 downto 0);
440                     mux8_3_b : IN std_logic_vector(31 downto 0);
441                     mux8_3_c : IN std_logic_vector(31 downto 0);
442                     mux8_3_d : IN std_logic_vector(31 downto 0);
443                     mux8_3_e : IN std_logic_vector(31 downto 0);
444                     mux8_3_f : IN std_logic_vector(31 downto 0);
445                     mux8_3_g : IN std_logic_vector(31 downto 0);
446                     mux8_3_h : IN std_logic_vector(31 downto 0);
447                     mux8_4_a : IN std_logic_vector(31 downto 0);
448                     mux8_4_b : IN std_logic_vector(31 downto 0);
449                     mux8_4_c : IN std_logic_vector(31 downto 0);
450                     mux8_4_d : IN std_logic_vector(31 downto 0);
451                     mux8_4_e : IN std_logic_vector(31 downto 0);
452                     mux8_4_f : IN std_logic_vector(31 downto 0);
453                     mux8_4_g : IN std_logic_vector(31 downto 0);
454                     mux8_4_h : IN std_logic_vector(31 downto 0);
455                     salida_m5 : IN std_logic_vector(31 downto 0);
```

```
456                 z04 : IN std_logic_vector(31 downto 0);
457                 enable_registers_10  : IN std_logic;
458                 enable_registers_35  : IN std_logic;
459                 enable_registers_43  : IN std_logic;
460                 enable_registers_51  : IN std_logic;
461                 enable_mux8_3 : IN std_logic_vector(2 downto 0);
462                 enable_mux8_4 : IN std_logic_vector(2 downto 0);
463                 enable_mux2_31 : IN std_logic;
464                 enable_mux2_32 : IN std_logic;
465                 operation_sr7 : IN std_logic_vector(5 downto 0);
466                 operation_nd_sr7 : IN std_logic;
467                 sclr_sr7 : IN std_logic;
468                 ce_sr7 : IN std_logic;
469                 sr35 : OUT std_logic_vector(31 downto 0);
470                 sr43 : OUT std_logic_vector(31 downto 0);
471                 sr51 : OUT std_logic_vector(31 downto 0);
472                 sr10 : OUT std_logic_vector(31 downto 0);
473                 operation_rfd_sr7 : OUT std_logic;
474                 underflow_sr7 : OUT std_logic;
475                 overflow_sr7 : OUT std_logic;
476                 invalid_op_sr7 : OUT std_logic;
477                 rdy_sr7 : OUT std_logic
478                 );
479         END COMPONENT;
480
481 COMPONENT Seccion_SR8
482         PORT(
483                 clk : IN std_logic;
484                 reset : IN std_logic;
485                 mux7_6_a : IN std_logic_vector(31 downto 0);
486                 mux7_6_b : IN std_logic_vector(31 downto 0);
487                 mux7_6_c : IN std_logic_vector(31 downto 0);
488                 mux7_6_d : IN std_logic_vector(31 downto 0);
489                 mux7_6_e : IN std_logic_vector(31 downto 0);
490                 mux7_6_f : IN std_logic_vector(31 downto 0);
491                 mux7_6_g : IN std_logic_vector(31 downto 0);
492                 mux7_7_a : IN std_logic_vector(31 downto 0);
493                 mux7_7_b : IN std_logic_vector(31 downto 0);
494                 mux7_7_c : IN std_logic_vector(31 downto 0);
495                 mux7_7_d : IN std_logic_vector(31 downto 0);
496                 mux7_7_e : IN std_logic_vector(31 downto 0);
497                 mux7_7_f : IN std_logic_vector(31 downto 0);
498                 mux7_7_g : IN std_logic_vector(31 downto 0);
499                 z04_1 : IN std_logic_vector(31 downto 0);
500                 enable_registers_11  : IN std_logic;
501                 enable_registers_36  : IN std_logic;
502                 enable_registers_44  : IN std_logic;
503                 enable_mux7_6 : IN std_logic_vector(2 downto 0);
504                 enable_mux7_7 : IN std_logic_vector(2 downto 0);
505                 enable_mux2_33 : IN std_logic;
506                 operation_sr8 : IN std_logic_vector(5 downto 0);
507                 operation_nd_sr8 : IN std_logic;
508                 sclr_sr8 : IN std_logic;
```

```
509                     ce_sr8 : IN std_logic;
510                     sr36 : OUT std_logic_vector(31 downto 0);
511                     sr44 : OUT std_logic_vector(31 downto 0);
512                     sr11 : OUT std_logic_vector(31 downto 0);
513                     operation_rfd_sr8 : OUT std_logic;
514                     underflow_sr8 : OUT std_logic;
515                     overflow_sr8 : OUT std_logic;
516                     invalid_op_sr8 : OUT std_logic;
517                     rdy_sr8 : OUT std_logic
518                     );
519             END COMPONENT;
520
521     COMPONENT Seccion_M1
522             PORT(
523                     clk : IN std_logic;
524                     reset : IN std_logic;
525                     mux4_1_a : IN std_logic_vector(31 downto 0);
526                     mux4_1_b : IN std_logic_vector(31 downto 0);
527                     mux4_1_c : IN std_logic_vector(31 downto 0);
528                     mux4_1_d : IN std_logic_vector(31 downto 0);
529                     mux5_1_a : IN std_logic_vector(31 downto 0);
530                     mux5_1_b : IN std_logic_vector(31 downto 0);
531                     mux5_1_c : IN std_logic_vector(31 downto 0);
532                     mux5_1_d : IN std_logic_vector(31 downto 0);
533                     mux5_1_e : IN std_logic_vector(31 downto 0);
534                     enable_registers_55  : IN std_logic;
535                     enable_mux4_1 : IN std_logic_vector(1 downto 0);
536                     enable_mux5_1 : IN std_logic_vector(2 downto 0);
537                     operation_nd_M1 : IN std_logic;
538                     sclr_M1 : IN std_logic;
539                     ce_M1 : IN std_logic;
540                     result_M1 : OUT std_logic_vector(31 downto 0);
541                     sr55 : OUT std_logic_vector(31 downto 0);
542                     operation_rfd_M1 : OUT std_logic;
543                     underflow_M1 : OUT std_logic;
544                     overflow_M1 : OUT std_logic;
545                     invalid_op_M1 : OUT std_logic;
546                     rdy_M1 : OUT std_logic
547                     );
548             END COMPONENT;
549
550     COMPONENT Seccion_M2
551             PORT(
552                     clk : IN std_logic;
553                     reset : IN std_logic;
554                     mux4_2_a : IN std_logic_vector(31 downto 0);
555                     mux4_2_b : IN std_logic_vector(31 downto 0);
556                     mux4_2_c : IN std_logic_vector(31 downto 0);
557                     mux4_2_d : IN std_logic_vector(31 downto 0);
558                     mux5_2_a : IN std_logic_vector(31 downto 0);
559                     mux5_2_b : IN std_logic_vector(31 downto 0);
560                     mux5_2_c : IN std_logic_vector(31 downto 0);
561                     mux5_2_d : IN std_logic_vector(31 downto 0);
```

```
562                     mux5_2_e : IN std_logic_vector(31 downto 0);
563                     enable_registers_58  : IN std_logic;
564                     enable_mux4_2 : IN std_logic_vector(1 downto 0);
565                     enable_mux5_2 : IN std_logic_vector(2 downto 0);
566                     operation_nd_M2 : IN std_logic;
567                     sclr_M2 : IN std_logic;
568                     ce_M2 : IN std_logic;
569                     sr58 : OUT std_logic_vector(31 downto 0);
570                     operation_rfd_M2 : OUT std_logic;
571                     result_M2 : OUT std_logic_vector(31 downto 0);
572                     underflow_M2 : OUT std_logic;
573                     overflow_M2 : OUT std_logic;
574                     invalid_op_M2 : OUT std_logic;
575                     rdy_M2 : OUT std_logic
576                     );
577             END COMPONENT;
578
579     COMPONENT Seccion_M3
580             PORT(
581                     clk : IN std_logic;
582                     reset : IN std_logic;
583                     mux4_3_a : IN std_logic_vector(31 downto 0);
584                     mux4_3_b : IN std_logic_vector(31 downto 0);
585                     mux4_3_c : IN std_logic_vector(31 downto 0);
586                     mux4_3_d : IN std_logic_vector(31 downto 0);
587                     mux3_1_a : IN std_logic_vector(31 downto 0);
588                     mux3_1_b : IN std_logic_vector(31 downto 0);
589                     mux3_1_c : IN std_logic_vector(31 downto 0);
590                     enable_registers_49  : IN std_logic;
591                     enable_mux4_3 : IN std_logic_vector(1 downto 0);
592                     enable_mux3_1 : IN std_logic_vector(1 downto 0);
593                     operation_nd_M3 : IN std_logic;
594                     sclr_M3 : IN std_logic;
595                     ce_M3 : IN std_logic;
596                     sr49 : OUT std_logic_vector(31 downto 0);
597                     operation_rfd_M3 : OUT std_logic;
598                     result_M3 : OUT std_logic_vector(31 downto 0);
599                     underflow_M3 : OUT std_logic;
600                     overflow_M3 : OUT std_logic;
601                     invalid_op_M3 : OUT std_logic;
602                     rdy_M3 : OUT std_logic
603                     );
604             END COMPONENT;
605
606     COMPONENT Seccion_M4
607             PORT(
608                     clk : IN std_logic;
609                     reset : IN std_logic;
610                     mux4_4_a : IN std_logic_vector(31 downto 0);
611                     mux4_4_b : IN std_logic_vector(31 downto 0);
612                     mux4_4_c : IN std_logic_vector(31 downto 0);
613                     mux4_4_d : IN std_logic_vector(31 downto 0);
614                     mux3_2_a : IN std_logic_vector(31 downto 0);
```

```
615                     mux3_2_b : IN std_logic_vector(31 downto 0);
616                     mux3_2_c : IN std_logic_vector(31 downto 0);
617                     enable_registers_50  : IN std_logic;
618                     enable_mux4_4 : IN std_logic_vector(1 downto 0);
619                     enable_mux3_2 : IN std_logic_vector(1 downto 0);
620                     operation_nd_M4 : IN std_logic;
621                     sclr_M4 : IN std_logic;
622                     ce_M4 : IN std_logic;
623                     sr50 : OUT std_logic_vector(31 downto 0);
624                     operation_rfd_M4 : OUT std_logic;
625                     result_M4 : OUT std_logic_vector(31 downto 0);
626                     underflow_M4 : OUT std_logic;
627                     overflow_M4 : OUT std_logic;
628                     invalid_op_M4 : OUT std_logic;
629                     rdy_M4 : OUT std_logic
630                     );
631             END COMPONENT;
632
633     COMPONENT Seccion_M5
634             PORT(
635                     clk : IN std_logic;
636                     reset : IN std_logic;
637                     mux3_3_a : IN std_logic_vector(31 downto 0);
638                     mux3_3_b : IN std_logic_vector(31 downto 0);
639                     mux3_3_c : IN std_logic_vector(31 downto 0);
640                     mux3_4_a : IN std_logic_vector(31 downto 0);
641                     mux3_4_b : IN std_logic_vector(31 downto 0);
642                     mux3_4_c : IN std_logic_vector(31 downto 0);
643                     enable_registers_59  : IN std_logic;
644                     enable_mux3_3 : IN std_logic_vector(1 downto 0);
645                     enable_mux3_4 : IN std_logic_vector(1 downto 0);
646                     operation_nd_M5 : IN std_logic;
647                     sclr_M5 : IN std_logic;
648                     ce_M5 : IN std_logic;
649                     sr59 : OUT std_logic_vector(31 downto 0);
650                     operation_rfd_M5 : OUT std_logic;
651                     result_M5 : OUT std_logic_vector(31 downto 0);
652                     underflow_M5 : OUT std_logic;
653                     overflow_M5 : OUT std_logic;
654                     invalid_op_M5 : OUT std_logic;
655                     rdy_M5 : OUT std_logic
656                     );
657             END COMPONENT;
658
659     COMPONENT Seccion_M6
660             PORT(
661                     clk : IN std_logic;
662                     reset : IN std_logic;
663                     mux4_5_a : IN std_logic_vector(31 downto 0);
664                     mux4_5_b : IN std_logic_vector(31 downto 0);
665                     mux4_5_c : IN std_logic_vector(31 downto 0);
666                     mux4_5_d : IN std_logic_vector(31 downto 0);
667                     mux3_5_a : IN std_logic_vector(31 downto 0);
```

```
668                 mux3_5_b : IN std_logic_vector(31 downto 0);
669                 mux3_5_c : IN std_logic_vector(31 downto 0);
670                 enable_registers_56  : IN std_logic;
671                 enable_mux4_5 : IN std_logic_vector(1 downto 0);
672                 enable_mux3_5 : IN std_logic_vector(1 downto 0);
673                 operation_nd_M6 : IN std_logic;
674                 sclr_M6 : IN std_logic;
675                 ce_M6 : IN std_logic;
676                 sr56 : OUT std_logic_vector(31 downto 0);
677                 operation_rfd_M6 : OUT std_logic;
678                 result_M6 : OUT std_logic_vector(31 downto 0);
679                 underflow_M6 : OUT std_logic;
680                 overflow_M6 : OUT std_logic;
681                 invalid_op_M6 : OUT std_logic;
682                 rdy_M6 : OUT std_logic
683                 );
684         END COMPONENT;
685
686 COMPONENT Seccion_M7
687         PORT(
688                 clk : IN std_logic;
689                 reset : IN std_logic;
690                 mux3_6_a : IN std_logic_vector(31 downto 0);
691                 mux3_6_b : IN std_logic_vector(31 downto 0);
692                 mux3_6_c : IN std_logic_vector(31 downto 0);
693                 mux2_34_a : IN std_logic_vector(31 downto 0);
694                 mux2_34_b : IN std_logic_vector(31 downto 0);
695                 enable_registers_57  : IN std_logic;
696                 enable_mux3_6 : IN std_logic_vector(1 downto 0);
697                 enable_mux2_34 : IN std_logic;
698                 operation_nd_M7 : IN std_logic;
699                 sclr_M7 : IN std_logic;
700                 ce_M7 : IN std_logic;
701                 sr57 : OUT std_logic_vector(31 downto 0);
702                 operation_rfd_M7 : OUT std_logic;
703                 result_M7 : OUT std_logic_vector(31 downto 0);
704                 underflow_M7 : OUT std_logic;
705                 overflow_M7 : OUT std_logic;
706                 invalid_op_M7 : OUT std_logic;
707                 rdy_M7 : OUT std_logic
708                 );
709         END COMPONENT;
710
711 COMPONENT Seccion_M8
712         PORT(
713                 clk : IN std_logic;
714                 reset : IN std_logic;
715                 mux4_6_a : IN std_logic_vector(31 downto 0);
716                 mux4_6_b : IN std_logic_vector(31 downto 0);
717                 mux4_6_c : IN std_logic_vector(31 downto 0);
718                 mux4_6_d : IN std_logic_vector(31 downto 0);
719                 mux3_7_a : IN std_logic_vector(31 downto 0);
720                 mux3_7_b : IN std_logic_vector(31 downto 0);
```

```
721                 mux3_7_c : IN std_logic_vector(31 downto 0);
722                 enable_registers_54  : IN std_logic;
723                 enable_mux4_6 : IN std_logic_vector(1 downto 0);
724                 enable_mux3_7 : IN std_logic_vector(1 downto 0);
725                 operation_nd_M8 : IN std_logic;
726                 sclr_M8 : IN std_logic;
727                 ce_M8 : IN std_logic;
728                 sr54 : OUT std_logic_vector(31 downto 0);
729                 operation_rfd_M8 : OUT std_logic;
730                 underflow_M8 : OUT std_logic;
731                 overflow_M8 : OUT std_logic;
732                 invalid_op_M8 : OUT std_logic;
733                 rdy_M8 : OUT std_logic
734                 );
735         END COMPONENT;
736
737 COMPONENT FSM_7
738         PORT(
739                 clk : IN std_logic;
740                 inicio : IN std_logic;
741                 m : IN std_logic_vector(4 downto 0);
742                 reset : IN std_logic;
743                 banderas1,banderas2 : IN std_logic_vector(15 downto 0);
744          estado_sig : out std_logic_vector(4 downto 0);
745                 senales_de_control : OUT std_logic_vector(272 downto 0)
746                 );
747         END COMPONENT;
748
749 COMPONENT memoria_prueba
750     generic(INIT_00,INIT_01,INIT_02,INIT_03,INIT_04,INIT_05,INIT_06,INIT_07,INIT_08, INIT_09,
751     INIT_0A, INIT_0B, INIT_0C, INIT_0D, INIT_0E, INIT_0F: std_logic_vector(31 downto 0);
752          ancho_dir : integer;
753          ancho_dat : integer);
754         PORT(
755                 clockA : IN std_logic;
756                 clockB : IN std_logic;
757                 ram_enableA : IN std_logic;
758                 ram_enableB : IN std_logic;
759                 write_enableA : IN std_logic;
760                 write_enableB : IN std_logic;
761                 input_dataA : IN std_logic_vector(31 downto 0);
762                 input_dataB : IN std_logic_vector(31 downto 0);
763                 addressA : IN std_logic_vector(3 downto 0);
764                 addressB : IN std_logic_vector(3 downto 0);
765                 ram_outputA : OUT std_logic_vector(31 downto 0);
766                 ram_outputB : OUT std_logic_vector(31 downto 0)
767                 );
768         END COMPONENT;
769 --señales usadas para interactuar con el driver
770 signal ram_enable_driver_00,ram_enable_driver_00_1,ram_enable_driver_01,ram_enable_driver_01_1,
771 ram_enable_driver_02,ram_enable_driver_02_1,ram_enable_driver_03,ram_enable_driver_03_1,
772 ram_enable_driver_04,ram_enable_driver_04_1,ram_enable_driver_05,ram_enable_driver_05_1,
773 ram_enable_driver_06,ram_enable_driver_06_1  : std_logic;
```

```
774 signal ram_output_driver_00,ram_output_driver_00_1,ram_output_driver_01,ram_output_driver_01_1,
775 ram_output_driver_02,ram_output_driver_02_1,ram_output_driver_03,ram_output_driver_03_1,
776 ram_output_driver_04,ram_output_driver_04_1,ram_output_driver_05,ram_output_driver_05_1,
777 ram_output_driver_06,ram_output_driver_06_1 : std_logic_vector(31 downto 0);
778
779 ---señales de prueba para el uso de las memorias
780 signal z00,z00_1,z01,z01_1,z02,z02_1,z03,z03_1,z04,z04_1,z05,z05_1,z06,z06_1,z00_o,z00_1_o,z01_o
781 ,z01_1_o,z02_o,z02_1_o,z03_o,z03_1_o,z04_o,z04_1_o,z05_o,z05_1_o,z06_o,
782 z06_1_o : STD_LOGIC_VECTOR (31 downto 0);
783 signal direccion : std_logic_vector(3 downto 0):="0000";
784 signal habilita_w,habilita1,habilita2,habilita3 : std_logic:='0';
785 ----
786 signal estado : std_logic_vector(4 downto 0);
787 signal enable_registers_2,enable_registers_3,enable_registers_4,enable_registers_5,
788 enable_registers_6,enable_registers_7,enable_registers_8,enable_registers_9,
789 enable_registers_10 : std_logic;
790 signal enable_registers_11,enable_registers_12,enable_registers_13,enable_registers_14,
791 enable_registers_15,enable_registers_16,enable_registers_17,enable_registers_18,
792 enable_registers_19,enable_registers_20 : std_logic;
793 signal enable_registers_21,enable_registers_28,enable_registers_29,
794 enable_registers_30 : std_logic;
795 signal enable_registers_31,enable_registers_32,enable_registers_33,enable_registers_34,
796 enable_registers_35,enable_registers_36,enable_registers_37,enable_registers_38,
797 enable_registers_39,enable_registers_40 : std_logic;
798 signal enable_registers_41,enable_registers_42,enable_registers_43,enable_registers_44,
799 enable_registers_45,enable_registers_46,enable_registers_47,enable_registers_48,
800 enable_registers_49,enable_registers_50 : std_logic;
801 signal enable_registers_51,enable_registers_52,enable_registers_53,enable_registers_54,
802 enable_registers_55,enable_registers_56,enable_registers_57,enable_registers_58,
803 enable_registers_59 : std_logic;
804 signal enable_mux11_1,enable_mux11_2 : std_logic_vector(3 downto 0);
805 signal enable_mux9_1,enable_mux9_2,enable_mux9_3 : std_logic_vector(3 downto 0);
806 signal enable_mux8_1,enable_mux8_2,enable_mux8_3,
807 enable_mux8_4 : std_logic_vector(2 downto 0);
808 signal enable_mux7_1,enable_mux7_2,enable_mux7_3,enable_mux7_4,enable_mux7_5,
809 enable_mux7_6,enable_mux7_7 : std_logic_vector(2 downto 0);
810 signal enable_mux5_1,enable_mux5_2 : std_logic_vector(2 downto 0);
811 signal enable_mux4_1,enable_mux4_2,enable_mux4_3,enable_mux4_4,enable_mux4_5,
812 enable_mux4_6 : std_logic_vector(1 downto 0);
813 signal enable_mux3_1,enable_mux3_2,enable_mux3_3,enable_mux3_4,enable_mux3_5,
814 enable_mux3_6,enable_mux3_7 : std_logic_vector(1 downto 0);
815 signal enable_mux2 : std_logic_vector(34 downto 1);
816 signal operation_sr1,operation_sr2,operation_sr3,operation_sr4,operation_sr5,
817 operation_sr6,operation_sr7,operation_sr8 : std_logic_vector(5 downto 0):="000000";
818 signal sclr_sr1,sclr_sr2,sclr_sr3,sclr_sr4,sclr_sr5,sclr_sr6,sclr_sr7,sclr_sr8,sclr_M1,
819 sclr_M2,sclr_M3,sclr_M4,sclr_M5,sclr_M6,sclr_M7,sclr_M8 : std_logic;
820 signal ce_sr1,ce_sr2,ce_sr3,ce_sr4,ce_sr5,ce_sr6,ce_sr7,ce_sr8,ce_M1,ce_M2,ce_M3,ce_M4,
821 ce_M5,ce_M6,ce_M7,ce_M8 : std_logic;
822 signal operation_nd_sr1,operation_nd_sr2,operation_nd_sr3,operation_nd_sr4,operation_nd_sr5,
823 operation_nd_sr6,operation_nd_sr7,operation_nd_sr8,operation_nd_M1,operation_nd_M2,
824 operation_nd_M3,operation_nd_M4,operation_nd_M5,operation_nd_M6,operation_nd_M7,
825 operation_nd_M8 : std_logic;
826 signal senales_de_control_1: std_logic_vector(272 downto 0);
```

```
827 signal banderas_1,banderas_2 : std_logic_vector(15 downto 0);
828 signal operation_rfd_sr1,operation_rfd_sr2,operation_rfd_sr3,operation_rfd_sr4,operation_rfd_sr5,
829 operation_rfd_sr6,operation_rfd_sr7,operation_rfd_sr8,operation_rfd_M1,operation_rfd_M2,
830 operation_rfd_M3,operation_rfd_M4,operation_rfd_M5,operation_rfd_M6,operation_rfd_M7,
831 operation_rfd_M8 : std_logic;
832 signal rdy_sr1,rdy_sr2,rdy_sr3,rdy_sr4,rdy_sr5,rdy_sr6,rdy_sr7,rdy_sr8,rdy_M1,rdy_M2,rdy_M3,
833 rdy_M4,rdy_M5,rdy_M6,rdy_M7,rdy_M8 : std_logic;
834
835 attribute maxdelay : string;
836 signal sr2,sr3,sr4,sr5,sr6,sr7,sr8,sr9,sr10,sr11,sr12,sr13,sr14,sr15,sr16,sr17,sr18,sr19,sr20,
837 sr21,sr28,sr29,sr30,sr31,sr32,sr33,sr34,sr35,sr36,sr37,sr38,sr39,sr40,sr41,sr42,sr43,sr44,sr45,
838 sr46,sr47,sr48,sr49,sr50,sr51,sr52,sr53,sr54,sr55,sr56,sr57,
839 sr58,sr59 : std_logic_vector(31 downto 0);
840 attribute maxdelay of sr42: signal is "10 ns";
841
842 signal salida_m1,salida_m2,salida_m3,salida_m4,salida_m5,salida_m6,
843 salida_m7 : std_logic_vector(31 downto 0);
844 signal estado_sig : std_logic_vector(4 downto 0);
845 signal inicio1: std_logic;
846
847 attribute keep : string;
848 attribute equivalent_register_removal: string;
849 signal R22,R23,R24,R25,R26,R27 : std_logic_vector(31 downto 0);
850 attribute equivalent_register_removal of R22,R23,R24,R25,R26,R27: signal is "no";
851 attribute keep of R22,R23,R24,R25,R26,R27: signal is "true";
852
853 constant cero_cinco : std_logic_vector(31 downto 0) :="00111111000000000000000000000000";--0.5 FP
854 constant dos : std_logic_vector(31 downto 0)        :="01000000000000000000000000000000";--2 FP
855 constant P623 : std_logic_vector(31 downto 0)       :=x"3F1F9D07";--  0.6234898 en FP
856 constant menos_P222 : std_logic_vector(31 downto 0) :=x"BE63DC87";-- -0.22252093 en FP
857 constant menos_P900 : std_logic_vector(31 downto 0) :=x"BF66A5E5";-- -0.90096885 en FP
858 constant P781 : std_logic_vector(31 downto 0)       :=x"3F48261C";--  0.7818315 en FP
859 constant menos_P781 : std_logic_vector(31 downto 0) :=x"BF48261C";-- -0.7818315 en FP
860 constant P974 : std_logic_vector(31 downto 0)       :=x"3F7994E0";--  0.9749279 en FP
861 constant menos_P974 : std_logic_vector(31 downto 0) :=x"BF7994E0";-- -0.9749279 en FP
862 constant P433 : std_logic_vector(31 downto 0)       :=x"3EDE2602";--  0.43388373 en FP
863 constant menos_P433 : std_logic_vector(31 downto 0) :=x"BEDE2602";-- -0.43388373 en FP
864 begin
865 -------Asignaciòn de Constantes Cx
866 process (clk,reset)
867 begin
868    if reset = '1' then
869       R22<="00000000000000000000000000000000";
870                 R23<="00000000000000000000000000000000";
871                 R24<="00000000000000000000000000000000";
872                 R25<="00000000000000000000000000000000";
873                 R26<="00000000000000000000000000000000";
874                 R27<="00000000000000000000000000000000";
875    elsif (clk'event and clk = '1') then
876       if estado = "00001" then
877          case (mu) is
878                                when "0001" =>
879                                        R22<= P623;
```

```
880                                         R23<=menos_P222;
881                                         R24<=menos_P900;
882                                         R25<=P781;
883                                         R26<=P974;
884                                         R27<=P433;
885                                 when "0010" =>
886                                         R22<= menos_P222;
887                                         R23<= menos_P900;
888                                         R24<=P623;
889                                         R25<=P974;
890                                         R26<=menos_P433;
891                                         R27<=menos_P781;
892                                 when "0011" =>
893                                         R22<= menos_P900;
894                                         R23<=P623;
895                                         R24<=menos_P222;
896                                         R25<=P433;
897                                         R26<=menos_P781;
898                                         R27<=P974;
899                                 when "0100" =>
900                                         R22<= menos_P900;
901                                         R23<=P623;
902                                         R24<=menos_P222;
903                                         R25<=menos_P433;
904                                         R26<=P781;
905                                         R27<=menos_P974;
906                                 when "0101" =>
907                                         R22<= menos_P222;
908                                         R23<=menos_P900;
909                                         R24<=P623;
910                                         R25<=menos_P974;
911                                         R26<=P433;
912                                         R27<=P781;
913                                 when others =>
914                                         R22<= P623;
915                                         R23<=menos_P222;
916                                         R24<=menos_P900;
917                                         R25<=menos_P781;
918                                         R26<=menos_P974;
919                                         R27<=menos_P433;
920                         end case;
921         else
922            R22<=R22;
923                         R23<=R23;
924                         R24<=R24;
925                         R25<=R25;
926                         R26<=R26;
927                         R27<=R27;
928         end if;
929     end if;
930 end process;
931 --------async trap and reset
932 Inst_async_trap_and_reset_forward: async_trap_and_reset3
```

```
933  generic map(  retardo => 4)
934  PORT MAP(
935                  async_sig => inicio,
936                  outclk => clk,
937                  auto_reset => '1',
938                  reset => not reset,
939                  out_sync_sig => inicio1
940          );
941  ------------------------------inicio de SR1
942  Inst_Seccion_SR1: Seccion_SR1 PORT MAP(
943                  clk => clk,
944                  reset => reset,
945                  mux7_1_a => sr4,
946                  mux7_1_b => sr45,
947                  mux7_1_c => sr6,
948                  mux7_1_d => sr41,
949                  mux7_1_e => sr29,
950                  mux7_1_f => sr33,
951                  mux7_1_g => sr17,
952                  mux7_2_a => sr14,
953                  mux7_2_b => sr43,
954                  mux7_2_c => sr12,
955                  mux7_2_d => sr51,
956                  mux7_2_e => sr36,
957                  mux7_2_f => sr32,
958                  mux7_2_g => dos,
959                  salida_m2 => salida_m2,
960                  z01 => z01,
961                  z05 => z05,
962                  j5 => j5,
963                  sr29 => sr29,
964                  sr37 => sr37,
965                  sr4 => sr4,
966                  sr12 => sr12,
967                  sr16 => sr16,
968                  enable_registers_4 => enable_registers_4,
969                  enable_registers_12 => enable_registers_12,
970                  enable_registers_16 => enable_registers_16,
971                  enable_registers_29 => enable_registers_29,
972                  enable_registers_37 => enable_registers_37,
973                  enable_mux7_1 => enable_mux7_1,
974                  enable_mux7_2 => enable_mux7_2,
975                  enable_mux2_1 => enable_mux2(1),
976                  enable_mux2_2 => enable_mux2(2),
977                  enable_mux2_3 => enable_mux2(3),
978                  enable_mux2_4 => enable_mux2(4),
979                  operation_sr1 => operation_sr1,
980                  operation_nd_sr1 => operation_nd_sr1,
981                  operation_rfd_sr1 => operation_rfd_sr1,
982                  sclr_sr1 => sclr_sr1,
983                  ce_sr1 => ce_sr1,
984                  underflow_sr1 => underflow_sr1,
985                  overflow_sr1 => overflow_sr1,
```

```
986                 invalid_op_sr1 => invalid_op_sr1,
987                 rdy_sr1 => rdy_sr1
988         );
989
990
991 ------------------------------------------fin de SR1
992 ------------------------------------------inicio de SR2
993 Inst_Seccion_SR2: Seccion_SR2 PORT MAP(
994                 clk => clk,
995                 reset => reset,
996                 mux8_1_a => sr5,
997                 mux8_1_b => sr46,
998                 mux8_1_c => sr7,
999                 mux8_1_d => sr42,
1000                mux8_1_e => sr3,
1001                mux8_1_f => sr30,
1002                mux8_1_g => sr34,
1003                mux8_1_h => sr18,
1004                mux7_3_a => sr15,
1005                mux7_3_b => sr44,
1006                mux7_3_c => sr13,
1007                mux7_3_d => sr35,
1008                mux7_3_e => sr46,
1009                mux7_3_f => sr31,
1010                mux7_3_g => dos,
1011                salida_m4 => salida_m4,
1012                salida_m2 => salida_m2,
1013                z01_1 => z01_1,
1014                z05_1 => z05_1,
1015                j4 => j4,
1016                sr30 => sr30,
1017                sr38 => sr38,
1018                sr46 => sr46,
1019                sr5 => sr5,
1020                sr13 => sr13,
1021                sr17 => sr17,
1022                enable_registers_5 => enable_registers_5,
1023                enable_registers_13 => enable_registers_13,
1024                enable_registers_17 => enable_registers_17,
1025                enable_registers_30 => enable_registers_30,
1026                enable_registers_38 => enable_registers_38,
1027                enable_registers_46 => enable_registers_46,
1028                enable_mux8_1 => enable_mux8_1,
1029                enable_mux7_3 => enable_mux7_3,
1030                enable_mux2_5 => enable_mux2(5),
1031                enable_mux2_6 => enable_mux2(6),
1032                enable_mux2_7 => enable_mux2(7),
1033                enable_mux2_8 => enable_mux2(8),
1034                enable_mux2_9 => enable_mux2(9),
1035                operation_sr2 => operation_sr2,
1036                operation_nd_sr2 => operation_nd_sr2,
1037                operation_rfd_sr2 => operation_rfd_sr2,
1038                sclr_sr2 => sclr_sr2,
```

```
1039                    ce_sr2 => ce_sr2,
1040                    underflow_sr2 => underflow_sr2,
1041                    overflow_sr2 => overflow_sr2,
1042                    invalid_op_sr2 => invalid_op_sr2,
1043                    rdy_sr2 => rdy_sr2
1044            );
1045
1046 ------------------------------------fin de SR2------
1047 ----------------inicio SR3-------------
1048  Inst_Seccion_SR3: Seccion_SR3 PORT MAP(
1049                    clk => clk,
1050                    reset => reset,
1051                    mux11_1_a => sr6,
1052                    mux11_1_b => sr40,
1053                    mux11_1_c => sr49,
1054                    mux11_1_d => sr8,
1055                    mux11_1_e => sr56,
1056                    mux11_1_f => sr51,
1057                    mux11_1_g => sr33,
1058                    mux11_1_h => sr29,
1059                    mux11_1_i => sr19,
1060                    mux11_1_j => sr2,
1061                    mux11_1_k => sr30,
1062
1063                    mux11_2_a => sr12,
1064                    mux11_2_b => sr54,
1065                    mux11_2_c => sr36,
1066                    mux11_2_d => sr10,
1067                    mux11_2_e => sr57,
1068                    mux11_2_f => sr52,
1069                    mux11_2_g => sr32,
1070                    mux11_2_h => sr36,
1071                    mux11_2_i => dos,
1072                    mux11_2_j => sr45,
1073                    mux11_2_k => sr34,
1074
1075                    salida_m1 => salida_m1,
1076                    z02 => z02,
1077                    z06 => z06,
1078                    j3 => j3,
1079                    sr31 => sr31,
1080                    sr39 => sr39,
1081                    sr47 => sr47,
1082                    sr6 => sr6,
1083                    sr14 => sr14,
1084                    sr18 => sr18,
1085                    sr45 => sr45,
1086                    enable_registers_6  => enable_registers_6,
1087                    enable_registers_14 => enable_registers_14,
1088                    enable_registers_18 => enable_registers_18,
1089                    enable_registers_31 => enable_registers_31,
1090                    enable_registers_39 => enable_registers_39,
1091                    enable_registers_47 => enable_registers_47,
```

```
1092                    enable_registers_45 => enable_registers_45,
1093                    enable_mux11_1 => enable_mux11_1,
1094                    enable_mux11_2 => enable_mux11_2,
1095                    enable_mux2_10 => enable_mux2(10),
1096                    enable_mux2_11 => enable_mux2(11),
1097                    enable_mux2_12 => enable_mux2(12),
1098                    enable_mux2_13 => enable_mux2(13),
1099                    enable_mux2_14 => enable_mux2(14),
1100                    operation_sr3 => operation_sr3,
1101                    operation_nd_sr3 => operation_nd_sr3,
1102                    operation_rfd_sr3 => operation_rfd_sr3,
1103                    sclr_sr3 => sclr_sr3,
1104                    ce_sr3 => ce_sr3,
1105                    underflow_sr3 => underflow_sr3,
1106                    overflow_sr3 => overflow_sr3,
1107                    invalid_op_sr3 => invalid_op_sr3,
1108                    rdy_sr3 => rdy_sr3
1109            );
1110
1111 ----------------------------------fin de SR3-------------------
1112 ---------------------------------Inicio de SR4--------------------
1113
1114 Inst_Seccion_SR4: Seccion_SR4 PORT MAP(
1115                    clk => clk,
1116                    reset => reset,
1117                    mux9_1_a => sr7,
1118                    mux9_1_b => sr55,
1119                    mux9_1_c => sr50,
1120                    mux9_1_d => sr9,
1121                    mux9_1_e => sr45,
1122                    mux9_1_f => sr53,
1123                    mux9_1_g => sr34,
1124                    mux9_1_h => sr30,
1125                    mux9_1_i => sr20,
1126                    mux9_2_a => sr13,
1127                    mux9_2_b => sr56,
1128                    mux9_2_c => sr39,
1129                    mux9_2_d => sr11,
1130                    mux9_2_e => sr52,
1131                    mux9_2_f => sr54,
1132                    mux9_2_g => sr31,
1133                    mux9_2_h => sr35,
1134                    mux9_2_i => dos,
1135                    salida_m2 => salida_m2,
1136                    salida_m3 => salida_m3,
1137                    z02_1 => z02_1,
1138                    z06_1 => z06_1,
1139                    j2 => j2,
1140                    sr40 => sr40,
1141                    sr48 => sr48,
1142                    sr7 => sr7,
1143                    sr15 => sr15,
1144                    sr19 => sr19,
```

```
1145                    sr32 => sr32,
1146                    enable_registers_7 => enable_registers_7,
1147                    enable_registers_15 => enable_registers_15,
1148                    enable_registers_19 => enable_registers_19,
1149                    enable_registers_32 => enable_registers_32,
1150                    enable_registers_40 => enable_registers_40,
1151                    enable_registers_48 => enable_registers_48,
1152                    enable_mux9_1 => enable_mux9_1,
1153                    enable_mux9_2 => enable_mux9_2,
1154                    enable_mux2_15 => enable_mux2(15),
1155                    enable_mux2_16 => enable_mux2(16),
1156                    enable_mux2_17 => enable_mux2(17),
1157                    enable_mux2_18 => enable_mux2(18),
1158                    enable_mux2_19 => enable_mux2(19),
1159                    operation_sr4 => operation_sr4,
1160                    operation_nd_sr4 => operation_nd_sr4,
1161                    operation_rfd_sr4 => operation_rfd_sr4,
1162                    sclr_sr4 => sclr_sr4,
1163                    ce_sr4 => ce_sr4,
1164                    underflow_sr4 => underflow_sr4,
1165                    overflow_sr4 => overflow_sr4,
1166                    invalid_op_sr4 => invalid_op_sr4,
1167                    rdy_sr4 => rdy_sr4
1168            );
1169
1170 ------------------------------fin de SR4-------------------
1171 ----------------------Inicio de secciòn SR5
1172  Inst_Seccion_SR5: Seccion_SR5 PORT MAP(
1173                    clk => clk,
1174                    reset => reset,
1175                    mux9_3_a => sr8,
1176                    mux9_3_b => sr31,
1177                    mux9_3_c => sr45,
1178                    mux9_3_d => sr29,
1179                    mux9_3_e => sr49,
1180                    mux9_3_f => sr2,
1181                    mux9_3_g => sr40,
1182                    mux9_3_h => sr16,
1183                    mux7_4_a => sr10,
1184                    mux7_4_b => sr33,
1185                    mux7_4_c => sr47,
1186                    mux7_4_d => sr50,
1187                    mux7_4_e => sr44,
1188                    mux7_4_f => sr38,
1189                    mux7_4_g => dos,
1190                    salida_m1 => salida_m1,
1191                    salida_m6 => salida_m6,
1192                    z00 => z00,
1193                    z03 => z03,
1194                    j0 => j0,
1195                    j6 => j6,
1196                    sr33 => sr33,
1197                    sr41 => sr41,
```

```
1198                    sr52 => sr52,
1199                    sr2 => sr2,
1200                    sr8 => sr8,
1201                    sr21 => sr21,
1202                    sr28_0 => sr28,
1203                    enable_registers_2 => enable_registers_2,
1204                    enable_registers_8 => enable_registers_8,
1205                    enable_registers_21 => enable_registers_21,
1206                    enable_registers_28 => enable_registers_28,
1207                    enable_registers_33 => enable_registers_33,
1208                    enable_registers_41 => enable_registers_41,
1209                    enable_registers_52 => enable_registers_52,
1210                    enable_mux9_3 => enable_mux9_3,
1211                    enable_mux7_4 => enable_mux7_4,
1212                    enable_mux2_20 => enable_mux2(20),
1213                    enable_mux2_21 => enable_mux2(21),
1214                    enable_mux2_22 => enable_mux2(22),
1215                    enable_mux2_23 => enable_mux2(23),
1216                    enable_mux2_24 => enable_mux2(24),
1217                    enable_mux2_25 => enable_mux2(25),
1218                    operation_sr5 => operation_sr5,
1219                    operation_nd_sr5 => operation_nd_sr5,
1220                    operation_rfd_sr5 => operation_rfd_sr5,
1221                    sclr_sr5 => sclr_sr5,
1222                    ce_sr5 => ce_sr5,
1223                    underflow_sr5 => underflow_sr5,
1224                    overflow_sr5 => overflow_sr5,
1225                    invalid_op_sr5 => invalid_op_sr5,
1226                    rdy_sr5 => rdy_sr5
1227            );
1228 ----------------------fin de SR5-----------
1229 ----------------------Inicio de SR6---------------
1230 Inst_Seccion_SR6: Seccion_SR6 PORT MAP(
1231                    clk => clk,
1232                    reset => reset,
1233                    mux8_2_a => sr9,
1234                    mux8_2_b => sr32,
1235                    mux8_2_c => sr46,
1236                    mux8_2_d => sr30,
1237                    mux8_2_e => sr51,
1238                    mux8_2_f => sr3,
1239                    mux8_2_g => sr43,
1240                    mux8_2_h => sr21,
1241                    mux7_5_a => sr11,
1242                    mux7_5_b => sr34,
1243                    mux7_5_c => sr48,
1244                    mux7_5_d => sr52,
1245                    mux7_5_e => sr45,
1246                    mux7_5_f => sr37,
1247                    mux7_5_g => dos,
1248                    salida_m2 => salida_m2,
1249                    salida_m7 => salida_m7,
1250                    z00_1 => z00_1,
```

```
1251                    z03_1 => z03_1,
1252                    j1 => j1,
1253                    sr34 => sr34,
1254                    sr42 => sr42,
1255                    sr53 => sr53,
1256                    sr3 => sr3,
1257                    sr9 => sr9,
1258                    sr20 => sr20,
1259                    enable_registers_3 => enable_registers_3,
1260                    enable_registers_9 => enable_registers_9,
1261                    enable_registers_20 => enable_registers_20,
1262                    enable_registers_34 => enable_registers_34,
1263                    enable_registers_42 => enable_registers_42,
1264                    enable_registers_53 => enable_registers_53,
1265                    enable_mux8_2 => enable_mux8_2,
1266                    enable_mux7_5 => enable_mux7_5,
1267                    enable_mux2_26 => enable_mux2(26),
1268                    enable_mux2_27 => enable_mux2(27),
1269                    enable_mux2_28 => enable_mux2(28),
1270                    enable_mux2_29 => enable_mux2(29),
1271                    enable_mux2_30 => enable_mux2(30),
1272                    operation_sr6 => operation_sr6,
1273                    operation_nd_sr6 => operation_nd_sr6,
1274                    operation_rfd_sr6 => operation_rfd_sr6,
1275                    sclr_sr6 => sclr_sr6,
1276                    ce_sr6 => ce_sr6,
1277                    underflow_sr6 => underflow_sr6,
1278                    overflow_sr6 => overflow_sr6,
1279                    invalid_op_sr6 => invalid_op_sr6,
1280                    rdy_sr6 => rdy_sr6
1281            );
1282 ----------------fin de SR6----------------
1283 ----------------inicio de SR7----------------
1284 Inst_Seccion_SR7: Seccion_SR7 PORT MAP(
1285                    clk => clk,
1286                    reset => reset,
1287                    mux8_3_a => sr4,
1288                    mux8_3_b => sr54,
1289                    mux8_3_c => sr37,
1290                    mux8_3_d => sr31,
1291                    mux8_3_e => sr47,
1292                    mux8_3_f => sr46,
1293                    mux8_3_g => sr53,
1294                    mux8_3_h => sr40,
1295                    mux8_4_a => sr14,
1296                    mux8_4_b => sr55,
1297                    mux8_4_c => sr31,
1298                    mux8_4_d => sr33,
1299                    mux8_4_e => sr48,
1300                    mux8_4_f => sr53,
1301                    mux8_4_g => sr54,
1302                    mux8_4_h => sr38,
1303                    salida_m5 => salida_m5,
```

```
1304                    z04 => z04,
1305                    sr35 => sr35,
1306                    sr43 => sr43,
1307                    sr51 => sr51,
1308                    sr10 => sr10,
1309                    enable_registers_10 => enable_registers_10,
1310                    enable_registers_35 => enable_registers_35,
1311                    enable_registers_43 => enable_registers_43,
1312                    enable_registers_51 => enable_registers_51,
1313                    enable_mux8_3 => enable_mux8_3,
1314                    enable_mux8_4 => enable_mux8_4,
1315                    enable_mux2_31 => enable_mux2(31),
1316                    enable_mux2_32 => enable_mux2(32),
1317                    operation_sr7 => operation_sr7,
1318                    operation_nd_sr7 => operation_nd_sr7,
1319                    operation_rfd_sr7 => operation_rfd_sr7,
1320                    sclr_sr7 => sclr_sr7,
1321                    ce_sr7 => ce_sr7,
1322                    underflow_sr7 => underflow_sr7,
1323                    overflow_sr7 => overflow_sr7,
1324                    invalid_op_sr7 => invalid_op_sr7,
1325                    rdy_sr7 => rdy_sr7
1326            );
1327
1328 ----------------fin de SR7--------------
1329 ----------Inicio de SR8--------
1330  Inst_Seccion_SR8: Seccion_SR8 PORT MAP(
1331                    clk => clk,
1332                    reset => reset,
1333                    mux7_6_a => sr5,
1334                    mux7_6_b => sr58,
1335                    mux7_6_c => sr38,
1336                    mux7_6_d => sr32,
1337                    mux7_6_e => sr49,
1338                    mux7_6_f => sr29,
1339                    mux7_6_g => sr43,
1340                    mux7_7_a => sr15,
1341                    mux7_7_b => sr59,
1342                    mux7_7_c => sr32,
1343                    mux7_7_d => sr34,
1344                    mux7_7_e => sr50,
1345                    mux7_7_f => sr33,
1346                    mux7_7_g => sr37,
1347                    z04_1 => z04_1,
1348                    sr36 => sr36,
1349                    sr44 => sr44,
1350                    sr11 => sr11,
1351                    enable_registers_11 => enable_registers_11,
1352                    enable_registers_36 => enable_registers_36,
1353                    enable_registers_44 => enable_registers_44,
1354                    enable_mux7_6 => enable_mux7_6,
1355                    enable_mux7_7 => enable_mux7_7,
1356                    enable_mux2_33 => enable_mux2(33),
```

```
1357                    operation_sr8 => operation_sr8,
1358                    operation_nd_sr8 => operation_nd_sr8,
1359                    operation_rfd_sr8 => operation_rfd_sr8,
1360                    sclr_sr8 => sclr_sr8,
1361                    ce_sr8 => ce_sr8,
1362                    underflow_sr8 => underflow_sr8,
1363                    overflow_sr8 => overflow_sr8,
1364                    invalid_op_sr8 => invalid_op_sr8,
1365                    rdy_sr8 => rdy_sr8
1366            );
1367
1368 ----------fin de SR8----------------
1369 ---------Inicio de M1----------------
1370  Inst_Seccion_M1: Seccion_M1 PORT MAP(
1371                    clk => clk,
1372                    reset => reset,
1373                    mux4_1_a => cero_cinco,
1374                    mux4_1_b => R22,
1375                    mux4_1_c => R27,
1376                    mux4_1_d => R26,
1377                    mux5_1_a => sr33,
1378                    mux5_1_b => sr41,
1379                    mux5_1_c => sr35,
1380                    mux5_1_d => sr40,
1381                    mux5_1_e => sr38,
1382                    sr55 => sr55,
1383                    enable_registers_55 => enable_registers_55,
1384                    enable_mux4_1 => enable_mux4_1,
1385                    enable_mux5_1 => enable_mux5_1,
1386                    operation_nd_M1 => operation_nd_M1,
1387                    operation_rfd_M1 => operation_rfd_M1,
1388                    sclr_M1 => sclr_M1,
1389                    ce_M1 => ce_M1,
1390                    result_M1 =>salida_m1,
1391                    underflow_M1 => underflow_M1,
1392                    overflow_M1 => overflow_M1,
1393                    invalid_op_M1 => invalid_op_M1,
1394                    rdy_M1 => rdy_M1
1395            );
1396 ----------Fin de M1----------------
1397 ---------Inicio de M2----------
1398  Inst_Seccion_M2: Seccion_M2 PORT MAP(
1399                    clk => clk,
1400                    reset => reset,
1401                    mux4_2_a => cero_cinco,
1402                    mux4_2_b => R23,
1403                    mux4_2_c => R27,
1404                    mux4_2_d => R25,
1405                    mux5_2_a => sr34,
1406                    mux5_2_b => sr43,
1407                    mux5_2_c => sr36,
1408                    mux5_2_d => sr35,
1409                    mux5_2_e => sr37,
```

```
1410                    sr58 => sr58,
1411                    enable_registers_58 => enable_registers_58,
1412                    enable_mux4_2 => enable_mux4_2,
1413                    enable_mux5_2 => enable_mux5_2,
1414                    operation_nd_M2 => operation_nd_M2,
1415                    operation_rfd_M2 => operation_rfd_M2,
1416                    sclr_M2 => sclr_M2,
1417                    ce_M2 => ce_M2,
1418                    result_M2 => salida_m2,
1419                    underflow_M2 => underflow_M2,
1420                    overflow_M2 => overflow_M2,
1421                    invalid_op_M2 => invalid_op_M2,
1422                    rdy_M2 => rdy_M2
1423            );
1424 --------Fin de M2-----------------
1425 ---------Inicio de M3----------
1426 Inst_Seccion_M3: Seccion_M3 PORT MAP(
1427                    clk => clk,
1428                    reset => reset,
1429                    mux4_3_a => sr42,
1430                    mux4_3_b => sr41,
1431                    mux4_3_c => sr35,
1432                    mux4_3_d => sr37,
1433                    mux3_1_a => R22,
1434                    mux3_1_b => R24,
1435                    mux3_1_c => R26,
1436                    sr49 => sr49,
1437                    enable_registers_49 => enable_registers_49,
1438                    enable_mux4_3 => enable_mux4_3,
1439                    enable_mux3_1 => enable_mux3_1,
1440                    operation_nd_M3 => operation_nd_M3,
1441                    operation_rfd_M3 => operation_rfd_M3,
1442                    sclr_M3 => sclr_M3,
1443                    ce_M3 => ce_M3,
1444                    result_M3 => salida_m3,
1445                    underflow_M3 => underflow_M3,
1446                    overflow_M3 => overflow_M3,
1447                    invalid_op_M3 => invalid_op_M3,
1448                    rdy_M3 => rdy_M3
1449            );
1450 ----------Fin de M3---------
1451 ---------Inicio de M4
1452 Inst_Seccion_M4: Seccion_M4 PORT MAP(
1453                    clk => clk,
1454                    reset => reset,
1455                    mux4_4_a => R23,
1456                    mux4_4_b => R22,
1457                    mux4_4_c => R26,
1458                    mux4_4_d => R25,
1459                    mux3_2_a => sr44,
1460                    mux3_2_b => sr43,
1461                    mux3_2_c => sr36,
1462                    sr50 => sr50,
```

```
1463                    enable_registers_50 => enable_registers_50,
1464                    enable_mux4_4 => enable_mux4_4,
1465                    enable_mux3_2 => enable_mux3_2,
1466                    operation_nd_M4 => operation_nd_M4,
1467                    operation_rfd_M4 => operation_rfd_M4,
1468                    sclr_M4 => sclr_M4,
1469                    ce_M4 => ce_M4,
1470                    result_M4 => salida_m4,
1471                    underflow_M4 => underflow_M4,
1472                    overflow_M4 => overflow_M4,
1473                    invalid_op_M4 => invalid_op_M4,
1474                    rdy_M4 => rdy_M4
1475            );
1476 --------Fin de M4
1477 ---------Inicio de M5------
1478 Inst_Seccion_M5: Seccion_M5 PORT MAP(
1479                    clk => clk,
1480                    reset => reset,
1481                    mux3_3_a => R23,
1482                    mux3_3_b => R24,
1483                    mux3_3_c => R25,
1484                    mux3_4_a => sr41,
1485                    mux3_4_b => sr42,
1486                    mux3_4_c => sr39,
1487                    sr59 => sr59,
1488                    enable_registers_59 => enable_registers_59,
1489                    enable_mux3_3 => enable_mux3_3,
1490                    enable_mux3_4 => enable_mux3_4,
1491                    operation_nd_M5 => operation_nd_M5,
1492                    operation_rfd_M5 => operation_rfd_M5,
1493                    sclr_M5 => sclr_M5,
1494                    ce_M5 => ce_M5,
1495                    result_M5 => salida_m5,
1496                    underflow_M5 => underflow_M5,
1497                    overflow_M5 => overflow_M5,
1498                    invalid_op_M5 => invalid_op_M5,
1499                    rdy_M5 => rdy_M5
1500            );
1501 --------Fin de M5----------
1502 ----------Inicio de M6--------
1503 Inst_Seccion_M6: Seccion_M6 PORT MAP(
1504                    clk => clk,
1505                    reset => reset,
1506                    mux4_5_a => sr43,
1507                    mux4_5_b => sr44,
1508                    mux4_5_c => sr38,
1509                    mux4_5_d => sr40,
1510                    mux3_5_a => R24,
1511                    mux3_5_b => R22,
1512                    mux3_5_c => R27,
1513                    sr56 => sr56,
1514                    enable_registers_56 => enable_registers_56,
1515                    enable_mux4_5 => enable_mux4_5,
```

```
1516                    enable_mux3_5 => enable_mux3_5,
1517                    operation_nd_M6 => operation_nd_M6,
1518                    operation_rfd_M6 => operation_rfd_M6,
1519                    sclr_M6 => sclr_M6,
1520                    ce_M6 => ce_M6,
1521                    result_M6 => salida_m6,
1522                    underflow_M6 => underflow_M6,
1523                    overflow_M6 => overflow_M6,
1524                    invalid_op_M6 => invalid_op_M6,
1525                    rdy_M6 => rdy_M6
1526            );
1527 --------Fin de M6-----------------------
1528 -------Inicio de M7-------
1529 Inst_Seccion_M7: Seccion_M7 PORT MAP(
1530                    clk => clk,
1531                    reset => reset,
1532                    mux3_6_a => sr42,
1533                    mux3_6_b => sr37,
1534                    mux3_6_c => sr40,
1535                    mux2_34_a => R23,
1536                    mux2_34_b => R25,
1537                    sr57 => sr57,
1538                    enable_registers_57 => enable_registers_57,
1539                    enable_mux3_6 => enable_mux3_6,
1540                    enable_mux2_34 => enable_mux2(34),
1541                    operation_nd_M7 => operation_nd_M7,
1542                    operation_rfd_M7 => operation_rfd_M7,
1543                    sclr_M7 => sclr_M7,
1544                    ce_M7 => ce_M7,
1545                    result_M7 => salida_m7,
1546                    underflow_M7 => underflow_M7,
1547                    overflow_M7 => overflow_M7,
1548                    invalid_op_M7 => invalid_op_M7,
1549                    rdy_M7 => rdy_M7
1550            );
1551 --------Fin de M7----------
1552 -------Inicio de M8-------
1553 Inst_Seccion_M8: Seccion_M8 PORT MAP(
1554                    clk => clk,
1555                    reset => reset,
1556                    mux4_6_a => R24,
1557                    mux4_6_b => R26,
1558                    mux4_6_c => R25,
1559                    mux4_6_d => R27,
1560                    mux3_7_a => sr44,
1561                    mux3_7_b => sr39,
1562                    mux3_7_c => sr38,
1563                    sr54 => sr54,
1564                    enable_registers_54 => enable_registers_54,
1565                    enable_mux4_6 => enable_mux4_6,
1566                    enable_mux3_7 => enable_mux3_7,
1567                    operation_nd_M8 => operation_nd_M8,
1568                    operation_rfd_M8 => operation_rfd_M8,
```

```
1569                    sclr_M8 => sclr_M8,
1570                    ce_M8 => ce_M8,
1571                    underflow_M8 => underflow_M8,
1572                    overflow_M8 => overflow_M8,
1573                    invalid_op_M8 => invalid_op_M8,
1574                    rdy_M8 => rdy_M8
1575            );
1576 -----Fin de M8--------
1577
1578 Inst_FSM_7: FSM_7 PORT MAP(
1579                    clk => clk,
1580                    inicio => inicio1,
1581                    m => m,
1582                    reset => reset,
1583                    banderas1 => banderas_1,
1584        banderas2 => banderas_2,
1585        estado_sig => estado_sig,
1586                    senales_de_control => senales_de_control_1
1587            );
1588
1589 banderas_1 <= operation_rfd_sr1 & operation_rfd_sr2 & operation_rfd_sr3 & operation_rfd_sr4 &
1590 operation_rfd_sr5 & operation_rfd_sr6 & operation_rfd_sr7 & operation_rfd_sr8 &
1591 operation_rfd_m1 & operation_rfd_m2 & operation_rfd_m3 & operation_rfd_m4 & operation_rfd_m5 &
1592 operation_rfd_m6 & operation_rfd_m7 & operation_rfd_m8 ;
1593 banderas_11 <= banderas_1;
1594 banderas_2 <= rdy_sr1 & rdy_sr2 & rdy_sr3 & rdy_sr4 & rdy_sr5 & rdy_sr6 & rdy_sr7 & rdy_sr8 &
1595 rdy_M1 & rdy_M2 & rdy_M3 & rdy_M4 & rdy_M5 & rdy_M6 & rdy_M7 & rdy_M8;
1596 banderas_22 <= banderas_2;
1597
1598 ce_sr8 <= senales_de_control_1(0);
1599 ce_sr7 <= senales_de_control_1(1);
1600 ce_sr6 <= senales_de_control_1(2);
1601 ce_sr5 <= senales_de_control_1(3);
1602 ce_sr4 <= senales_de_control_1(4);
1603 ce_sr3 <= senales_de_control_1(5);
1604 ce_sr2 <= senales_de_control_1(6);
1605 ce_sr1 <= senales_de_control_1(7);
1606 ocupado <= senales_de_control_1(8);
1607 ce_M8 <= senales_de_control_1(9);
1608 ce_M7 <= senales_de_control_1(10);
1609 ce_M6 <= senales_de_control_1(11);
1610 ce_M5 <= senales_de_control_1(12);
1611 ce_M4 <= senales_de_control_1(13);
1612 ce_M3 <= senales_de_control_1(14);
1613 ce_M2 <= senales_de_control_1(15);
1614 ce_M1 <= senales_de_control_1(16);
1615 operation_nd_M8 <= senales_de_control_1(17);
1616 operation_nd_M7 <= senales_de_control_1(18);
1617 operation_nd_M6 <= senales_de_control_1(19);
1618 operation_nd_M5 <= senales_de_control_1(20);
1619 operation_nd_M4 <= senales_de_control_1(21);
1620 operation_nd_M3 <= senales_de_control_1(22);
1621 operation_nd_M2 <= senales_de_control_1(23);
```

```
1622  operation_nd_M1 <= senales_de_control_1(24);
1623  sclr_M8 <= senales_de_control_1(25);
1624  sclr_M7 <= senales_de_control_1(26);
1625  sclr_M6 <= senales_de_control_1(27);
1626  sclr_M5 <= senales_de_control_1(28);
1627  sclr_M4 <= senales_de_control_1(29);
1628  sclr_M3 <= senales_de_control_1(30);
1629  sclr_M2 <= senales_de_control_1(31);
1630  sclr_M1 <= senales_de_control_1(32);
1631  operation_nd_sr8 <= senales_de_control_1(33);
1632  operation_nd_sr7 <= senales_de_control_1(34);
1633  operation_nd_sr6 <= senales_de_control_1(35);
1634  operation_nd_sr5 <= senales_de_control_1(36);
1635  operation_nd_sr4 <= senales_de_control_1(37);
1636  operation_nd_sr3 <= senales_de_control_1(38);
1637  operation_nd_sr2 <= senales_de_control_1(39);
1638  operation_nd_sr1 <= senales_de_control_1(40);
1639  sclr_sr8 <= senales_de_control_1(41);
1640  sclr_sr7 <= senales_de_control_1(42);
1641  sclr_sr6 <= senales_de_control_1(43);
1642  sclr_sr5 <= senales_de_control_1(44);
1643  sclr_sr4 <= senales_de_control_1(45);
1644  sclr_sr3 <= senales_de_control_1(46);
1645  sclr_sr2 <= senales_de_control_1(47);
1646  sclr_sr1 <= senales_de_control_1(48);
1647  operation_sr8 <= senales_de_control_1(54 downto 49);
1648  operation_sr7 <= senales_de_control_1(60 downto 55);
1649  operation_sr6 <= senales_de_control_1(66 downto 61);
1650  operation_sr5 <= senales_de_control_1(72 downto 67);
1651  operation_sr4 <= senales_de_control_1(78 downto 73);
1652  operation_sr3 <= senales_de_control_1(84 downto 79);
1653  operation_sr2 <= senales_de_control_1(90 downto 85);
1654  operation_sr1 <= senales_de_control_1(96 downto 91);
1655  enable_mux2(34)<= senales_de_control_1(97);
1656  enable_mux2(33)<= senales_de_control_1(98);
1657  enable_mux2(32)<= senales_de_control_1(99);
1658  enable_mux2(31)<= senales_de_control_1(100);
1659  enable_mux2(30)<= senales_de_control_1(101);
1660  enable_mux2(29)<= senales_de_control_1(102);
1661  enable_mux2(28)<= senales_de_control_1(103);
1662  enable_mux2(27)<= senales_de_control_1(104);
1663  enable_mux2(26)<= senales_de_control_1(105);
1664  enable_mux2(25)<= senales_de_control_1(106);
1665  enable_mux2(24)<= senales_de_control_1(107);
1666  enable_mux2(23)<= senales_de_control_1(108);
1667  enable_mux2(22)<= senales_de_control_1(109);
1668  enable_mux2(21)<= senales_de_control_1(110);
1669  enable_mux2(20)<= senales_de_control_1(111);
1670  enable_mux2(19)<= senales_de_control_1(112);
1671  enable_mux2(18)<= senales_de_control_1(113);
1672  enable_mux2(17)<= senales_de_control_1(114);
1673  enable_mux2(16)<= senales_de_control_1(115);
1674  enable_mux2(15)<= senales_de_control_1(116);
```

```
1675  enable_mux2(14)<= senales_de_control_1(117);
1676  enable_mux2(13)<= senales_de_control_1(118);
1677  enable_mux2(12)<= senales_de_control_1(119);
1678  enable_mux2(11)<= senales_de_control_1(120);
1679  enable_mux2(10)<= senales_de_control_1(121);
1680  enable_mux2(9)<= senales_de_control_1(122);
1681  enable_mux2(8)<= senales_de_control_1(123);
1682  enable_mux2(7)<= senales_de_control_1(124);
1683  enable_mux2(6)<= senales_de_control_1(125);
1684  enable_mux2(5)<= senales_de_control_1(126);
1685  enable_mux2(4)<= senales_de_control_1(127);
1686  enable_mux2(3)<= senales_de_control_1(128);
1687  enable_mux2(2)<= senales_de_control_1(129);
1688  enable_mux2(1)<= senales_de_control_1(130);
1689  enable_mux3_7 <= senales_de_control_1(132 downto 131);
1690  enable_mux3_6 <= senales_de_control_1(134 downto 133);
1691  enable_mux3_5 <= senales_de_control_1(136 downto 135);
1692  enable_mux3_4 <= senales_de_control_1(138 downto 137);
1693  enable_mux3_3 <= senales_de_control_1(140 downto 139);
1694  enable_mux3_2 <= senales_de_control_1(142 downto 141);
1695  enable_mux3_1 <= senales_de_control_1(144 downto 143);
1696  enable_mux4_6 <= senales_de_control_1(146 downto 145);
1697  enable_mux4_5 <= senales_de_control_1(148 downto 147);
1698  enable_mux4_4 <= senales_de_control_1(150 downto 149);
1699  enable_mux4_3 <= senales_de_control_1(152 downto 151);
1700  enable_mux4_2 <= senales_de_control_1(154 downto 153);
1701  enable_mux4_1 <= senales_de_control_1(156 downto 155);
1702  enable_mux5_2 <= senales_de_control_1(159 downto 157);
1703  enable_mux5_1 <= senales_de_control_1(162 downto 160);
1704  enable_mux7_7 <= senales_de_control_1(165 downto 163);
1705  enable_mux7_6 <= senales_de_control_1(168 downto 166);
1706  enable_mux7_5 <= senales_de_control_1(171 downto 169);
1707  enable_mux7_4 <= senales_de_control_1(174 downto 172);
1708  enable_mux7_3 <= senales_de_control_1(177 downto 175);
1709  enable_mux7_2 <= senales_de_control_1(180 downto 178);
1710  enable_mux7_1 <= senales_de_control_1(183 downto 181);
1711  enable_mux8_4 <= senales_de_control_1(186 downto 184);
1712  enable_mux8_3 <= senales_de_control_1(189 downto 187);
1713  enable_mux8_2 <= senales_de_control_1(192 downto 190);
1714  enable_mux8_1 <= senales_de_control_1(195 downto 193);
1715  enable_mux9_3 <= senales_de_control_1(199 downto 196);
1716  enable_mux9_2 <= senales_de_control_1(203 downto 200);
1717  enable_mux9_1 <= senales_de_control_1(207 downto 204);
1718  enable_mux11_2 <= senales_de_control_1(211 downto 208);
1719  enable_mux11_1 <= senales_de_control_1(215 downto 212);
1720  enable_registers_59 <= senales_de_control_1(216);
1721  enable_registers_58 <= senales_de_control_1(217);
1722  enable_registers_57 <= senales_de_control_1(218);
1723  enable_registers_56 <= senales_de_control_1(219);
1724  enable_registers_55 <= senales_de_control_1(220);
1725  enable_registers_54 <= senales_de_control_1(221);
1726  enable_registers_53 <= senales_de_control_1(222);
1727  enable_registers_52 <= senales_de_control_1(223);
```

```
1728  enable_registers_51 <= senales_de_control_1(224);
1729  enable_registers_50 <= senales_de_control_1(225);
1730  enable_registers_49 <= senales_de_control_1(226);
1731  enable_registers_48 <= senales_de_control_1(227);
1732  enable_registers_47 <= senales_de_control_1(228);
1733  enable_registers_46 <= senales_de_control_1(229);
1734  enable_registers_45 <= senales_de_control_1(230);
1735  enable_registers_44 <= senales_de_control_1(231);
1736  enable_registers_43 <= senales_de_control_1(232);
1737  enable_registers_42 <= senales_de_control_1(233);
1738  enable_registers_41 <= senales_de_control_1(234);
1739  enable_registers_40 <= senales_de_control_1(235);
1740  enable_registers_39 <= senales_de_control_1(236);
1741  enable_registers_38 <= senales_de_control_1(237);
1742  enable_registers_37 <= senales_de_control_1(238);
1743  enable_registers_36 <= senales_de_control_1(239);
1744  enable_registers_35 <= senales_de_control_1(240);
1745  enable_registers_34 <= senales_de_control_1(241);
1746  enable_registers_33 <= senales_de_control_1(242);
1747  enable_registers_32 <= senales_de_control_1(243);
1748  enable_registers_31 <= senales_de_control_1(244);
1749  enable_registers_30 <= senales_de_control_1(245);
1750  enable_registers_29 <= senales_de_control_1(246);
1751  enable_registers_28 <= senales_de_control_1(247);
1752
1753  enable_registers_21 <=  '0' when (estado="11010" and estado_sig="00010") else
1754                          senales_de_control_1(248);
1755
1756  enable_registers_20 <= senales_de_control_1(249);
1757  enable_registers_19 <= senales_de_control_1(250);
1758  enable_registers_18 <= senales_de_control_1(251);
1759  enable_registers_17 <= senales_de_control_1(252);
1760  enable_registers_16 <= senales_de_control_1(253);
1761  enable_registers_15 <= senales_de_control_1(254);
1762  enable_registers_14 <= senales_de_control_1(255);
1763  enable_registers_13 <= senales_de_control_1(256);
1764  enable_registers_12 <= senales_de_control_1(257);
1765  enable_registers_11 <= senales_de_control_1(258);
1766  enable_registers_10 <= senales_de_control_1(259);
1767  enable_registers_9 <= senales_de_control_1(260);
1768  enable_registers_8 <= senales_de_control_1(261);
1769  enable_registers_7 <= senales_de_control_1(262);
1770  enable_registers_6 <= senales_de_control_1(263);
1771  enable_registers_5 <= senales_de_control_1(264);
1772  enable_registers_4 <= senales_de_control_1(265);
1773  enable_registers_3 <= senales_de_control_1(266);
1774  enable_registers_2 <= senales_de_control_1(267);
1775
1776  estado <= senales_de_control_1(272 downto 268);
1777
1778  z00_o   <= sr2;
1779  z00_1_o <= sr3;
1780  z01_o   <= sr4;
```

```
1781  z01_1_o <= sr5;
1782  z02_o   <= sr6;
1783  z02_1_o <= sr7;
1784  z03_o   <= sr8;
1785  z03_1_o <= sr9;
1786  z04_o   <= sr10;
1787  z04_1_o <= sr11;
1788  z05_o   <= sr12;
1789  z05_1_o <= sr13;
1790  z06_o   <= sr14;
1791  z06_1_o <= sr15;
1792
1793  j0_o <= sr21;
1794  j1_o <= sr20;
1795  j2_o <= sr19;
1796  j3_o <= sr18;
1797  j4_o <= sr17;
1798  j5_o <= sr16;
1799  j6_o <= sr28;
1800  --memoria de prueba-----
1801  process(clk,reset)
1802  begin
1803          if reset='1' then
1804                  direccion <= "0000";
1805          elsif (clk'event and clk = '1') then
1806                  if (estado="11010" and estado_sig="00010") then
1807                          direccion <= direccion +'1';
1808                  else
1809                          direccion <= direccion;
1810                  end if;
1811          else
1812                  direccion <= direccion;
1813          end if;
1814  end process;
1815
1816  process(clk)
1817  begin
1818          if (clk'event and clk = '1') then
1819                  if estado= "10101" or estado="10111" or estado="11001" then --s12 o s13 o s14
1820                          habilita_w<= '1';
1821                  else
1822                          habilita_w<= '0';
1823                  end if;
1824          end if;
1825  end process;
1826
1827
1828
1829  habilita1 <=    habilita_w when estado= "10101"or estado= "10110" else --s12 o s12w
1830
1831                  '0';
1832
1833
```

```
1834
1835 habilita2 <=     habilita_w when estado= "10111" or estado= "11000" else --s13 o s13w
1836                  '0';
1837
1838
1839
1840 habilita3 <=     habilita_w when estado= "11001"or estado= "11010" else --s14 o s14w
1841                  '0';
1842
1843
1844 ---Secciòn de los bloques de memoria manejados por el driver---
1845 ram_enable_driver_00   <= ram_enable_driver when bloque_ram_selection ="0000" else
1846                           '0';
1847
1848 ram_enable_driver_00_1 <= ram_enable_driver when bloque_ram_selection ="0001" else
1849                           '0';
1850
1851 ram_enable_driver_01   <= ram_enable_driver when bloque_ram_selection ="0010" else
1852                           '0';
1853
1854 ram_enable_driver_01_1 <= ram_enable_driver when bloque_ram_selection ="0011" else
1855                           '0';
1856
1857 ram_enable_driver_02   <= ram_enable_driver when bloque_ram_selection ="0100" else
1858                           '0';
1859
1860 ram_enable_driver_02_1 <= ram_enable_driver when bloque_ram_selection ="0101" else
1861                           '0';
1862
1863 ram_enable_driver_03   <= ram_enable_driver when bloque_ram_selection ="0110" else
1864                           '0';
1865
1866 ram_enable_driver_03_1 <= ram_enable_driver when bloque_ram_selection ="0111" else
1867                           '0';
1868
1869 ram_enable_driver_04   <= ram_enable_driver when bloque_ram_selection ="1000" else
1870                           '0';
1871
1872 ram_enable_driver_04_1 <= ram_enable_driver when bloque_ram_selection ="1001" else
1873                           '0';
1874
1875 ram_enable_driver_05   <= ram_enable_driver when bloque_ram_selection ="1010" else
1876                           '0';
1877
1878 ram_enable_driver_05_1 <= ram_enable_driver when bloque_ram_selection ="1011" else
1879                           '0';
1880
1881 ram_enable_driver_06   <= ram_enable_driver when bloque_ram_selection ="1100" else
1882                           '0';
1883
1884 ram_enable_driver_06_1 <= ram_enable_driver when bloque_ram_selection ="1101" else
1885                           '0';
1886
```

```
1887 ram_output_driver      <=   ram_output_driver_00   when bloque_ram_selection ="0000" else
1888                             ram_output_driver_00_1 when bloque_ram_selection ="0001" else
1889                             ram_output_driver_01   when bloque_ram_selection ="0010" else
1890                             ram_output_driver_01_1 when bloque_ram_selection ="0011" else
1891                             ram_output_driver_02   when bloque_ram_selection ="0100" else
1892                             ram_output_driver_02_1 when bloque_ram_selection ="0101" else
1893                             ram_output_driver_03   when bloque_ram_selection ="0110" else
1894                             ram_output_driver_03_1 when bloque_ram_selection ="0111" else
1895                             ram_output_driver_04   when bloque_ram_selection ="1000" else
1896                             ram_output_driver_04_1 when bloque_ram_selection ="1001" else
1897                             ram_output_driver_05   when bloque_ram_selection ="1010" else
1898                             ram_output_driver_05_1 when bloque_ram_selection ="1011" else
1899                             ram_output_driver_06   when bloque_ram_selection ="1100" else
1900                             ram_output_driver_06_1 when bloque_ram_selection ="1101" else
1901                             x"00000000";
1902
1903 Inst_memoria_z00: memoria_prueba
1904                  generic map(
1905                  INIT_00 =>x"42C80000",--100
1906                  INIT_01 =>x"42CA0000",
1907                  INIT_02 =>x"42CC0000",
1908                  INIT_03 =>x"42CE0000",
1909                  INIT_04 =>x"42D00000",
1910                  INIT_05 =>x"42D20000",
1911                  INIT_06 =>x"42D40000",
1912                  INIT_07 =>x"42D60000",
1913                  INIT_08 =>x"42D80000",
1914                  INIT_09 =>x"42DA0000",
1915                  INIT_0A =>x"42DC0000",
1916                  INIT_0B =>x"42DE0000",
1917                  INIT_0C =>x"42E00000",
1918                  INIT_0D =>x"42E20000",
1919                  INIT_0E =>x"42E40000",
1920                  INIT_0F =>x"42E60000",
1921                  ancho_dir => 4,
1922                  ancho_dat => 32
1923                  )
1924                 PORT MAP(
1925          clockA => clk_driver,
1926                  clockB => clk,
1927                  ram_enableA => ram_enable_driver_00,
1928                  ram_enableB => '1',
1929                  write_enableA => write_enable_driver,
1930                  write_enableB => habilita1,
1931                  input_dataA => input_data_driver,
1932                  input_dataB => z00_o,
1933                  ram_outputA => ram_output_driver_00,
1934                  ram_outputB => z00,
1935                  addressA => address_driver,
1936                  addressB => direccion
1937          );
1938
1939 Inst_memoria_z00_1: memoria_prueba
```

```
1940                    generic map(
1941                    INIT_00 =>x"43480000",--200
1942                    INIT_01 =>x"43490000",
1943                    INIT_02 =>x"434A0000",
1944                    INIT_03 =>x"434B0000",
1945                    INIT_04 =>x"434C0000",
1946                    INIT_05 =>x"434D0000",
1947                    INIT_06 =>x"434E0000",
1948                    INIT_07 =>x"434F0000",
1949                    INIT_08 =>x"43500000",
1950                    INIT_09 =>x"43510000",
1951                    INIT_0A =>x"43520000",
1952                    INIT_0B =>x"43530000",
1953                    INIT_0C =>x"43540000",
1954                    INIT_0D =>x"43550000",
1955                    INIT_0E =>x"43560000",
1956                    INIT_0F =>x"43570000",
1957                    ancho_dir => 4,
1958                    ancho_dat => 32
1959                    )
1960                    PORT MAP(
1961                    clockA => clk_driver,
1962                    clockB => clk,
1963                    ram_enableA => ram_enable_driver_00_1,
1964                    ram_enableB => '1',
1965                    write_enableA => write_enable_driver,
1966                    write_enableB => habilita1,
1967                    input_dataA => input_data_driver,
1968                    input_dataB => z00_1_o,
1969                    ram_outputA => ram_output_driver_00_1,
1970                    ram_outputB => z00_1,
1971                    addressA => address_driver,
1972                    addressB => direccion
1973            );
1974
1975    Inst_memoria_z01: memoria_prueba
1976                    generic map(
1977                    INIT_00 =>x"43960000",--300
1978                    INIT_01 =>x"43968000",
1979                    INIT_02 =>x"43970000",
1980                    INIT_03 =>x"43978000",
1981                    INIT_04 =>x"43980000",
1982                    INIT_05 =>x"43988000",
1983                    INIT_06 =>x"43990000",
1984                    INIT_07 =>x"43998000",
1985                    INIT_08 =>x"439A0000",
1986                    INIT_09 =>x"439A8000",
1987                    INIT_0A =>x"439B0000",
1988                    INIT_0B =>x"439B8000",
1989                    INIT_0C =>x"439C0000",
1990                    INIT_0D =>x"439C8000",
1991                    INIT_0E =>x"439D0000",
1992                    INIT_0F =>x"439D8000",
```

```
1993                    ancho_dir => 4,
1994                    ancho_dat => 32
1995                    )
1996                    PORT MAP(
1997                    clockA => clk_driver,
1998                    clockB => clk,
1999                    ram_enableA => ram_enable_driver_01,
2000                    ram_enableB => '1',
2001                    write_enableA => write_enable_driver,
2002                    write_enableB => habilita2,
2003                    input_dataA => input_data_driver,
2004                    input_dataB => z01_o,
2005                    ram_outputA => ram_output_driver_01,
2006                    ram_outputB => z01,
2007                    addressA => address_driver,
2008                    addressB => direccion
2009            );
2010
2011    Inst_memoria_z01_1: memoria_prueba
2012                    generic map(
2013                    INIT_00 =>x"43C80000",--400
2014                    INIT_01 =>x"43C88000",
2015                    INIT_02 =>x"43C90000",
2016                    INIT_03 =>x"43C98000",
2017                    INIT_04 =>x"43CA0000",
2018                    INIT_05 =>x"43CA8000",
2019                    INIT_06 =>x"43CB0000",
2020                    INIT_07 =>x"43CB8000",
2021                    INIT_08 =>x"43CC0000",
2022                    INIT_09 =>x"43CC8000",
2023                    INIT_0A =>x"43CD0000",
2024                    INIT_0B =>x"43CD8000",
2025                    INIT_0C =>x"43CE0000",
2026                    INIT_0D =>x"43CE8000",
2027                    INIT_0E =>x"43CF0000",
2028                    INIT_0F =>x"43CF8000",
2029                    ancho_dir => 4,
2030                    ancho_dat => 32
2031                    )
2032                    PORT MAP(
2033                    clockA => clk_driver,
2034                    clockB => clk,
2035                    ram_enableA => ram_enable_driver_01_1,
2036                    ram_enableB => '1',
2037                    write_enableA => write_enable_driver,
2038                    write_enableB => habilita2,
2039                    input_dataA => input_data_driver,
2040                    input_dataB => z01_1_o,
2041                    ram_outputA => ram_output_driver_01_1,
2042                    ram_outputB => z01_1,
2043                    addressA => address_driver,
2044                    addressB => direccion
2045            );
```

```
2046
2047 Inst_memoria_z02: memoria_prueba
2048                 generic map(
2049                 INIT_00 =>x"43FA0000",--500
2050                 INIT_01 =>x"43FA8000",
2051                 INIT_02 =>x"43FB0000",
2052                 INIT_03 =>x"43FB8000",
2053                 INIT_04 =>x"43FC0000",
2054                 INIT_05 =>x"43FC8000",
2055                 INIT_06 =>x"43FD0000",
2056                 INIT_07 =>x"43FD8000",
2057                 INIT_08 =>x"43FE0000",
2058                 INIT_09 =>x"43FE8000",
2059                 INIT_0A =>x"43FF0000",
2060                 INIT_0B =>x"43FF8000",
2061                 INIT_0C =>x"44000000",
2062                 INIT_0D =>x"44004000",
2063                 INIT_0E =>x"44008000",
2064                 INIT_0F =>x"4400C000",
2065                 ancho_dir => 4,
2066                 ancho_dat => 32
2067                 )
2068                 PORT MAP(
2069                 clockA => clk_driver,
2070                 clockB => clk,
2071                 ram_enableA => ram_enable_driver_02,
2072                 ram_enableB => '1',
2073                 write_enableA => write_enable_driver,
2074                 write_enableB => habilita2,
2075                 input_dataA => input_data_driver,
2076                 input_dataB => z02_o,
2077                 ram_outputA => ram_output_driver_02,
2078                 ram_outputB => z02,
2079                 addressA => address_driver,
2080                 addressB => direccion
2081         );
2082
2083 Inst_memoria_z02_1: memoria_prueba
2084                 generic map(
2085                 INIT_00 =>x"44160000",--600
2086                 INIT_01 =>x"44161000",
2087                 INIT_02 =>x"44162000",
2088                 INIT_03 =>x"44163000",
2089                 INIT_04 =>x"44164000",
2090                 INIT_05 =>x"44165000",
2091                 INIT_06 =>x"44166000",
2092                 INIT_07 =>x"44167000",
2093                 INIT_08 =>x"44168000",
2094                 INIT_09 =>x"44169000",
2095                 INIT_0A =>x"4416A000",
2096                 INIT_0B =>x"4416B000",
2097                 INIT_0C =>x"4416C000",
2098                 INIT_0D =>x"4416D000",
```

```
2099                    INIT_0E =>x"4416E000",
2100                    INIT_0F =>x"4416F000",
2101                    ancho_dir => 4,
2102                    ancho_dat => 32
2103                    )
2104                    PORT MAP(
2105                    clockA => clk_driver,
2106                    clockB => clk,
2107                    ram_enableA => ram_enable_driver_02_1,
2108                    ram_enableB => '1',
2109                    write_enableA => write_enable_driver,
2110                    write_enableB => habilita2,
2111                    input_dataA => input_data_driver,
2112                    input_dataB => z02_1_o,
2113                    ram_outputA => ram_output_driver_02_1,
2114                    ram_outputB => z02_1,
2115                    addressA => address_driver,
2116                    addressB => direccion
2117            );
2118
2119    Inst_memoria_z03: memoria_prueba
2120                    generic map(
2121                    INIT_00 =>x"442F0000",--700
2122                    INIT_01 =>x"442F1000",
2123                    INIT_02 =>x"442F2000",
2124                    INIT_03 =>x"442F3000",
2125                    INIT_04 =>x"442F4000",
2126                    INIT_05 =>x"442F5000",
2127                    INIT_06 =>x"442F6000",
2128                    INIT_07 =>x"442F7000",
2129                    INIT_08 =>x"442F8000",
2130                    INIT_09 =>x"442F9000",
2131                    INIT_0A =>x"442FA000",
2132                    INIT_0B =>x"442FB000",
2133                    INIT_0C =>x"442FC000",
2134                    INIT_0D =>x"442FD000",
2135                    INIT_0E =>x"442FE000",
2136                    INIT_0F =>x"442FF000",
2137                    ancho_dir => 4,
2138                    ancho_dat => 32
2139                    )
2140                    PORT MAP(
2141                    clockA => clk_driver,
2142                    clockB => clk,
2143                    ram_enableA => ram_enable_driver_03,
2144                    ram_enableB => '1',
2145                    write_enableA => write_enable_driver,
2146                    write_enableB => habilita2,
2147                    input_dataA => input_data_driver,
2148                    input_dataB => z03_o,
2149                    ram_outputA => ram_output_driver_03,
2150                    ram_outputB => z03,
2151                    addressA => address_driver,
```

```
2152                 addressB => direccion
2153         );
2154
2155 Inst_memoria_z03_1: memoria_prueba
2156                 generic map(
2157                 INIT_00 =>x"44480000",--800
2158                 INIT_01 =>x"44481000",
2159                 INIT_02 =>x"44482000",
2160                 INIT_03 =>x"44483000",
2161                 INIT_04 =>x"44484000",
2162                 INIT_05 =>x"44485000",
2163                 INIT_06 =>x"44486000",
2164                 INIT_07 =>x"44487000",
2165                 INIT_08 =>x"44488000",
2166                 INIT_09 =>x"44489000",
2167                 INIT_0A =>x"4448A000",
2168                 INIT_0B =>x"4448B000",
2169                 INIT_0C =>x"4448C000",
2170                 INIT_0D =>x"4448D000",
2171                 INIT_0E =>x"4448E000",
2172                 INIT_0F =>x"4448F000",
2173                 ancho_dir => 4,
2174                 ancho_dat => 32
2175                 )
2176                 PORT MAP(
2177                 clockA => clk_driver,
2178                 clockB => clk,
2179                 ram_enableA => ram_enable_driver_03_1,
2180                 ram_enableB => '1',
2181                 write_enableA => write_enable_driver,
2182                 write_enableB => habilita2,
2183                 input_dataA => input_data_driver,
2184                 input_dataB => z03_1_o,
2185                 ram_outputA => ram_output_driver_03_1,
2186                 ram_outputB => z03_1,
2187                 addressA => address_driver,
2188                 addressB => direccion
2189         );
2190
2191 Inst_memoria_z04: memoria_prueba
2192                 generic map(
2193                 INIT_00 =>x"44610000",--900
2194                 INIT_01 =>x"44611000",
2195                 INIT_02 =>x"44612000",
2196                 INIT_03 =>x"44613000",
2197                 INIT_04 =>x"44614000",
2198                 INIT_05 =>x"44615000",
2199                 INIT_06 =>x"44616000",
2200                 INIT_07 =>x"44617000",
2201                 INIT_08 =>x"44618000",
2202                 INIT_09 =>x"44619000",
2203                 INIT_0A =>x"4461A000",
2204                 INIT_0B =>x"4461B000",
```

```
2205                    INIT_0C =>x"4461C000",
2206                    INIT_0D =>x"4461D000",
2207                    INIT_0E =>x"4461E000",
2208                    INIT_0F =>x"4461F000",
2209                    ancho_dir => 4,
2210                    ancho_dat => 32
2211                    )
2212                    PORT MAP(
2213                    clockA => clk_driver,
2214                    clockB => clk,
2215                    ram_enableA => ram_enable_driver_04,
2216                    ram_enableB => '1',
2217                    write_enableA => write_enable_driver,
2218                    write_enableB => habilita2,
2219                    input_dataA => input_data_driver,
2220                    input_dataB => z04_o,
2221                    ram_outputA => ram_output_driver_04,
2222                    ram_outputB => z04,
2223                    addressA => address_driver,
2224                    addressB => direccion
2225            );
2226
2227    Inst_memoria_z04_1: memoria_prueba
2228                    generic map(
2229                    INIT_00 =>x"447A0000",--1000
2230                    INIT_01 =>x"447A1000",
2231                    INIT_02 =>x"447A2000",
2232                    INIT_03 =>x"447A3000",
2233                    INIT_04 =>x"447A4000",
2234                    INIT_05 =>x"447A5000",
2235                    INIT_06 =>x"447A6000",
2236                    INIT_07 =>x"447A7000",
2237                    INIT_08 =>x"447A8000",
2238                    INIT_09 =>x"447A9000",
2239                    INIT_0A =>x"447AA000",
2240                    INIT_0B =>x"447AB000",
2241                    INIT_0C =>x"447AC000",
2242                    INIT_0D =>x"447AD000",
2243                    INIT_0E =>x"447AE000",
2244                    INIT_0F =>x"447AF000",
2245                    ancho_dir => 4,
2246                    ancho_dat => 32
2247                    )
2248                    PORT MAP(
2249                    clockA => clk_driver,
2250                    clockB => clk,
2251                    ram_enableA => ram_enable_driver_04_1,
2252                    ram_enableB => '1',
2253                    write_enableA => write_enable_driver,
2254                    write_enableB => habilita2,
2255                    input_dataA => input_data_driver,
2256                    input_dataB => z04_1_o,
2257                    ram_outputA => ram_output_driver_04_1,
```

```
2258                    ram_outputB => z04_1,
2259                    addressA => address_driver,
2260                    addressB => direccion
2261            );
2262
2263    Inst_memoria_z05: memoria_prueba
2264                    generic map(
2265                    INIT_00 =>x"44898000",--1100
2266                    INIT_01 =>x"44898100",
2267                    INIT_02 =>x"44898200",
2268                    INIT_03 =>x"44898300",
2269                    INIT_04 =>x"44898400",
2270                    INIT_05 =>x"44898500",
2271                    INIT_06 =>x"44898600",
2272                    INIT_07 =>x"44898700",
2273                    INIT_08 =>x"44898800",
2274                    INIT_09 =>x"44898900",
2275                    INIT_0A =>x"44898A00",
2276                    INIT_0B =>x"44898B00",
2277                    INIT_0C =>x"44898C00",
2278                    INIT_0D =>x"44898D00",
2279                    INIT_0E =>x"44898E00",
2280                    INIT_0F =>x"44898F00",
2281                    ancho_dir => 4,
2282                    ancho_dat => 32
2283                    )
2284                    PORT MAP(
2285                    clockA => clk_driver,
2286                    clockB => clk,
2287                    ram_enableA => ram_enable_driver_05,
2288                    ram_enableB => '1',
2289                    write_enableA => write_enable_driver,
2290                    write_enableB => habilita3,
2291                    input_dataA => input_data_driver,
2292                    input_dataB => z05_o,
2293                    ram_outputA => ram_output_driver_05,
2294                    ram_outputB => z05,
2295                    addressA => address_driver,
2296                    addressB => direccion
2297            );
2298
2299    Inst_memoria_z05_1: memoria_prueba
2300                    generic map(
2301                    INIT_00 =>x"44960000",--1200
2302                    INIT_01 =>x"44961000",
2303                    INIT_02 =>x"44962000",
2304                    INIT_03 =>x"44963000",
2305                    INIT_04 =>x"44964000",
2306                    INIT_05 =>x"44965000",
2307                    INIT_06 =>x"44966000",
2308                    INIT_07 =>x"44967000",
2309                    INIT_08 =>x"44968000",
2310                    INIT_09 =>x"44969000",
```

```
2311                    INIT_0A =>x"4496A000",
2312                    INIT_0B =>x"4496B000",
2313                    INIT_0C =>x"4496C000",
2314                    INIT_0D =>x"4496D000",
2315                    INIT_0E =>x"4496E000",
2316                    INIT_0F =>x"4496F000",
2317                    ancho_dir => 4,
2318                    ancho_dat => 32
2319                    )
2320                    PORT MAP(
2321                    clockA => clk_driver,
2322                    clockB => clk,
2323                    ram_enableA => ram_enable_driver_05_1,
2324                    ram_enableB => '1',
2325                    write_enableA => write_enable_driver,
2326                    write_enableB => habilita3,
2327                    input_dataA => input_data_driver,
2328                    input_dataB => z05_1_o,
2329                    ram_outputA => ram_output_driver_05_1,
2330                    ram_outputB => z05_1,
2331                    addressA => address_driver,
2332                    addressB => direccion
2333            );
2334
2335    Inst_memoria_z06: memoria_prueba
2336                    generic map(
2337                    INIT_00 =>x"44A28000",--1300
2338                    INIT_01 =>x"44A28100",
2339                    INIT_02 =>x"44A28200",
2340                    INIT_03 =>x"44A28300",
2341                    INIT_04 =>x"44A28400",
2342                    INIT_05 =>x"44A28500",
2343                    INIT_06 =>x"44A28600",
2344                    INIT_07 =>x"44A28700",
2345                    INIT_08 =>x"44A28800",
2346                    INIT_09 =>x"44A28900",
2347                    INIT_0A =>x"44A28A00",
2348                    INIT_0B =>x"44A28B00",
2349                    INIT_0C =>x"44A28C00",
2350                    INIT_0D =>x"44A28D00",
2351                    INIT_0E =>x"44A28E00",
2352                    INIT_0F =>x"44A28F00",
2353                    ancho_dir => 4,
2354                    ancho_dat => 32
2355                    )
2356                    PORT MAP(
2357                    clockA => clk_driver,
2358                    clockB => clk,
2359                    ram_enableA => ram_enable_driver_06,
2360                    ram_enableB => '1',
2361                    write_enableA => write_enable_driver,
2362                    write_enableB => habilita3,
2363                    input_dataA => input_data_driver,
```

```
2364                    input_dataB => z06_o,
2365                    ram_outputA => ram_output_driver_06,
2366                    ram_outputB => z06,
2367                    addressA => address_driver,
2368                    addressB => direccion
2369            );
2370
2371    Inst_memoria_z06_1: memoria_prueba
2372                    generic map(
2373                    INIT_00 =>x"44AF0000",--1400
2374                    INIT_01 =>x"44AF1000",
2375                    INIT_02 =>x"44AF2000",
2376                    INIT_03 =>x"44AF3000",
2377                    INIT_04 =>x"44AF4000",
2378                    INIT_05 =>x"44AF5000",
2379                    INIT_06 =>x"44AF6000",
2380                    INIT_07 =>x"44AF7000",
2381                    INIT_08 =>x"44AF8000",
2382                    INIT_09 =>x"44AF9000",
2383                    INIT_0A =>x"44AFA000",
2384                    INIT_0B =>x"44AFB000",
2385                    INIT_0C =>x"44AFC000",
2386                    INIT_0D =>x"44AFD000",
2387                    INIT_0E =>x"44AFE000",
2388                    INIT_0F =>x"44AFF000",
2389                    ancho_dir => 4,
2390                    ancho_dat => 32
2391                    )
2392                    PORT MAP(
2393                    clockA => clk_driver,
2394                    clockB => clk,
2395                    ram_enableA => ram_enable_driver_06_1,
2396                    ram_enableB => '1',
2397                    write_enableA => write_enable_driver,
2398                    write_enableB => habilita3,
2399                    input_dataA => input_data_driver,
2400                    input_dataB => z06_1_o,
2401                    ram_outputA => ram_output_driver_06_1,
2402                    ram_outputB => z06_1,
2403                    addressA => address_driver,
2404                    addressB => direccion
2405            );
2406    ---
2407    end Behavioral;
```

G.2 Sección SR1.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
```

```
5  -- Create Date:    07:57:21 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR1 is
31     Port ( clk,reset : in  STD_LOGIC;
32                                 mux7_1_a,mux7_1_b,mux7_1_c,mux7_1_d,mux7_1_e,mux7_1_f,
33                                 mux7_1_g : in std_logic_vector(31 downto 0);
34                                 mux7_2_a,mux7_2_b,mux7_2_c,mux7_2_d,mux7_2_e,mux7_2_f,
35                                 mux7_2_g : in std_logic_vector(31 downto 0);
36                                 salida_m2 : in std_logic_vector(31 downto 0);
37                                 z01 : IN std_logic_vector(31 downto 0);
38                                 z05 : IN std_logic_vector(31 downto 0);
39                                 j5 : IN std_logic_vector(31 downto 0);
40                                 sr29,sr37,sr4,sr12,sr16 : out std_logic_vector(31 downto 0);
41                                 enable_registers_4 : IN std_logic;
42                                 enable_registers_12 : IN std_logic;
43                                 enable_registers_16 : IN std_logic;
44                                 enable_registers_29 : IN std_logic;
45                                 enable_registers_37 : IN std_logic;
46                                 enable_mux7_1,enable_mux7_2 : in std_logic_vector(2 downto 0);
47                                 enable_mux2_1 : IN std_logic;
48                                 enable_mux2_2 : IN std_logic;
49                                 enable_mux2_3 : IN std_logic;
50                                 enable_mux2_4 : IN std_logic;
51                                 operation_sr1 : in std_logic_vector(5 downto 0);
52                                 operation_nd_sr1 : in std_logic;
53                                 operation_rfd_sr1 : out std_logic;
54                                 sclr_sr1 : in std_logic;
55                                 ce_sr1 : in std_logic;
56                                 underflow_sr1 : out std_logic;
57                                 overflow_sr1 : out std_logic;
```

```
58                                  invalid_op_sr1 : out std_logic;
59                                  rdy_sr1 : out std_logic
60                              );
61 end Seccion_SR1;
62
63 architecture Behavioral of Seccion_SR1 is
64
65 COMPONENT mux2
66         PORT(
67                 a : IN std_logic_vector(31 downto 0);
68                 b : IN std_logic_vector(31 downto 0);
69                 control : IN std_logic;
70                 salida : OUT std_logic_vector(31 downto 0)
71                 );
72         END COMPONENT;
73
74 COMPONENT registro
75     generic(
76     valor_inicial : std_logic_vector(31 downto 0)
77     );
78         PORT(
79                 clk : IN std_logic;
80                 reset : IN std_logic;
81                 enable : IN std_logic;
82                 entrada : IN std_logic_vector(31 downto 0);
83                 salida : OUT std_logic_vector(31 downto 0)
84                 );
85         END COMPONENT;
86
87 COMPONENT suma_resta
88         PORT(
89                 a : IN std_logic_vector(31 downto 0);
90                 b : IN std_logic_vector(31 downto 0);
91                 operation : IN std_logic_vector(5 downto 0);
92                 operation_nd : IN std_logic;
93                 clk : IN std_logic;
94                 sclr : IN std_logic;
95                 ce : IN std_logic;
96                 operation_rfd : OUT std_logic;
97                 result : OUT std_logic_vector(31 downto 0);
98                 underflow : OUT std_logic;
99                 overflow : OUT std_logic;
100                invalid_op : OUT std_logic;
101                rdy : OUT std_logic
102                );
103        END COMPONENT;
104
105 COMPONENT mux7
106        PORT(
107                a : IN std_logic_vector(31 downto 0);
108                b : IN std_logic_vector(31 downto 0);
109                c : IN std_logic_vector(31 downto 0);
110                d : IN std_logic_vector(31 downto 0);
```

```
111                 e : IN std_logic_vector(31 downto 0);
112                 f : IN std_logic_vector(31 downto 0);
113                 g : IN std_logic_vector(31 downto 0);
114                 control : IN std_logic_vector(2 downto 0);
115                 salida : OUT std_logic_vector(31 downto 0)
116                 );
117         END COMPONENT;
118
119 signal result_sr1 : std_logic_vector(31 downto 0);
120 signal ir4,ir12,ir37,ir16 : std_logic_vector(31 downto 0);
121 signal sm7_1,sm7_2 : std_logic_vector(31 downto 0);
122
123 begin
124
125 ----------------------------inicio de SR1
126
127 registro_29: registro
128
129         generic map(
130         valor_inicial=>x"41400000"--valor decimal de 12
131         )
132
133         PORT MAP(
134                 clk => clk,
135                 reset => reset,
136                 enable => enable_registers_29,
137                 entrada => result_sr1,
138                 salida => sr29
139         );
140
141 registro_4: registro
142
143         generic map(
144         valor_inicial=>x"41400000"--valor decimal de 12
145         )
146
147         PORT MAP(
148                 clk => clk,
149                 reset => reset,
150                 enable => enable_registers_4,
151                 entrada => ir4,
152                 salida => sr4
153         );
154
155 registro_12: registro
156         generic map(
157         valor_inicial=>x"41400000"--valor decimal de 12
158         )
159         PORT MAP(
160                 clk => clk,
161                 reset => reset,
162                 enable => enable_registers_12,
163                 entrada => ir12,
```

```
164                     salida => sr12
165             );
166
167     mux7_1: mux7 PORT MAP(
168                     a => mux7_1_a,
169                     b => mux7_1_b,
170                     c => mux7_1_c,
171                     d => mux7_1_d,
172                     e => mux7_1_e,
173                     f => mux7_1_f,
174                     g => mux7_1_g,
175                     salida => sm7_1,
176                     control => enable_mux7_1
177             );
178
179
180
181     mux7_2: mux7 PORT MAP(
182                     a => mux7_2_a,
183                     b => mux7_2_b,
184                     c => mux7_2_c,
185                     d => mux7_2_d,
186                     e => mux7_2_e,
187                     f => mux7_2_f,
188                     g => mux7_2_g,
189                     salida => sm7_2,
190                     control => enable_mux7_2
191             );
192
193     suma_resta_1: suma_resta PORT MAP(
194                     a => sm7_1,
195                     b => sm7_2,
196                     operation => operation_sr1,
197                     operation_nd => operation_nd_sr1,
198                     operation_rfd => operation_rfd_sr1,
199                     clk => clk,
200                     sclr => sclr_sr1,
201                     ce => ce_sr1,
202                     result => result_sr1,
203                     underflow => underflow_sr1,
204                     overflow => overflow_sr1,
205                     invalid_op => invalid_op_sr1,
206                     rdy => rdy_sr1
207             );
208
209
210
211     mux2_1: mux2 PORT MAP(
212                     a => result_sr1,
213                     b => salida_m2,
214                     control => enable_mux2_1,
215                     salida => ir37
216             );
```

```
217
218 registro_37: registro
219
220         generic map(
221         valor_inicial=>x"41400000"--valor decimal de 12
222         )
223
224         PORT MAP(
225                 clk => clk,
226                 reset => reset,
227                 enable => enable_registers_37,
228                 entrada => ir37,
229                 salida => sr37
230         );
231
232 mux2_2: mux2 PORT MAP(
233                 a => result_sr1,
234                 b => z01,
235                 control => enable_mux2_2,
236                 salida => ir4
237         );
238
239 mux2_3: mux2 PORT MAP(
240                 a => result_sr1,
241                 b => z05,
242                 control => enable_mux2_3,
243                 salida => ir12
244         );
245
246 mux2_4: mux2 PORT MAP(
247                 a => result_sr1,
248                 b => j5,
249                 control => enable_mux2_4,
250                 salida => ir16
251         );
252
253 registro_16: registro
254
255         generic map(
256         valor_inicial=>x"41400000"--valor decimal de 12
257         )
258
259         PORT MAP(
260                 clk => clk,
261                 reset => reset,
262                 enable => enable_registers_16,
263                 entrada => ir16,
264                 salida => sr16
265         );
266
267 ------------------------------------------fin de SR1
268 end Behavioral;
```

G.3 Sección SR2.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    08:09:20 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR2 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR2 is
31     Port ( clk,reset : in  STD_LOGIC;
32                                 mux8_1_a,mux8_1_b,mux8_1_c,mux8_1_d,mux8_1_e,mux8_1_f,mux8_1_g,
33                                 mux8_1_h : in std_logic_vector(31 downto 0);
34                                 mux7_3_a,mux7_3_b,mux7_3_c,mux7_3_d,mux7_3_e,mux7_3_f,
35                                 mux7_3_g : in std_logic_vector(31 downto 0);
36                                 salida_m4,salida_m2 : in std_logic_vector(31 downto 0);
37                                 z01_1 : IN std_logic_vector(31 downto 0);
38                                 z05_1 : IN std_logic_vector(31 downto 0);
39                                 j4 : IN std_logic_vector(31 downto 0);
40                                 sr30,sr38,sr46,sr5,sr13,sr17 : out std_logic_vector(31 downto 0);
41                                 enable_registers_5  : IN std_logic;
42                                 enable_registers_13 : IN std_logic;
43                                 enable_registers_17 : IN std_logic;
44                                 enable_registers_30 : IN std_logic;
45                                 enable_registers_38 : IN std_logic;
46                                 enable_registers_46 : IN std_logic;
47                                 enable_mux8_1,enable_mux7_3 : in std_logic_vector(2 downto 0);
48                                 enable_mux2_5 : IN std_logic;
49                                 enable_mux2_6 : IN std_logic;
50                                 enable_mux2_7 : IN std_logic;
51                                 enable_mux2_8 : IN std_logic;
```

```
52                                  enable_mux2_9 : IN std_logic;
53                                  operation_sr2 : in std_logic_vector(5 downto 0);
54                                  operation_nd_sr2 : in std_logic;
55                                  operation_rfd_sr2 : out std_logic;
56                                  sclr_sr2 : in std_logic;
57                                  ce_sr2 : in std_logic;
58                                  underflow_sr2 : out std_logic;
59                                  overflow_sr2 : out std_logic;
60                                  invalid_op_sr2 : out std_logic;
61                                  rdy_sr2 : out std_logic
62          );
63 end Seccion_SR2;
64
65 architecture Behavioral of Seccion_SR2 is
66
67 COMPONENT mux2
68         PORT(
69                 a : IN std_logic_vector(31 downto 0);
70                 b : IN std_logic_vector(31 downto 0);
71                 control : IN std_logic;
72                 salida : OUT std_logic_vector(31 downto 0)
73                 );
74         END COMPONENT;
75
76 COMPONENT registro
77     generic(
78     valor_inicial : std_logic_vector(31 downto 0)
79     );
80         PORT(
81                 clk : IN std_logic;
82                 reset : IN std_logic;
83                 enable : IN std_logic;
84                 entrada : IN std_logic_vector(31 downto 0);
85                 salida : OUT std_logic_vector(31 downto 0)
86                 );
87         END COMPONENT;
88
89 COMPONENT suma_resta
90         PORT(
91                 a : IN std_logic_vector(31 downto 0);
92                 b : IN std_logic_vector(31 downto 0);
93                 operation : IN std_logic_vector(5 downto 0);
94                 operation_nd : IN std_logic;
95                 clk : IN std_logic;
96                 sclr : IN std_logic;
97                 ce : IN std_logic;
98                 operation_rfd : OUT std_logic;
99                 result : OUT std_logic_vector(31 downto 0);
100                underflow : OUT std_logic;
101                overflow : OUT std_logic;
102                invalid_op : OUT std_logic;
103                rdy : OUT std_logic
104                );
```

```
105         END COMPONENT;
106
107 COMPONENT mux8
108         PORT(
109                 a : IN std_logic_vector(31 downto 0);
110                 b : IN std_logic_vector(31 downto 0);
111                 c : IN std_logic_vector(31 downto 0);
112                 d : IN std_logic_vector(31 downto 0);
113                 e : IN std_logic_vector(31 downto 0);
114                 f : IN std_logic_vector(31 downto 0);
115                 g : IN std_logic_vector(31 downto 0);
116                 h : IN std_logic_vector(31 downto 0);
117                 control : IN std_logic_vector(2 downto 0);
118                 salida : OUT std_logic_vector(31 downto 0)
119                 );
120         END COMPONENT;
121
122 COMPONENT mux7
123         PORT(
124                 a : IN std_logic_vector(31 downto 0);
125                 b : IN std_logic_vector(31 downto 0);
126                 c : IN std_logic_vector(31 downto 0);
127                 d : IN std_logic_vector(31 downto 0);
128                 e : IN std_logic_vector(31 downto 0);
129                 f : IN std_logic_vector(31 downto 0);
130                 g : IN std_logic_vector(31 downto 0);
131                 control : IN std_logic_vector(2 downto 0);
132                 salida : OUT std_logic_vector(31 downto 0)
133                 );
134         END COMPONENT;
135
136 signal result_sr2 : std_logic_vector(31 downto 0);
137 signal ir38,ir46,ir5,ir13,ir17 : std_logic_vector(31 downto 0);
138 signal sm8_1,sm7_3 : std_logic_vector(31 downto 0);
139
140
141 begin
142
143 registro_30: registro
144
145         generic map(
146         valor_inicial=>x"41400000"--valor decimal de 12
147         )
148
149         PORT MAP(
150                 clk => clk,
151                 reset => reset,
152                 enable => enable_registers_30,
153                 entrada => result_sr2,
154                 salida => sr30
155         );
156
157 registro_38: registro
```

```
158
159         generic map(
160         valor_inicial=>x"41400000"--valor decimal de 12
161         )
162
163         PORT MAP(
164                 clk => clk,
165                 reset => reset,
166                 enable => enable_registers_38,
167                 entrada => ir38,
168                 salida => sr38
169         );
170
171 registro_46: registro
172
173         generic map(
174         valor_inicial=>x"41400000"--valor decimal de 12
175         )
176
177         PORT MAP(
178                 clk => clk,
179                 reset => reset,
180                 enable => enable_registers_46,
181                 entrada => ir46,
182                 salida =>sr46
183         );
184
185 registro_5: registro
186
187         generic map(
188         valor_inicial=>x"41400000"--valor decimal de 12
189         )
190
191         PORT MAP(
192                 clk => clk,
193                 reset => reset,
194                 enable => enable_registers_5,
195                 entrada => ir5,
196                 salida => sr5
197         );
198
199 registro_13: registro
200
201         generic map(
202         valor_inicial=>x"41400000"--valor decimal de 12
203         )
204
205         PORT MAP(
206                 clk => clk,
207                 reset => reset,
208                 enable => enable_registers_13,
209                 entrada => ir13,
210                 salida => sr13
```

```
211         );
212
213 registro_17: registro
214
215         generic map(
216         valor_inicial=>x"41400000"--valor decimal de 12
217         )
218
219         PORT MAP(
220                 clk => clk,
221                 reset => reset,
222                 enable => enable_registers_17,
223                 entrada => ir17,
224                 salida => sr17
225         );
226
227 mux8_1: mux8 PORT MAP(
228                 a => mux8_1_a,
229                 b => mux8_1_b,
230                 c => mux8_1_c,
231                 d => mux8_1_d,
232                 e => mux8_1_e,
233                 f => mux8_1_f,
234                 g => mux8_1_g,
235                 h => mux8_1_h,
236                 control => enable_mux8_1,
237                 salida => sm8_1
238         );
239
240 mux7_3: mux7 PORT MAP(
241                 a => mux7_3_a,
242                 b => mux7_3_b,
243                 c => mux7_3_c,
244                 d => mux7_3_d,
245                 e => mux7_3_e,
246                 f => mux7_3_f,
247                 g => mux7_3_g,
248                 control => enable_mux7_3,
249                 salida => sm7_3
250         );
251
252 suma_resta_2: suma_resta PORT MAP(
253                 a => sm8_1,
254                 b => sm7_3,
255                 operation => operation_sr2,
256                 operation_nd => operation_nd_sr2,
257                 operation_rfd => operation_rfd_sr2,
258                 clk => clk,
259                 sclr => sclr_sr2,
260                 ce => ce_sr2,
261                 result => result_sr2,
262                 underflow => underflow_sr2,
263                 overflow => overflow_sr2,
```

```
264                 invalid_op => invalid_op_sr2,
265                 rdy => rdy_sr2
266         );
267
268 mux2_5: mux2 PORT MAP(
269                 a => result_sr2,
270                 b => salida_m4,
271                 control => enable_mux2_5,
272                 salida => ir38
273         );
274
275 mux2_6: mux2 PORT MAP(
276                 a => result_sr2,
277                 b => salida_m2,
278                 control => enable_mux2_6,
279                 salida => ir46
280         );
281
282 mux2_7: mux2 PORT MAP(
283                 a => result_sr2,
284                 b => z01_1,
285                 control => enable_mux2_7,
286                 salida => ir5
287         );
288
289 mux2_8: mux2 PORT MAP(
290                 a => result_sr2,
291                 b => z05_1,
292                 control => enable_mux2_8,
293                 salida => ir13
294         );
295
296 mux2_9: mux2 PORT MAP(
297                 a => result_sr2,
298                 b => j4,
299                 control => enable_mux2_9,
300                 salida => ir17
301         );
302
303 end Behavioral;
```

G.4 Sección SR3.vhd

```
1 ----------------------------------------------------------------------------------
2 -- Company:
3 -- Engineer:
4 --
5 -- Create Date:    08:43:20 08/13/2010
6 -- Design Name:
7 -- Module Name:    Seccion_SR3 - Behavioral
8 -- Project Name:
```

```
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR3 is
31     Port ( clk,reset : in  STD_LOGIC;
32                                 mux11_1_a,mux11_1_b,mux11_1_c,mux11_1_d,mux11_1_e,mux11_1_f,
33                                 mux11_1_g,mux11_1_h,mux11_1_i,mux11_1_j,
34                                 mux11_1_k : in std_logic_vector(31 downto 0);
35                                 mux11_2_a,mux11_2_b,mux11_2_c,mux11_2_d,mux11_2_e,mux11_2_f,
36                                 mux11_2_g,mux11_2_h,mux11_2_i,mux11_2_j,
37                                 mux11_2_k : in std_logic_vector(31 downto 0);
38                                 salida_m1 : in std_logic_vector(31 downto 0);
39                                 z02 : IN std_logic_vector(31 downto 0);
40                                 z06 : IN std_logic_vector(31 downto 0);
41                                 j3 : IN std_logic_vector(31 downto 0);
42                                 sr31,sr39,sr47,sr6,sr14,sr18,
43                                 sr45 : out std_logic_vector(31 downto 0);
44                                 enable_registers_6  : IN std_logic;
45                                 enable_registers_14  : IN std_logic;
46                                 enable_registers_18  : IN std_logic;
47                                 enable_registers_31  : IN std_logic;
48                                 enable_registers_39  : IN std_logic;
49                                 enable_registers_47  : IN std_logic;
50                                 enable_registers_45  : IN std_logic;
51                                 enable_mux11_1,
52                                 enable_mux11_2 : in std_logic_vector(3 downto 0);
53                                 enable_mux2_10 : IN std_logic;
54                                 enable_mux2_11 : IN std_logic;
55                                 enable_mux2_12 : IN std_logic;
56                                 enable_mux2_13 : IN std_logic;
57                                 enable_mux2_14 : IN std_logic;
58                                 operation_sr3 : in std_logic_vector(5 downto 0);
59                                 operation_nd_sr3 : in std_logic;
60                                 operation_rfd_sr3 : out std_logic;
61                                 sclr_sr3 : in std_logic;
```

```
62                                  ce_sr3 : in std_logic;
63                                  underflow_sr3 : out std_logic;
64                                  overflow_sr3 : out std_logic;
65                                  invalid_op_sr3 : out std_logic;
66                                  rdy_sr3 : out std_logic
67          );
68  end Seccion_SR3;
69
70  architecture Behavioral of Seccion_SR3 is
71
72  COMPONENT mux2
73          PORT(
74                  a : IN std_logic_vector(31 downto 0);
75                  b : IN std_logic_vector(31 downto 0);
76                  control : IN std_logic;
77                  salida : OUT std_logic_vector(31 downto 0)
78                  );
79          END COMPONENT;
80
81  COMPONENT registro
82      generic(
83      valor_inicial : std_logic_vector(31 downto 0)
84      );
85          PORT(
86                  clk : IN std_logic;
87                  reset : IN std_logic;
88                  enable : IN std_logic;
89                  entrada : IN std_logic_vector(31 downto 0);
90                  salida : OUT std_logic_vector(31 downto 0)
91                  );
92          END COMPONENT;
93
94  COMPONENT suma_resta
95          PORT(
96                  a : IN std_logic_vector(31 downto 0);
97                  b : IN std_logic_vector(31 downto 0);
98                  operation : IN std_logic_vector(5 downto 0);
99                  operation_nd : IN std_logic;
100                 clk : IN std_logic;
101                 sclr : IN std_logic;
102                 ce : IN std_logic;
103                 operation_rfd : OUT std_logic;
104                 result : OUT std_logic_vector(31 downto 0);
105                 underflow : OUT std_logic;
106                 overflow : OUT std_logic;
107                 invalid_op : OUT std_logic;
108                 rdy : OUT std_logic
109                 );
110         END COMPONENT;
111
112 COMPONENT mux11
113         PORT(
114                 a : IN std_logic_vector(31 downto 0);
```

```
115                 b : IN std_logic_vector(31 downto 0);
116                 c : IN std_logic_vector(31 downto 0);
117                 d : IN std_logic_vector(31 downto 0);
118                 e : IN std_logic_vector(31 downto 0);
119                 f : IN std_logic_vector(31 downto 0);
120                 g : IN std_logic_vector(31 downto 0);
121                 h : IN std_logic_vector(31 downto 0);
122                 i : IN std_logic_vector(31 downto 0);
123                 j : IN std_logic_vector(31 downto 0);
124                 k : IN std_logic_vector(31 downto 0);
125                 control : IN std_logic_vector(3 downto 0);
126                 salida : OUT std_logic_vector(31 downto 0)
127                 );
128         END COMPONENT;
129
130 signal result_sr33 : std_logic_vector(31 downto 0);
131 signal ir47,ir6,ir14,ir18,ir45 : std_logic_vector(31 downto 0);
132 signal sm11_1,sm11_2 : std_logic_vector(31 downto 0);
133
134 begin
135
136 registro_31: registro
137         generic map(
138         valor_inicial=>x"41400000"--valor decimal de 12
139         )
140         PORT MAP(
141                 clk => clk,
142                 reset => reset,
143                 enable => enable_registers_31,
144                 entrada => result_sr33,
145                 salida => sr31
146         );
147
148 registro_39: registro
149         generic map(
150         valor_inicial=>x"41400000"--valor decimal de 12
151         )
152         PORT MAP(
153                 clk => clk,
154                 reset => reset,
155                 enable => enable_registers_39,
156                 entrada => result_sr33,
157                 salida => sr39
158         );
159
160 registro_47: registro
161
162         generic map(
163         valor_inicial=>x"41400000"--valor decimal de 12
164         )
165
166         PORT MAP(
167                 clk => clk,
```

```
168                 reset => reset,
169                 enable => enable_registers_47,
170                 entrada => ir47,
171                 salida => sr47
172         );
173
174 registro_6: registro
175         generic map(
176         valor_inicial=>x"40C00000"--valor decimal de 6
177         )
178         PORT MAP(
179                 clk => clk,
180                 reset => reset,
181                 enable => enable_registers_6,
182                 entrada => ir6,
183                 salida => sr6
184         );
185
186 registro_14: registro
187         generic map(
188         valor_inicial=>x"41400000"--valor decimal de 12
189         )
190         PORT MAP(
191                 clk => clk,
192                 reset => reset,
193                 enable => enable_registers_14,
194                 entrada => ir14,
195                 salida => sr14
196         );
197
198 registro_18: registro
199         generic map(
200         valor_inicial=>x"41400000"--valor decimal de 12
201         )
202         PORT MAP(
203                 clk => clk,
204                 reset => reset,
205                 enable => enable_registers_18,
206                 entrada => ir18,
207                 salida => sr18
208         );
209
210 registro_45: registro
211         generic map(
212         valor_inicial=>x"42340000"--valor decimal de 45
213         )
214         PORT MAP(
215                 clk => clk,
216                 reset => reset,
217                 enable => enable_registers_45,
218                 entrada => ir45,
219                 salida => sr45
220         );
```

```
221
222 mux11_1: mux11 PORT MAP(
223                     a => mux11_1_a,
224                     b => mux11_1_b,
225                     c => mux11_1_c,
226                     d => mux11_1_d,
227                     e => mux11_1_e,
228                     f => mux11_1_f,
229                     g => mux11_1_g,
230                     h => mux11_1_h,
231                     i => mux11_1_i,
232                     j => mux11_1_j,
233                     k => mux11_1_k,
234                     salida => sm11_1,
235                     control => enable_mux11_1
236           );
237
238 mux11_2: mux11 PORT MAP(
239                     a => mux11_2_a,
240             b => mux11_2_b,
241                     c => mux11_2_c,
242                     d => mux11_2_d,
243                     e => mux11_2_e,
244                     f => mux11_2_f,
245                     g => mux11_2_g,
246                     h => mux11_2_h,
247                     i => mux11_2_i,
248                     j => mux11_2_j,
249                     k => mux11_2_k,
250                     salida => sm11_2,
251                     control => enable_mux11_2
252           );
253
254 suma_resta_3: suma_resta PORT MAP(
255                     a => sm11_1,
256                     b => sm11_2,
257                     operation => operation_sr3,
258                     operation_nd => operation_nd_sr3,
259                     operation_rfd => operation_rfd_sr3,
260                     clk => clk,
261                     sclr => sclr_sr3,
262                     ce => ce_sr3,
263                     result => result_sr33,
264                     underflow => underflow_sr3,
265                     overflow => overflow_sr3,
266                     invalid_op => invalid_op_sr3,
267                     rdy => rdy_sr3
268           );
269
270 mux2_10: mux2 PORT MAP(
271                     a => result_sr33,
272                     b => salida_m1,
273                     control => enable_mux2_10,
```

```
274                     salida => ir47
275             );
276
277     mux2_11: mux2 PORT MAP(
278                     a => result_sr33,
279                     b => z02,
280                     control => enable_mux2_11,
281                     salida => ir6
282             );
283
284     mux2_12: mux2 PORT MAP(
285                     a => result_sr33,
286                     b => z06,
287                     control => enable_mux2_12,
288                     salida => ir14
289             );
290
291     mux2_13: mux2 PORT MAP(
292                     a => result_sr33,
293                     b => j3,
294                     control => enable_mux2_13,
295                     salida => ir18
296             );
297
298     mux2_14: mux2 PORT MAP(
299                     a => result_sr33,
300                     b => salida_m1,
301                     control => enable_mux2_14,
302                     salida => ir45
303             );
304
305
306     end Behavioral;
```

G.5 Sección SR4.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    10:24:17 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR4 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
```

```
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR4 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux9_1_a,mux9_1_b,mux9_1_c,mux9_1_d,mux9_1_e,mux9_1_f,mux9_1_g,mux9_1_h,
33 mux9_1_i : in std_logic_vector(31 downto 0);
34 mux9_2_a,mux9_2_b,mux9_2_c,mux9_2_d,mux9_2_e,mux9_2_f,mux9_2_g,mux9_2_h,
35 mux9_2_i : in std_logic_vector(31 downto 0);
36 salida_m2,salida_m3 : in std_logic_vector(31 downto 0);
37 z02_1 : IN std_logic_vector(31 downto 0);
38 z06_1 : IN std_logic_vector(31 downto 0);
39 j2 : IN std_logic_vector(31 downto 0);
40 sr40,sr48,sr7,sr15,sr19,sr32 : out std_logic_vector(31 downto 0);
41 enable_registers_7  : IN std_logic;
42 enable_registers_15  : IN std_logic;
43 enable_registers_19  : IN std_logic;
44 enable_registers_32  : IN std_logic;
45 enable_registers_40  : IN std_logic;
46 enable_registers_48  : IN std_logic;
47 enable_mux9_1,enable_mux9_2 : in std_logic_vector(3 downto 0);
48 enable_mux2_15 : IN std_logic;
49 enable_mux2_16 : IN std_logic;
50 enable_mux2_17 : IN std_logic;
51 enable_mux2_18 : IN std_logic;
52 enable_mux2_19 : IN std_logic;
53 operation_sr4 : in std_logic_vector(5 downto 0);
54 operation_nd_sr4 : in std_logic;
55 operation_rfd_sr4 : out std_logic;
56 sclr_sr4 : in std_logic;
57 ce_sr4 : in std_logic;
58 underflow_sr4 : out std_logic;
59 overflow_sr4 : out std_logic;
60 invalid_op_sr4 : out std_logic;
61 rdy_sr4 : out std_logic
62 );
63 end Seccion_SR4;
64
65 architecture Behavioral of Seccion_SR4 is
66
67 COMPONENT mux2
68         PORT(
```

```
69                  a : IN std_logic_vector(31 downto 0);
70                  b : IN std_logic_vector(31 downto 0);
71                  control : IN std_logic;
72                  salida : OUT std_logic_vector(31 downto 0)
73                  );
74          END COMPONENT;
75
76  COMPONENT registro
77      generic(
78      valor_inicial : std_logic_vector(31 downto 0)
79      );
80          PORT(
81                  clk : IN std_logic;
82                  reset : IN std_logic;
83                  enable : IN std_logic;
84                  entrada : IN std_logic_vector(31 downto 0);
85                  salida : OUT std_logic_vector(31 downto 0)
86                  );
87          END COMPONENT;
88
89  COMPONENT suma_resta
90          PORT(
91                  a : IN std_logic_vector(31 downto 0);
92                  b : IN std_logic_vector(31 downto 0);
93                  operation : IN std_logic_vector(5 downto 0);
94                  operation_nd : IN std_logic;
95                  clk : IN std_logic;
96                  sclr : IN std_logic;
97                  ce : IN std_logic;
98                  operation_rfd : OUT std_logic;
99                  result : OUT std_logic_vector(31 downto 0);
100                 underflow : OUT std_logic;
101                 overflow : OUT std_logic;
102                 invalid_op : OUT std_logic;
103                 rdy : OUT std_logic
104                 );
105         END COMPONENT;
106
107 COMPONENT mux9
108         PORT(
109                 a : IN std_logic_vector(31 downto 0);
110                 b : IN std_logic_vector(31 downto 0);
111                 c : IN std_logic_vector(31 downto 0);
112                 d : IN std_logic_vector(31 downto 0);
113                 e : IN std_logic_vector(31 downto 0);
114                 f : IN std_logic_vector(31 downto 0);
115                 g : IN std_logic_vector(31 downto 0);
116                 h : IN std_logic_vector(31 downto 0);
117                 i : IN std_logic_vector(31 downto 0);
118                 control : IN std_logic_vector(3 downto 0);
119                 salida : OUT std_logic_vector(31 downto 0)
120                 );
121         END COMPONENT;
```

```
122
123 signal result_sr4 : std_logic_vector(31 downto 0);
124 signal ir40,ir48,ir7,ir15,ir19 : std_logic_vector(31 downto 0);
125 signal sm9_1,sm9_2 : std_logic_vector(31 downto 0);
126
127 begin
128
129 registro_32: registro
130
131         generic map(
132         valor_inicial=>x"41400000"--valor decimal de 12
133         )
134
135         PORT MAP(
136                 clk => clk,
137                 reset => reset,
138                 enable => enable_registers_32,
139                 entrada => result_sr4,
140                 salida => sr32
141         );
142
143 registro_40: registro
144
145         generic map(
146         valor_inicial=>x"41400000"--valor decimal de 12
147         )
148
149         PORT MAP(
150                 clk => clk,
151                 reset => reset,
152                 enable => enable_registers_40,
153                 entrada => ir40,
154                 salida => sr40
155         );
156
157 registro_48: registro
158
159         generic map(
160         valor_inicial=>x"41400000"--valor decimal de 12
161         )
162
163         PORT MAP(
164                 clk => clk,
165                 reset => reset,
166                 enable => enable_registers_48,
167                 entrada => ir48,
168                 salida => sr48
169         );
170
171 registro_7: registro
172
173         generic map(
174         valor_inicial=>x"41400000"--valor decimal de 12
```

```
175         )
176
177         PORT MAP(
178                 clk => clk,
179                 reset => reset,
180                 enable => enable_registers_7,
181                 entrada => ir7,
182                 salida => sr7
183         );
184
185 registro_15: registro
186
187         generic map(
188         valor_inicial=>x"41400000"--valor decimal de 12
189         )
190
191         PORT MAP(
192                 clk => clk,
193                 reset => reset,
194                 enable => enable_registers_15,
195                 entrada => ir15,
196                 salida => sr15
197         );
198
199 registro_19: registro
200
201         generic map(
202         valor_inicial=>x"41400000"--valor decimal de 12
203         )
204
205         PORT MAP(
206                 clk => clk,
207                 reset => reset,
208                 enable => enable_registers_19,
209                 entrada => ir19,
210                 salida => sr19
211         );
212
213 mux9_1: mux9 PORT MAP(
214                 a => mux9_1_a,
215                 b => mux9_1_b,
216                 c => mux9_1_c,
217                 d => mux9_1_d,
218                 e => mux9_1_e,
219                 f => mux9_1_f,
220                 g => mux9_1_g,
221                 h => mux9_1_h,
222                 i => mux9_1_i,
223                 salida => sm9_1,
224                 control => enable_mux9_1
225         );
226
227 mux9_2: mux9 PORT MAP(
```

```
228                     a => mux9_2_a,
229             b => mux9_2_b,
230                     c => mux9_2_c,
231                     d => mux9_2_d,
232                     e => mux9_2_e,
233                     f => mux9_2_f,
234                     g => mux9_2_g,
235                     h => mux9_2_h,
236                     i => mux9_2_i,
237                     salida => sm9_2,
238                     control => enable_mux9_2
239         );
240
241 suma_resta_4: suma_resta PORT MAP(
242                     a => sm9_1,
243                     b => sm9_2,
244                     operation => operation_sr4,
245                     operation_nd => operation_nd_sr4,
246                     operation_rfd => operation_rfd_sr4,
247                     clk => clk,
248                     sclr => sclr_sr4,
249                     ce => ce_sr4,
250                     result => result_sr4,
251                     underflow => underflow_sr4,
252                     overflow => overflow_sr4,
253                     invalid_op => invalid_op_sr4,
254                     rdy => rdy_sr4
255         );
256
257 mux2_15: mux2 PORT MAP(
258                     a => result_sr4,
259                     b => salida_m3,
260                     control => enable_mux2_15,
261                     salida => ir40
262         );
263
264 mux2_16: mux2 PORT MAP(
265                     a => result_sr4,
266                     b => salida_m2,
267                     control => enable_mux2_16,
268                     salida => ir48
269         );
270
271 mux2_17: mux2 PORT MAP(
272                     a => result_sr4,
273                     b => z02_1,
274                     control => enable_mux2_17,
275                     salida => ir7
276         );
277
278 mux2_18: mux2 PORT MAP(
279                     a => result_sr4,
280                     b => z06_1,
```

```
281                 control => enable_mux2_18,
282                 salida => ir15
283         );
284
285 mux2_19: mux2 PORT MAP(
286                 a => result_sr4,
287                 b => j2,
288                 control => enable_mux2_19,
289                 salida => ir19
290         );
291
292 end Behavioral;
```

G.6 Sección SR5.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:37:23 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR5 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR5 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux9_3_a,mux9_3_b,mux9_3_c,mux9_3_d,mux9_3_e,mux9_3_f,mux9_3_g,
33 mux9_3_h : in std_logic_vector(31 downto 0);
34 mux7_4_a,mux7_4_b,mux7_4_c,mux7_4_d,mux7_4_e,mux7_4_f,
35 mux7_4_g : in std_logic_vector(31 downto 0);
36 salida_m1,salida_m6 : in std_logic_vector(31 downto 0);
```

```
37  z00 : IN std_logic_vector(31 downto 0);
38  z03 : IN std_logic_vector(31 downto 0);
39  j6 : IN std_logic_vector(31 downto 0);
40  j0 : IN std_logic_vector(31 downto 0);
41  sr33,sr41,sr52,sr2,sr8,sr21,sr28_0 : out std_logic_vector(31 downto 0);
42  enable_registers_2  : IN std_logic;
43  enable_registers_8  : IN std_logic;
44  enable_registers_21  : IN std_logic;
45  enable_registers_28  : IN std_logic;
46  enable_registers_33  : IN std_logic;
47  enable_registers_41  : IN std_logic;
48  enable_registers_52  : IN std_logic;
49  enable_mux9_3 : in std_logic_vector(3 downto 0);
50  enable_mux7_4 : in std_logic_vector(2 downto 0);
51  enable_mux2_20 : IN std_logic;
52  enable_mux2_21 : IN std_logic;
53  enable_mux2_22 : IN std_logic;
54  enable_mux2_23 : IN std_logic;
55  enable_mux2_24 : IN std_logic;
56  enable_mux2_25 : IN std_logic;
57  operation_sr5 : in std_logic_vector(5 downto 0);
58  operation_nd_sr5 : in std_logic;
59  operation_rfd_sr5 : out std_logic;
60  sclr_sr5 : in std_logic;
61  ce_sr5 : in std_logic;
62  underflow_sr5 : out std_logic;
63  overflow_sr5 : out std_logic;
64  invalid_op_sr5 : out std_logic;
65  rdy_sr5 : out std_logic
66  );
67  end Seccion_SR5;
68
69  architecture Behavioral of Seccion_SR5 is
70
71  COMPONENT mux2
72          PORT(
73                  a : IN std_logic_vector(31 downto 0);
74                  b : IN std_logic_vector(31 downto 0);
75                  control : IN std_logic;
76                  salida : OUT std_logic_vector(31 downto 0)
77                  );
78          END COMPONENT;
79
80  COMPONENT registro
81      generic(
82      valor_inicial : std_logic_vector(31 downto 0)
83      );
84          PORT(
85                  clk : IN std_logic;
86                  reset : IN std_logic;
87                  enable : IN std_logic;
88                  entrada : IN std_logic_vector(31 downto 0);
89                  salida : OUT std_logic_vector(31 downto 0)
```

```
90                  );
91          END COMPONENT;
92
93  COMPONENT suma_resta
94          PORT(
95                  a : IN std_logic_vector(31 downto 0);
96                  b : IN std_logic_vector(31 downto 0);
97                  operation : IN std_logic_vector(5 downto 0);
98                  operation_nd : IN std_logic;
99                  clk : IN std_logic;
100                 sclr : IN std_logic;
101                 ce : IN std_logic;
102                 operation_rfd : OUT std_logic;
103                 result : OUT std_logic_vector(31 downto 0);
104                 underflow : OUT std_logic;
105                 overflow : OUT std_logic;
106                 invalid_op : OUT std_logic;
107                 rdy : OUT std_logic
108                 );
109         END COMPONENT;
110
111 COMPONENT mux9
112         PORT(
113                 a : IN std_logic_vector(31 downto 0);
114                 b : IN std_logic_vector(31 downto 0);
115                 c : IN std_logic_vector(31 downto 0);
116                 d : IN std_logic_vector(31 downto 0);
117                 e : IN std_logic_vector(31 downto 0);
118                 f : IN std_logic_vector(31 downto 0);
119                 g : IN std_logic_vector(31 downto 0);
120                 h : IN std_logic_vector(31 downto 0);
121                 i : IN std_logic_vector(31 downto 0);
122                 control : IN std_logic_vector(3 downto 0);
123                 salida : OUT std_logic_vector(31 downto 0)
124                 );
125         END COMPONENT;
126
127 COMPONENT mux7
128         PORT(
129                 a : IN std_logic_vector(31 downto 0);
130                 b : IN std_logic_vector(31 downto 0);
131                 c : IN std_logic_vector(31 downto 0);
132                 d : IN std_logic_vector(31 downto 0);
133                 e : IN std_logic_vector(31 downto 0);
134                 f : IN std_logic_vector(31 downto 0);
135                 g : IN std_logic_vector(31 downto 0);
136                 control : IN std_logic_vector(2 downto 0);
137                 salida : OUT std_logic_vector(31 downto 0)
138                 );
139         END COMPONENT;
140
141 signal result_sr5 : std_logic_vector(31 downto 0);
142 signal ir41,ir52,ir2,ir8,ir28,sr28,ir21 : std_logic_vector(31 downto 0);
```

```
143 signal sm9_3,sm7_4 : std_logic_vector(31 downto 0);
144
145 begin
146
147 registro_33: registro
148
149         generic map(
150         valor_inicial=>x"41400000"--valor decimal de 12
151         )
152
153         PORT MAP(
154                 clk => clk,
155                 reset => reset,
156                 enable => enable_registers_33,
157                 entrada => result_sr5,
158                 salida => sr33
159         );
160
161 registro_41: registro
162
163         generic map(
164         valor_inicial=>x"41400000"--valor decimal de 12
165         )
166
167         PORT MAP(
168                 clk => clk,
169                 reset => reset,
170                 enable => enable_registers_41,
171                 entrada => ir41,
172                 salida => sr41
173         );
174
175 registro_52: registro
176
177         generic map(
178         valor_inicial=>x"41400000"--valor decimal de 12
179         )
180
181         PORT MAP(
182                 clk => clk,
183                 reset => reset,
184                 enable => enable_registers_52,
185                 entrada => ir52,
186                 salida => sr52
187         );
188
189 registro_2: registro
190         generic map(
191         valor_inicial=>x"40000000"--valor decimal de 2
192         )
193         PORT MAP(
194                 clk => clk,
195                 reset => reset,
```

```
196                 enable => enable_registers_2,
197                 entrada => ir2,
198                 salida => sr2
199         );
200
201 registro_8: registro
202          generic map(
203         valor_inicial=>x"41000000"--valor decimal de 8
204         )
205         PORT MAP(
206                 clk => clk,
207                 reset => reset,
208                 enable => enable_registers_8,
209                 entrada => ir8,
210                 salida => sr8
211         );
212
213 registro_28: registro
214
215         generic map(
216         valor_inicial=>x"41400000"--valor decimal de 12
217         )
218
219         PORT MAP(
220                 clk => clk,
221                 reset => reset,
222                 enable => enable_registers_28,
223                 entrada => ir28,
224                 salida => sr28
225         );
226
227 registro_21: registro
228
229         generic map(
230         valor_inicial=>x"41400000"--valor decimal de 12
231         )
232
233         PORT MAP(
234                 clk => clk,
235                 reset => reset,
236                 enable => enable_registers_21,
237                 entrada => ir21,
238                 salida => sr21
239         );
240
241 mux9_3: mux9 PORT MAP(
242                 a => mux9_3_a,
243                 b => mux9_3_b,
244                 c => mux9_3_c,
245                 d => mux9_3_d,
246                 e => mux9_3_e,
247                 f => mux9_3_f,
248                 g => mux9_3_g,
```

```
249                     h => mux9_3_h,
250                     i => sr28,
251                     salida => sm9_3,
252                     control => enable_mux9_3
253             );
254
255     mux7_4: mux7 PORT MAP(
256                     a => mux7_4_a,
257                b => mux7_4_b,
258                     c => mux7_4_c,
259                     d => mux7_4_d,
260                     e => mux7_4_e,
261                     f => mux7_4_f,
262                     g => mux7_4_g,
263                     salida => sm7_4,
264                     control => enable_mux7_4
265             );
266
267     suma_resta_5: suma_resta PORT MAP(
268                     a => sm9_3,
269                     b => sm7_4,
270                     operation => operation_sr5,
271                     operation_nd => operation_nd_sr5,
272                     operation_rfd => operation_rfd_sr5,
273                     clk => clk,
274                     sclr => sclr_sr5,
275                     ce => ce_sr5,
276                     result => result_sr5,
277                     underflow => underflow_sr5,
278                     overflow => overflow_sr5,
279                     invalid_op => invalid_op_sr5,
280                     rdy => rdy_sr5
281             );
282
283     mux2_20: mux2 PORT MAP(
284                     a => result_sr5,
285                     b => salida_m1,
286                     control => enable_mux2_20,
287                     salida => ir41
288             );
289
290     mux2_21: mux2 PORT MAP(
291                     a => result_sr5,
292                     b => salida_m6,
293                     control => enable_mux2_21,
294                     salida => ir52
295             );
296
297     mux2_22: mux2 PORT MAP(
298                     a => result_sr5,
299                     b => z00,
300                     control => enable_mux2_22,
301                     salida => ir2
```

```
302         );
303
304 mux2_23: mux2 PORT MAP(
305                 a => result_sr5,
306                 b => z03,
307                 control => enable_mux2_23,
308                 salida => ir8
309         );
310
311 mux2_24: mux2 PORT MAP(
312                 a => result_sr5,
313                 b => j6,
314                 control => enable_mux2_24,
315                 salida => ir28
316         );
317
318 mux2_25: mux2 PORT MAP(
319                 a => sr28,
320                 b => j0,
321                 control => enable_mux2_25,
322                 salida => ir21
323         );
324 sr28_0 <= sr28;
325
326 end Behavioral;
```

G.7 Sección SR6.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    12:18:14 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR6 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR6 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux8_2_a,mux8_2_b,mux8_2_c,mux8_2_d,mux8_2_e,mux8_2_f,mux8_2_g,
33 mux8_2_h : in std_logic_vector(31 downto 0);
34 mux7_5_a,mux7_5_b,mux7_5_c,mux7_5_d,mux7_5_e,mux7_5_f,
35 mux7_5_g : in std_logic_vector(31 downto 0);
36 salida_m2,salida_m7 : in std_logic_vector(31 downto 0);
37 z00_1 : IN std_logic_vector(31 downto 0);
38 z03_1 : IN std_logic_vector(31 downto 0);
39 j1 : IN std_logic_vector(31 downto 0);
40 sr34,sr42,sr53,sr3,sr9,sr20 : out std_logic_vector(31 downto 0);
41 enable_registers_3  : IN std_logic;
42 enable_registers_9  : IN std_logic;
43 enable_registers_20  : IN std_logic;
44 enable_registers_34  : IN std_logic;
45 enable_registers_42  : IN std_logic;
46 enable_registers_53  : IN std_logic;
47 enable_mux8_2,enable_mux7_5 : in std_logic_vector(2 downto 0);
48 enable_mux2_26 : IN std_logic;
49 enable_mux2_27 : IN std_logic;
50 enable_mux2_28 : IN std_logic;
51 enable_mux2_29 : IN std_logic;
52 enable_mux2_30 : IN std_logic;
53 operation_sr6 : in std_logic_vector(5 downto 0);
54 operation_nd_sr6 : in std_logic;
55 operation_rfd_sr6 : out std_logic;
56 sclr_sr6 : in std_logic;
57 ce_sr6 : in std_logic;
58 underflow_sr6 : out std_logic;
59 overflow_sr6 : out std_logic;
60 invalid_op_sr6 : out std_logic;
61 rdy_sr6 : out std_logic
62 );
63 end Seccion_SR6;
64
65 architecture Behavioral of Seccion_SR6 is
66
67 COMPONENT mux2
68         PORT(
69                 a : IN std_logic_vector(31 downto 0);
70                 b : IN std_logic_vector(31 downto 0);
71                 control : IN std_logic;
72                 salida : OUT std_logic_vector(31 downto 0)
73                 );
74         END COMPONENT;
75
76 COMPONENT registro
```

```
77      generic(
78      valor_inicial : std_logic_vector(31 downto 0)
79      );
80          PORT(
81                  clk : IN std_logic;
82                  reset : IN std_logic;
83                  enable : IN std_logic;
84                  entrada : IN std_logic_vector(31 downto 0);
85                  salida : OUT std_logic_vector(31 downto 0)
86                  );
87          END COMPONENT;
88
89  COMPONENT suma_resta
90          PORT(
91                  a : IN std_logic_vector(31 downto 0);
92                  b : IN std_logic_vector(31 downto 0);
93                  operation : IN std_logic_vector(5 downto 0);
94                  operation_nd : IN std_logic;
95                  clk : IN std_logic;
96                  sclr : IN std_logic;
97                  ce : IN std_logic;
98                  operation_rfd : OUT std_logic;
99                  result : OUT std_logic_vector(31 downto 0);
100                 underflow : OUT std_logic;
101                 overflow : OUT std_logic;
102                 invalid_op : OUT std_logic;
103                 rdy : OUT std_logic
104                 );
105         END COMPONENT;
106
107 COMPONENT mux8
108         PORT(
109                 a : IN std_logic_vector(31 downto 0);
110                 b : IN std_logic_vector(31 downto 0);
111                 c : IN std_logic_vector(31 downto 0);
112                 d : IN std_logic_vector(31 downto 0);
113                 e : IN std_logic_vector(31 downto 0);
114                 f : IN std_logic_vector(31 downto 0);
115                 g : IN std_logic_vector(31 downto 0);
116                 h : IN std_logic_vector(31 downto 0);
117                 control : IN std_logic_vector(2 downto 0);
118                 salida : OUT std_logic_vector(31 downto 0)
119                 );
120         END COMPONENT;
121
122 COMPONENT mux7
123         PORT(
124                 a : IN std_logic_vector(31 downto 0);
125                 b : IN std_logic_vector(31 downto 0);
126                 c : IN std_logic_vector(31 downto 0);
127                 d : IN std_logic_vector(31 downto 0);
128                 e : IN std_logic_vector(31 downto 0);
129                 f : IN std_logic_vector(31 downto 0);
```

```
130                 g : IN std_logic_vector(31 downto 0);
131                 control : IN std_logic_vector(2 downto 0);
132                 salida : OUT std_logic_vector(31 downto 0)
133                 );
134         END COMPONENT;
135
136 signal result_sr6 : std_logic_vector(31 downto 0);
137 signal ir42,ir53,ir3,ir9,ir20 : std_logic_vector(31 downto 0);
138 signal sm8_2,sm7_5 : std_logic_vector(31 downto 0);
139
140 begin
141
142 registro_34: registro
143
144         generic map(
145         valor_inicial=>x"41400000"--valor decimal de 12
146         )
147
148         PORT MAP(
149                 clk => clk,
150                 reset => reset,
151                 enable => enable_registers_34,
152                 entrada => result_sr6,
153                 salida => sr34
154         );
155
156 registro_42: registro
157
158         generic map(
159         valor_inicial=>x"41400000"--valor decimal de 12
160         )
161
162         PORT MAP(
163                 clk => clk,
164                 reset => reset,
165                 enable => enable_registers_42,
166                 entrada => ir42,
167                 salida => sr42
168         );
169
170 registro_53: registro
171
172         generic map(
173         valor_inicial=>x"41400000"--valor decimal de 12
174         )
175
176         PORT MAP(
177                 clk => clk,
178                 reset => reset,
179                 enable => enable_registers_53,
180                 entrada => ir53,
181                 salida => sr53
182         );
```

```
183
184 registro_3: registro
185
186         generic map(
187         valor_inicial=>x"41400000"--valor decimal de 12
188         )
189
190         PORT MAP(
191                 clk => clk,
192                 reset => reset,
193                 enable => enable_registers_3,
194                 entrada => ir3,
195                 salida => sr3
196         );
197
198 registro_9: registro
199
200         generic map(
201         valor_inicial=>x"41400000"--valor decimal de 12
202         )
203
204         PORT MAP(
205                 clk => clk,
206                 reset => reset,
207                 enable => enable_registers_9,
208                 entrada => ir9,
209                 salida => sr9
210         );
211
212 registro_20: registro
213
214         generic map(
215         valor_inicial=>x"41400000"--valor decimal de 12
216         )
217
218         PORT MAP(
219                 clk => clk,
220                 reset => reset,
221                 enable => enable_registers_20,
222                 entrada => ir20,
223                 salida => sr20
224         );
225
226 mux8_2: mux8 PORT MAP(
227                 a => mux8_2_a,
228                 b => mux8_2_b,
229                 c => mux8_2_c,
230                 d => mux8_2_d,
231                 e => mux8_2_e,
232                 f => mux8_2_f,
233                 g => mux8_2_g,
234                 h => mux8_2_h,
235                 control => enable_mux8_2,
```

```
236                     salida => sm8_2
237             );
238
239     mux7_5: mux7 PORT MAP(
240                     a => mux7_5_a,
241                     b => mux7_5_b,
242                     c => mux7_5_c,
243                     d => mux7_5_d,
244                     e => mux7_5_e,
245                     f => mux7_5_f,
246                     g => mux7_5_g,
247                     control => enable_mux7_5,
248                     salida => sm7_5
249             );
250
251     suma_resta_6: suma_resta PORT MAP(
252                     a => sm8_2,
253                     b => sm7_5,
254                     operation => operation_sr6,
255                     operation_nd => operation_nd_sr6,
256                     operation_rfd => operation_rfd_sr6,
257                     clk => clk,
258                     sclr => sclr_sr6,
259                     ce => ce_sr6,
260                     result => result_sr6,
261                     underflow => underflow_sr6,
262                     overflow => overflow_sr6,
263                     invalid_op => invalid_op_sr6,
264                     rdy => rdy_sr6
265             );
266
267     mux2_26: mux2 PORT MAP(
268                     a => result_sr6,
269                     b => salida_m2,
270                     control => enable_mux2_26,
271                     salida => ir42
272             );
273
274     mux2_27: mux2 PORT MAP(
275                     a => result_sr6,
276                     b => salida_m7,
277                     control => enable_mux2_27,
278                     salida => ir53
279             );
280
281     mux2_28: mux2 PORT MAP(
282                     a => result_sr6,
283                     b => z00_1,
284                     control => enable_mux2_28,
285                     salida => ir3
286             );
287
288     mux2_29: mux2 PORT MAP(
```

```
289                     a => result_sr6,
290                     b => z03_1,
291                     control => enable_mux2_29,
292                     salida => ir9
293             );
294 
295     mux2_30: mux2 PORT MAP(
296                     a => result_sr6,
297                     b => j1,
298                     control => enable_mux2_30,
299                     salida => ir20
300             );
301 
302     end Behavioral;
```

G.8 Sección SR7.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    13:05:46 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR7 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24 
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29 
30 entity Seccion_SR7 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux8_3_a,mux8_3_b,mux8_3_c,mux8_3_d,mux8_3_e,mux8_3_f,mux8_3_g,
33 mux8_3_h : in std_logic_vector(31 downto 0);
34 mux8_4_a,mux8_4_b,mux8_4_c,mux8_4_d,mux8_4_e,mux8_4_f,mux8_4_g,
```

```
35 mux8_4_h : in std_logic_vector(31 downto 0);
36 salida_m5 : in std_logic_vector(31 downto 0);
37 z04 : IN std_logic_vector(31 downto 0);
38 sr35,sr43,sr51,sr10 : out std_logic_vector(31 downto 0);
39 enable_registers_10  : IN std_logic;
40 enable_registers_35  : IN std_logic;
41 enable_registers_43  : IN std_logic;
42 enable_registers_51  : IN std_logic;
43 enable_mux8_3,enable_mux8_4 : in std_logic_vector(2 downto 0);
44 enable_mux2_31 : IN std_logic;
45 enable_mux2_32 : IN std_logic;
46 operation_sr7 : in std_logic_vector(5 downto 0);
47 operation_nd_sr7 : in std_logic;
48 operation_rfd_sr7 : out std_logic;
49 sclr_sr7 : in std_logic;
50 ce_sr7 : in std_logic;
51 underflow_sr7 : out std_logic;
52 overflow_sr7 : out std_logic;
53 invalid_op_sr7 : out std_logic;
54 rdy_sr7 : out std_logic
55  );
56 end Seccion_SR7;
57
58 architecture Behavioral of Seccion_SR7 is
59
60 COMPONENT mux2
61         PORT(
62                 a : IN std_logic_vector(31 downto 0);
63                 b : IN std_logic_vector(31 downto 0);
64                 control : IN std_logic;
65                 salida : OUT std_logic_vector(31 downto 0)
66                 );
67         END COMPONENT;
68
69 COMPONENT registro
70     generic(
71     valor_inicial : std_logic_vector(31 downto 0)
72     );
73         PORT(
74                 clk : IN std_logic;
75                 reset : IN std_logic;
76                 enable : IN std_logic;
77                 entrada : IN std_logic_vector(31 downto 0);
78                 salida : OUT std_logic_vector(31 downto 0)
79                 );
80         END COMPONENT;
81
82 COMPONENT suma_resta
83         PORT(
84                 a : IN std_logic_vector(31 downto 0);
85                 b : IN std_logic_vector(31 downto 0);
86                 operation : IN std_logic_vector(5 downto 0);
87                 operation_nd : IN std_logic;
```

```
88                  clk : IN std_logic;
89                  sclr : IN std_logic;
90                  ce : IN std_logic;
91                  operation_rfd : OUT std_logic;
92                  result : OUT std_logic_vector(31 downto 0);
93                  underflow : OUT std_logic;
94                  overflow : OUT std_logic;
95                  invalid_op : OUT std_logic;
96                  rdy : OUT std_logic
97                  );
98          END COMPONENT;
99
100 COMPONENT mux8
101         PORT(
102                 a : IN std_logic_vector(31 downto 0);
103                 b : IN std_logic_vector(31 downto 0);
104                 c : IN std_logic_vector(31 downto 0);
105                 d : IN std_logic_vector(31 downto 0);
106                 e : IN std_logic_vector(31 downto 0);
107                 f : IN std_logic_vector(31 downto 0);
108                 g : IN std_logic_vector(31 downto 0);
109                 h : IN std_logic_vector(31 downto 0);
110                 control : IN std_logic_vector(2 downto 0);
111                 salida : OUT std_logic_vector(31 downto 0)
112                 );
113         END COMPONENT;
114
115 signal result_sr7 : std_logic_vector(31 downto 0);
116 signal ir51,ir10 : std_logic_vector(31 downto 0);
117 signal sm8_3,sm8_4 : std_logic_vector(31 downto 0);
118
119 begin
120
121 registro_35: registro
122
123         generic map(
124         valor_inicial=>x"41400000"--valor decimal de 12
125         )
126
127         PORT MAP(
128                 clk => clk,
129                 reset => reset,
130                 enable => enable_registers_35,
131                 entrada => result_sr7,
132                 salida => sr35
133         );
134
135 registro_43: registro
136
137         generic map(
138         valor_inicial=>x"41400000"--valor decimal de 12
139         )
140
```

```
141         PORT MAP(
142                 clk => clk,
143                 reset => reset,
144                 enable => enable_registers_43,
145                 entrada => result_sr7,
146                 salida => sr43
147         );
148
149 registro_51: registro
150
151         generic map(
152         valor_inicial=>x"41400000"--valor decimal de 12
153         )
154
155         PORT MAP(
156                 clk => clk,
157                 reset => reset,
158                 enable => enable_registers_51,
159                 entrada => ir51,
160                 salida => sr51
161         );
162
163 registro_10: registro
164         generic map(
165         valor_inicial=>x"41200000"--valor decimal de 10
166         )
167         PORT MAP(
168                 clk => clk,
169                 reset => reset,
170                 enable => enable_registers_10,
171                 entrada => ir10,
172                 salida => sr10
173         );
174
175 mux8_3: mux8 PORT MAP(
176                 a => mux8_3_a,
177                 b => mux8_3_b,
178                 c => mux8_3_c,
179                 d => mux8_3_d,
180                 e => mux8_3_e,
181                 f => mux8_3_f,
182                 g => mux8_3_g,
183                 h => mux8_3_h,
184                 control => enable_mux8_3,
185                 salida => sm8_3
186         );
187
188 mux8_4: mux8 PORT MAP(
189                 a => mux8_4_a,
190                 b => mux8_4_b,
191                 c => mux8_4_c,
192                 d => mux8_4_d,
193                 e => mux8_4_e,
```

```
194                     f => mux8_4_f,
195                     g => mux8_4_g,
196                     h => mux8_4_h,
197                     control => enable_mux8_4,
198                     salida => sm8_4
199             );
200 
201     suma_resta_7: suma_resta PORT MAP(
202                     a => sm8_3,
203                     b => sm8_4,
204                     operation => operation_sr7,
205                     operation_nd => operation_nd_sr7,
206                     operation_rfd => operation_rfd_sr7,
207                     clk => clk,
208                     sclr => sclr_sr7,
209                     ce => ce_sr7,
210                     result => result_sr7,
211                     underflow => underflow_sr7,
212                     overflow => overflow_sr7,
213                     invalid_op => invalid_op_sr7,
214                     rdy => rdy_sr7
215             );
216 
217     mux2_31: mux2 PORT MAP(
218                     a => result_sr7,
219                     b => salida_m5,
220                     control => enable_mux2_31,
221                     salida => ir51
222             );
223 
224     mux2_32: mux2 PORT MAP(
225                     a => result_sr7,
226                     b => z04,
227                     control => enable_mux2_32,
228                     salida => ir10
229             );
230 
231     end Behavioral;
```

G.9 Sección SR8.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    15:32:51 08/13/2010
6  -- Design Name:
7  -- Module Name:    Seccion_SR8 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
```

```
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_SR8 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux7_6_a,mux7_6_b,mux7_6_c,mux7_6_d,mux7_6_e,mux7_6_f,
33 mux7_6_g : in std_logic_vector(31 downto 0);
34 mux7_7_a,mux7_7_b,mux7_7_c,mux7_7_d,mux7_7_e,mux7_7_f,
35 mux7_7_g : in std_logic_vector(31 downto 0);
36 z04_1 : IN std_logic_vector(31 downto 0);
37 sr36,sr44,sr11 : out std_logic_vector(31 downto 0);
38 enable_registers_11  : IN std_logic;
39 enable_registers_36  : IN std_logic;
40 enable_registers_44  : IN std_logic;
41 enable_mux7_6,enable_mux7_7 : in std_logic_vector(2 downto 0);
42 enable_mux2_33 : IN std_logic;
43 operation_sr8 : in std_logic_vector(5 downto 0);
44 operation_nd_sr8 : in std_logic;
45 operation_rfd_sr8 : out std_logic;
46 sclr_sr8 : in std_logic;
47 ce_sr8 : in std_logic;
48 underflow_sr8 : out std_logic;
49 overflow_sr8 : out std_logic;
50 invalid_op_sr8 : out std_logic;
51 rdy_sr8 : out std_logic
52 );
53 end Seccion_SR8;
54
55 architecture Behavioral of Seccion_SR8 is
56
57 COMPONENT mux2
58         PORT(
59                 a : IN std_logic_vector(31 downto 0);
60                 b : IN std_logic_vector(31 downto 0);
61                 control : IN std_logic;
62                 salida : OUT std_logic_vector(31 downto 0)
63                 );
```

```
64          END COMPONENT;
65
66  COMPONENT registro
67      generic(
68      valor_inicial : std_logic_vector(31 downto 0)
69      );
70          PORT(
71                  clk : IN std_logic;
72                  reset : IN std_logic;
73                  enable : IN std_logic;
74                  entrada : IN std_logic_vector(31 downto 0);
75                  salida : OUT std_logic_vector(31 downto 0)
76                  );
77          END COMPONENT;
78
79  COMPONENT suma_resta
80          PORT(
81                  a : IN std_logic_vector(31 downto 0);
82                  b : IN std_logic_vector(31 downto 0);
83                  operation : IN std_logic_vector(5 downto 0);
84                  operation_nd : IN std_logic;
85                  clk : IN std_logic;
86                  sclr : IN std_logic;
87                  ce : IN std_logic;
88                  operation_rfd : OUT std_logic;
89                  result : OUT std_logic_vector(31 downto 0);
90                  underflow : OUT std_logic;
91                  overflow : OUT std_logic;
92                  invalid_op : OUT std_logic;
93                  rdy : OUT std_logic
94                  );
95          END COMPONENT;
96
97  COMPONENT mux7
98          PORT(
99                  a : IN std_logic_vector(31 downto 0);
100                 b : IN std_logic_vector(31 downto 0);
101                 c : IN std_logic_vector(31 downto 0);
102                 d : IN std_logic_vector(31 downto 0);
103                 e : IN std_logic_vector(31 downto 0);
104                 f : IN std_logic_vector(31 downto 0);
105                 g : IN std_logic_vector(31 downto 0);
106                 control : IN std_logic_vector(2 downto 0);
107                 salida : OUT std_logic_vector(31 downto 0)
108                 );
109         END COMPONENT;
110
111 signal result_sr8 : std_logic_vector(31 downto 0);
112 signal ir11 : std_logic_vector(31 downto 0);
113 signal sm7_6,sm7_7 : std_logic_vector(31 downto 0);
114
115 begin
116
```

```
117 registro_36: registro
118
119         generic map(
120         valor_inicial=>x"41400000"--valor decimal de 12
121         )
122
123         PORT MAP(
124                 clk => clk,
125                 reset => reset,
126                 enable => enable_registers_36,
127                 entrada => result_sr8,
128                 salida => sr36
129         );
130
131 registro_44: registro
132
133         generic map(
134         valor_inicial=>x"41400000"--valor decimal de 12
135         )
136
137         PORT MAP(
138                 clk => clk,
139                 reset => reset,
140                 enable => enable_registers_44,
141                 entrada => result_sr8,
142                 salida => sr44
143         );
144
145 registro_11: registro
146
147         generic map(
148         valor_inicial=>x"41400000"--valor decimal de 12
149         )
150
151         PORT MAP(
152                 clk => clk,
153                 reset => reset,
154                 enable => enable_registers_11,
155                 entrada => ir11,
156                 salida => sr11
157         );
158
159 mux7_6: mux7 PORT MAP(
160                 a => mux7_6_a,
161                 b => mux7_6_b,
162                 c => mux7_6_c,
163                 d => mux7_6_d,
164                 e => mux7_6_e,
165                 f => mux7_6_f,
166                 g => mux7_6_g,
167                 control => enable_mux7_6,
168                 salida => sm7_6
169         );
```

```
170
171 mux7_7: mux7 PORT MAP(
172                 a => mux7_7_a,
173                 b => mux7_7_b,
174                 c => mux7_7_c,
175                 d => mux7_7_d,
176                 e => mux7_7_e,
177                 f => mux7_7_f,
178                 g => mux7_7_g,
179                 control => enable_mux7_7,
180                 salida => sm7_7
181         );
182
183 suma_resta_8: suma_resta PORT MAP(
184                 a => sm7_6,
185                 b => sm7_7,
186                 operation => operation_sr8,
187                 operation_nd => operation_nd_sr8,
188                 operation_rfd => operation_rfd_sr8,
189                 clk => clk,
190                 sclr => sclr_sr8,
191                 ce => ce_sr8,
192                 result => result_sr8,
193                 underflow => underflow_sr8,
194                 overflow => overflow_sr8,
195                 invalid_op => invalid_op_sr8,
196                 rdy => rdy_sr8
197         );
198
199 mux2_33: mux2 PORT MAP(
200                 a => result_sr8,
201                 b => z04_1,
202                 control => enable_mux2_33,
203                 salida => ir11
204         );
205
206 end Behavioral;
```

G.10 Sección M1.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:     12:07:54 08/16/2010
6  -- Design Name:
7  -- Module Name:     Seccion_M1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
```

```
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M1 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux4_1_a,mux4_1_b,mux4_1_c,mux4_1_d : in std_logic_vector(31 downto 0);
33 mux5_1_a,mux5_1_b,mux5_1_c,mux5_1_d,mux5_1_e : in std_logic_vector(31 downto 0);
34 sr55 : out std_logic_vector(31 downto 0);
35 enable_registers_55  : IN std_logic;
36 enable_mux4_1 : in std_logic_vector(1 downto 0);
37 enable_mux5_1 : in std_logic_vector(2 downto 0);
38 operation_nd_M1 : in std_logic;
39 operation_rfd_M1 : out std_logic;
40 sclr_M1 : in std_logic;
41 ce_M1 : in std_logic;
42 result_M1 : OUT std_logic_vector(31 downto 0);
43 underflow_M1 : out std_logic;
44 overflow_M1 : out std_logic;
45 invalid_op_M1 : out std_logic;
46 rdy_M1 : out std_logic
47 );
48 end Seccion_M1;
49
50 architecture Behavioral of Seccion_M1 is
51
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
59                 ce : IN std_logic;
60                 operation_rfd : OUT std_logic;
61                 result : OUT std_logic_vector(31 downto 0);
62                 underflow : OUT std_logic;
63                 overflow : OUT std_logic;
64                 invalid_op : OUT std_logic;
```

```
65                  rdy : OUT std_logic
66                  );
67          END COMPONENT;
68
69  COMPONENT mux5
70          PORT(
71                  a : IN std_logic_vector(31 downto 0);
72                  b : IN std_logic_vector(31 downto 0);
73                  c : IN std_logic_vector(31 downto 0);
74                  d : IN std_logic_vector(31 downto 0);
75                  e : IN std_logic_vector(31 downto 0);
76                  control : IN std_logic_vector(2 downto 0);
77                  salida : OUT std_logic_vector(31 downto 0)
78                  );
79          END COMPONENT;
80
81  COMPONENT mux4
82          PORT(
83                  a : IN std_logic_vector(31 downto 0);
84                  b : IN std_logic_vector(31 downto 0);
85                  c : IN std_logic_vector(31 downto 0);
86                  d : IN std_logic_vector(31 downto 0);
87                  control : IN std_logic_vector(1 downto 0);
88                  salida : OUT std_logic_vector(31 downto 0)
89                  );
90          END COMPONENT;
91
92  COMPONENT registro
93      generic(
94      valor_inicial : std_logic_vector(31 downto 0)
95      );
96          PORT(
97                  clk : IN std_logic;
98                  reset : IN std_logic;
99                  enable : IN std_logic;
100                 entrada : IN std_logic_vector(31 downto 0);
101                 salida : OUT std_logic_vector(31 downto 0)
102                 );
103         END COMPONENT;
104
105 signal result_M11 : std_logic_vector(31 downto 0);
106 signal sm4_1,sm5_1 : std_logic_vector(31 downto 0);
107
108 begin
109
110 Inst_multiplicacion_M1: multiplicacion PORT MAP(
111                 a => sm4_1,
112                 b => sm5_1,
113                 operation_nd => operation_nd_M1,
114                 operation_rfd => operation_rfd_M1,
115                 clk => clk,
116                 sclr => sclr_M1,
117                 ce => ce_M1,
```

```
118                 result => result_M11,
119                 underflow => underflow_M1,
120                 overflow => overflow_M1,
121                 invalid_op => invalid_op_M1,
122                 rdy => rdy_M1
123         );
124
125 Inst_mux5_1: mux5 PORT MAP(
126                 a => mux5_1_a,
127                 b => mux5_1_b,
128                 c => mux5_1_c,
129                 d => mux5_1_d,
130                 e => mux5_1_e,
131                 control => enable_mux5_1,
132                 salida => sm5_1
133         );
134
135 Inst_mux4_1: mux4 PORT MAP(
136                 a => mux4_1_a,
137                 b => mux4_1_b,
138                 c => mux4_1_c,
139                 d => mux4_1_d,
140                 control => enable_mux4_1,
141                 salida => sm4_1
142         );
143
144 registro_55: registro
145
146         generic map(
147         valor_inicial=>x"41400000"--valor decimal de 12
148         )
149
150         PORT MAP(
151                 clk => clk,
152                 reset => reset,
153                 enable => enable_registers_55,
154                 entrada => result_M11,
155                 salida => sr55
156         );
157
158 result_M1 <= result_M11;
159
160 end Behavioral;
```

G.11 Sección M2.vhd

```
1 ----------------------------------------------------------------------------------
2 -- Company:
3 -- Engineer:
4 --
5 -- Create Date:    17:27:11 08/16/2010
```

```
6  -- Design Name:
7  -- Module Name:    Seccion_M2 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M2 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux4_2_a,mux4_2_b,mux4_2_c,mux4_2_d : in std_logic_vector(31 downto 0);
33 mux5_2_a,mux5_2_b,mux5_2_c,mux5_2_d,mux5_2_e : in std_logic_vector(31 downto 0);
34 sr58 : out std_logic_vector(31 downto 0);
35 enable_registers_58  : IN std_logic;
36 enable_mux4_2 : in std_logic_vector(1 downto 0);
37 enable_mux5_2 : in std_logic_vector(2 downto 0);
38 operation_nd_M2 : in std_logic;
39 operation_rfd_M2 : out std_logic;
40 sclr_M2 : in std_logic;
41 ce_M2 : in std_logic;
42 result_M2 : OUT std_logic_vector(31 downto 0);
43 underflow_M2 : out std_logic;
44 overflow_M2 : out std_logic;
45 invalid_op_M2 : out std_logic;
46 rdy_M2 : out std_logic
47 );
48 end Seccion_M2;
49
50 architecture Behavioral of Seccion_M2 is
51
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
```

```
59                  ce : IN std_logic;
60                  operation_rfd : OUT std_logic;
61                  result : OUT std_logic_vector(31 downto 0);
62                  underflow : OUT std_logic;
63                  overflow : OUT std_logic;
64                  invalid_op : OUT std_logic;
65                  rdy : OUT std_logic
66                  );
67          END COMPONENT;
68
69  COMPONENT mux5
70          PORT(
71                  a : IN std_logic_vector(31 downto 0);
72                  b : IN std_logic_vector(31 downto 0);
73                  c : IN std_logic_vector(31 downto 0);
74                  d : IN std_logic_vector(31 downto 0);
75                  e : IN std_logic_vector(31 downto 0);
76                  control : IN std_logic_vector(2 downto 0);
77                  salida : OUT std_logic_vector(31 downto 0)
78                  );
79          END COMPONENT;
80
81  COMPONENT mux4
82          PORT(
83                  a : IN std_logic_vector(31 downto 0);
84                  b : IN std_logic_vector(31 downto 0);
85                  c : IN std_logic_vector(31 downto 0);
86                  d : IN std_logic_vector(31 downto 0);
87                  control : IN std_logic_vector(1 downto 0);
88                  salida : OUT std_logic_vector(31 downto 0)
89                  );
90          END COMPONENT;
91
92  COMPONENT registro
93      generic(
94      valor_inicial : std_logic_vector(31 downto 0)
95      );
96          PORT(
97                  clk : IN std_logic;
98                  reset : IN std_logic;
99                  enable : IN std_logic;
100                 entrada : IN std_logic_vector(31 downto 0);
101                 salida : OUT std_logic_vector(31 downto 0)
102                 );
103         END COMPONENT;
104
105 signal result_M22 : std_logic_vector(31 downto 0);
106 signal sm4_2,sm5_2 : std_logic_vector(31 downto 0);
107
108 begin
109
110 Inst_multiplicacion_2: multiplicacion PORT MAP(
111                 a => sm4_2,
```

```
112                     b => sm5_2,
113                     operation_nd => operation_nd_M2,
114                     operation_rfd => operation_rfd_M2,
115                     clk => clk,
116                     sclr => sclr_M2,
117                     ce => ce_M2,
118                     result => result_M22,
119                     underflow => underflow_M2,
120                     overflow => overflow_M2,
121                     invalid_op => invalid_op_M2,
122                     rdy => rdy_M2
123             );
124
125     Inst_mux5_2: mux5 PORT MAP(
126                     a => mux5_2_a,
127                     b => mux5_2_b,
128                     c => mux5_2_c,
129                     d => mux5_2_d,
130                     e => mux5_2_e,
131                     control => enable_mux5_2,
132                     salida => sm5_2
133             );
134
135     Inst_mux4_2: mux4 PORT MAP(
136                     a => mux4_2_a,
137                     b => mux4_2_b,
138                     c => mux4_2_c,
139                     d => mux4_2_d,
140                     control => enable_mux4_2,
141                     salida => sm4_2
142             );
143
144     registro_58: registro
145
146             generic map(
147             valor_inicial=>x"41400000"--valor decimal de 12
148             )
149
150             PORT MAP(
151                     clk => clk,
152                     reset => reset,
153                     enable => enable_registers_58,
154                     entrada => result_M22,
155                     salida => sr58
156             );
157
158     result_M2 <= result_M22;
159
160     end Behavioral;
```

G.12 Sección M3.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:     18:40:27 08/16/2010
6  -- Design Name:
7  -- Module Name:     Seccion_M3 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M3 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux4_3_a,mux4_3_b,mux4_3_c,mux4_3_d : in std_logic_vector(31 downto 0);
33 mux3_1_a,mux3_1_b,mux3_1_c : in std_logic_vector(31 downto 0);
34 sr49 : out std_logic_vector(31 downto 0);
35 enable_registers_49  : IN std_logic;
36 enable_mux4_3 : in std_logic_vector(1 downto 0);
37 enable_mux3_1 : in std_logic_vector(1 downto 0);
38 operation_nd_M3 : in std_logic;
39 operation_rfd_M3 : out std_logic;
40 sclr_M3 : in std_logic;
41 ce_M3 : in std_logic;
42 result_M3 : OUT std_logic_vector(31 downto 0);
43 underflow_M3 : out std_logic;
44 overflow_M3 : out std_logic;
45 invalid_op_M3 : out std_logic;
46 rdy_M3 : out std_logic
47 );
48 end Seccion_M3;
49
50 architecture Behavioral of Seccion_M3 is
51
```

```
52  COMPONENT multiplicacion
53          PORT(
54                  a : IN std_logic_vector(31 downto 0);
55                  b : IN std_logic_vector(31 downto 0);
56                  operation_nd : IN std_logic;
57                  clk : IN std_logic;
58                  sclr : IN std_logic;
59                  ce : IN std_logic;
60                  operation_rfd : OUT std_logic;
61                  result : OUT std_logic_vector(31 downto 0);
62                  underflow : OUT std_logic;
63                  overflow : OUT std_logic;
64                  invalid_op : OUT std_logic;
65                  rdy : OUT std_logic
66                  );
67          END COMPONENT;
68
69  COMPONENT mux4
70          PORT(
71                  a : IN std_logic_vector(31 downto 0);
72                  b : IN std_logic_vector(31 downto 0);
73                  c : IN std_logic_vector(31 downto 0);
74                  d : IN std_logic_vector(31 downto 0);
75                  control : IN std_logic_vector(1 downto 0);
76                  salida : OUT std_logic_vector(31 downto 0)
77                  );
78          END COMPONENT;
79
80  COMPONENT mux3
81          PORT(
82                  a : IN std_logic_vector(31 downto 0);
83                  b : IN std_logic_vector(31 downto 0);
84                  c : IN std_logic_vector(31 downto 0);
85                  control : IN std_logic_vector(1 downto 0);
86                  salida : OUT std_logic_vector(31 downto 0)
87                  );
88          END COMPONENT;
89
90  COMPONENT registro
91      generic(
92      valor_inicial : std_logic_vector(31 downto 0)
93      );
94          PORT(
95                  clk : IN std_logic;
96                  reset : IN std_logic;
97                  enable : IN std_logic;
98                  entrada : IN std_logic_vector(31 downto 0);
99                  salida : OUT std_logic_vector(31 downto 0)
100                 );
101         END COMPONENT;
102
103
104 signal result_M33 : std_logic_vector(31 downto 0);
```

```
105 signal sm3_1,sm4_3 : std_logic_vector(31 downto 0);
106
107 begin
108
109 Inst_multiplicacion_M3: multiplicacion PORT MAP(
110                 a => sm3_1,
111                 b => sm4_3,
112                 operation_nd => operation_nd_M3,
113                 operation_rfd => operation_rfd_M3,
114                 clk => clk,
115                 sclr => sclr_M3,
116                 ce => ce_M3,
117                 result => result_M33,
118                 underflow => underflow_M3,
119                 overflow => overflow_M3,
120                 invalid_op => invalid_op_M3,
121                 rdy => rdy_M3
122         );
123
124 Inst_mux4_3: mux4 PORT MAP(
125                 a => mux4_3_a,
126                 b => mux4_3_b,
127                 c => mux4_3_c,
128                 d => mux4_3_d,
129                 control => enable_mux4_3,
130                 salida => sm4_3
131         );
132
133 mux3_1: mux3 PORT MAP(
134                 a => mux3_1_a,
135                 b => mux3_1_b,
136                 c => mux3_1_c,
137                 control => enable_mux3_1,
138                 salida => sm3_1
139         );
140
141 registro_49: registro
142
143         generic map(
144         valor_inicial=>x"41400000"--valor decimal de 12
145         )
146
147         PORT MAP(
148                 clk => clk,
149                 reset => reset,
150                 enable => enable_registers_49,
151                 entrada => result_M33,
152                 salida => sr49
153         );
154
155 result_M3 <= result_M33;
156
157 end Behavioral;
```

G.13 Sección M4.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    08:11:58 08/17/2010
6  -- Design Name:
7  -- Module Name:    Seccion_M4 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M4 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux4_4_a,mux4_4_b,mux4_4_c,mux4_4_d : in std_logic_vector(31 downto 0);
33 mux3_2_a,mux3_2_b,mux3_2_c : in std_logic_vector(31 downto 0);
34 sr50 : out std_logic_vector(31 downto 0);
35 enable_registers_50  : IN std_logic;
36 enable_mux4_4 : in std_logic_vector(1 downto 0);
37 enable_mux3_2 : in std_logic_vector(1 downto 0);
38 operation_nd_M4 : in std_logic;
39 operation_rfd_M4 : out std_logic;
40 sclr_M4 : in std_logic;
41 ce_M4 : in std_logic;
42 result_M4 : OUT std_logic_vector(31 downto 0);
43 underflow_M4 : out std_logic;
44 overflow_M4 : out std_logic;
45 invalid_op_M4 : out std_logic;
46 rdy_M4 : out std_logic
47 );
48 end Seccion_M4;
49
50 architecture Behavioral of Seccion_M4 is
51
```

```
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
59                 ce : IN std_logic;
60                 operation_rfd : OUT std_logic;
61                 result : OUT std_logic_vector(31 downto 0);
62                 underflow : OUT std_logic;
63                 overflow : OUT std_logic;
64                 invalid_op : OUT std_logic;
65                 rdy : OUT std_logic
66                 );
67         END COMPONENT;
68 
69 COMPONENT mux4
70         PORT(
71                 a : IN std_logic_vector(31 downto 0);
72                 b : IN std_logic_vector(31 downto 0);
73                 c : IN std_logic_vector(31 downto 0);
74                 d : IN std_logic_vector(31 downto 0);
75                 control : IN std_logic_vector(1 downto 0);
76                 salida : OUT std_logic_vector(31 downto 0)
77                 );
78         END COMPONENT;
79 
80 COMPONENT mux3
81         PORT(
82                 a : IN std_logic_vector(31 downto 0);
83                 b : IN std_logic_vector(31 downto 0);
84                 c : IN std_logic_vector(31 downto 0);
85                 control : IN std_logic_vector(1 downto 0);
86                 salida : OUT std_logic_vector(31 downto 0)
87                 );
88         END COMPONENT;
89 
90 COMPONENT registro
91     generic(
92     valor_inicial : std_logic_vector(31 downto 0)
93     );
94         PORT(
95                 clk : IN std_logic;
96                 reset : IN std_logic;
97                 enable : IN std_logic;
98                 entrada : IN std_logic_vector(31 downto 0);
99                 salida : OUT std_logic_vector(31 downto 0)
100                 );
101         END COMPONENT;
102 
103 
104 signal result_M44 : std_logic_vector(31 downto 0);
```

```
105 signal sm3_2,sm4_4 : std_logic_vector(31 downto 0);
106
107 begin
108
109 Inst_multiplicacion_M4: multiplicacion PORT MAP(
110                 a => sm4_4,
111                 b => sm3_2,
112                 operation_nd => operation_nd_M4,
113                 operation_rfd => operation_rfd_M4,
114                 clk => clk,
115                 sclr => sclr_M4,
116                 ce => ce_M4,
117                 result => result_M44,
118                 underflow => underflow_M4,
119                 overflow => overflow_M4,
120                 invalid_op => invalid_op_M4,
121                 rdy => rdy_M4
122         );
123
124 Inst_mux4_4: mux4 PORT MAP(
125                 a => mux4_4_a,
126                 b => mux4_4_b,
127                 c => mux4_4_c,
128                 d => mux4_4_d,
129                 control => enable_mux4_4,
130                 salida => sm4_4
131         );
132
133 mux3_2: mux3 PORT MAP(
134                 a => mux3_2_a,
135                 b => mux3_2_b,
136                 c => mux3_2_c,
137                 control => enable_mux3_2,
138                 salida => sm3_2
139         );
140
141 registro_50: registro
142
143         generic map(
144         valor_inicial=>x"41400000"--valor decimal de 12
145         )
146
147         PORT MAP(
148                 clk => clk,
149                 reset => reset,
150                 enable => enable_registers_50,
151                 entrada => result_M44,
152                 salida => sr50
153         );
154
155 result_M4 <= result_M44;
156
157 end Behavioral;
```

G.14 Sección M5.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:     15:14:59 08/17/2010
6  -- Design Name:
7  -- Module Name:     Seccion_M5 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M5 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux3_3_a,mux3_3_b,mux3_3_c : in std_logic_vector(31 downto 0);
33 mux3_4_a,mux3_4_b,mux3_4_c : in std_logic_vector(31 downto 0);
34 sr59             : out std_logic_vector(31 downto 0);
35 enable_registers_59  : IN std_logic;
36 enable_mux3_3 : in std_logic_vector(1 downto 0);
37 enable_mux3_4 : in std_logic_vector(1 downto 0);
38 operation_nd_M5 : in std_logic;
39 operation_rfd_M5 : out std_logic;
40 sclr_M5 : in std_logic;
41 ce_M5 : in std_logic;
42 result_M5 : OUT std_logic_vector(31 downto 0);
43 underflow_M5 : out std_logic;
44 overflow_M5 : out std_logic;
45 invalid_op_M5 : out std_logic;
46 rdy_M5 : out std_logic
47 );
48 end Seccion_M5;
49
50 architecture Behavioral of Seccion_M5 is
51
```

```
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
59                 ce : IN std_logic;
60                 operation_rfd : OUT std_logic;
61                 result : OUT std_logic_vector(31 downto 0);
62                 underflow : OUT std_logic;
63                 overflow : OUT std_logic;
64                 invalid_op : OUT std_logic;
65                 rdy : OUT std_logic
66                 );
67         END COMPONENT;
68 
69 COMPONENT mux3
70         PORT(
71                 a : IN std_logic_vector(31 downto 0);
72                 b : IN std_logic_vector(31 downto 0);
73                 c : IN std_logic_vector(31 downto 0);
74                 control : IN std_logic_vector(1 downto 0);
75                 salida : OUT std_logic_vector(31 downto 0)
76                 );
77         END COMPONENT;
78 
79 COMPONENT registro
80     generic(
81     valor_inicial : std_logic_vector(31 downto 0)
82     );
83         PORT(
84                 clk : IN std_logic;
85                 reset : IN std_logic;
86                 enable : IN std_logic;
87                 entrada : IN std_logic_vector(31 downto 0);
88                 salida : OUT std_logic_vector(31 downto 0)
89                 );
90         END COMPONENT;
91 
92 signal result_M55 : std_logic_vector(31 downto 0);
93 signal sm3_3,sm3_4 : std_logic_vector(31 downto 0);
94 
95 begin
96 
97 Inst_multiplicacion_M5: multiplicacion PORT MAP(
98                 a => sm3_3,
99                 b => sm3_4,
100                operation_nd => operation_nd_M5,
101                operation_rfd => operation_rfd_M5,
102                clk => clk,
103                sclr => sclr_M5,
104                ce => ce_M5,
```

```
105                 result => result_M55,
106                 underflow => underflow_M5,
107                 overflow => overflow_M5,
108                 invalid_op => invalid_op_M5,
109                 rdy => rdy_M5
110         );
111
112 mux3_3: mux3 PORT MAP(
113                 a => mux3_3_a,
114                 b => mux3_3_b,
115                 c => mux3_3_c,
116                 control => enable_mux3_3,
117                 salida => sm3_3
118         );
119
120 mux3_4: mux3 PORT MAP(
121                 a => mux3_4_a,
122                 b => mux3_4_b,
123                 c => mux3_4_c,
124                 control => enable_mux3_4,
125                 salida => sm3_4
126         );
127
128 registro_59: registro
129
130         generic map(
131         valor_inicial=>x"41400000"--valor decimal de 12
132         )
133
134         PORT MAP(
135                 clk => clk,
136                 reset => reset,
137                 enable => enable_registers_59,
138                 entrada => result_M55,
139                 salida => sr59
140         );
141
142 result_M5 <= result_M55;
143
144 end Behavioral;
```

G.15 Sección M6.vhd

```
1 ----------------------------------------------------------------------------------
2 -- Company:
3 -- Engineer:
4 --
5 -- Create Date:    15:47:50 08/17/2010
6 -- Design Name:
7 -- Module Name:    Seccion_M6 - Behavioral
8 -- Project Name:
```

```
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M6 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux4_5_a,mux4_5_b,mux4_5_c,mux4_5_d : in std_logic_vector(31 downto 0);
33 mux3_5_a,mux3_5_b,mux3_5_c : in std_logic_vector(31 downto 0);
34 sr56 : out std_logic_vector(31 downto 0);
35 enable_registers_56  : IN std_logic;
36 enable_mux4_5 : in std_logic_vector(1 downto 0);
37 enable_mux3_5 : in std_logic_vector(1 downto 0);
38 operation_nd_M6 : in std_logic;
39 operation_rfd_M6 : out std_logic;
40 sclr_M6 : in std_logic;
41 ce_M6 : in std_logic;
42 result_M6 : OUT std_logic_vector(31 downto 0);
43 underflow_M6 : out std_logic;
44 overflow_M6 : out std_logic;
45 invalid_op_M6 : out std_logic;
46 rdy_M6 : out std_logic
47 );
48 end Seccion_M6;
49
50 architecture Behavioral of Seccion_M6 is
51
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
59                 ce : IN std_logic;
60                 operation_rfd : OUT std_logic;
61                 result : OUT std_logic_vector(31 downto 0);
```

```
62                  underflow : OUT std_logic;
63                  overflow : OUT std_logic;
64                  invalid_op : OUT std_logic;
65                  rdy : OUT std_logic
66                  );
67          END COMPONENT;
68
69  COMPONENT mux4
70          PORT(
71                  a : IN std_logic_vector(31 downto 0);
72                  b : IN std_logic_vector(31 downto 0);
73                  c : IN std_logic_vector(31 downto 0);
74                  d : IN std_logic_vector(31 downto 0);
75                  control : IN std_logic_vector(1 downto 0);
76                  salida : OUT std_logic_vector(31 downto 0)
77                  );
78          END COMPONENT;
79
80  COMPONENT mux3
81          PORT(
82                  a : IN std_logic_vector(31 downto 0);
83                  b : IN std_logic_vector(31 downto 0);
84                  c : IN std_logic_vector(31 downto 0);
85                  control : IN std_logic_vector(1 downto 0);
86                  salida : OUT std_logic_vector(31 downto 0)
87                  );
88          END COMPONENT;
89
90  COMPONENT registro
91      generic(
92      valor_inicial : std_logic_vector(31 downto 0)
93      );
94          PORT(
95                  clk : IN std_logic;
96                  reset : IN std_logic;
97                  enable : IN std_logic;
98                  entrada : IN std_logic_vector(31 downto 0);
99                  salida : OUT std_logic_vector(31 downto 0)
100                 );
101         END COMPONENT;
102
103
104 signal result_M66 : std_logic_vector(31 downto 0);
105 signal sm3_5,sm4_5 : std_logic_vector(31 downto 0);
106
107 begin
108
109 Inst_multiplicacion_M6: multiplicacion PORT MAP(
110                 a => sm3_5,
111                 b => sm4_5,
112                 operation_nd => operation_nd_M6,
113                 operation_rfd => operation_rfd_M6,
114                 clk => clk,
```

```
115                     sclr => sclr_M6,
116                     ce => ce_M6,
117                     result => result_M66,
118                     underflow => underflow_M6,
119                     overflow => overflow_M6,
120                     invalid_op => invalid_op_M6,
121                     rdy => rdy_M6
122             );
123
124     Inst_mux4_5: mux4 PORT MAP(
125                     a => mux4_5_a,
126                     b => mux4_5_b,
127                     c => mux4_5_c,
128                     d => mux4_5_d,
129                     control => enable_mux4_5,
130                     salida => sm4_5
131             );
132
133     mux3_5: mux3 PORT MAP(
134                     a => mux3_5_a,
135                     b => mux3_5_b,
136                     c => mux3_5_c,
137                     control => enable_mux3_5,
138                     salida => sm3_5
139             );
140
141     registro_56: registro
142
143             generic map(
144             valor_inicial=>x"41400000"--valor decimal de 12
145             )
146
147             PORT MAP(
148                     clk => clk,
149                     reset => reset,
150                     enable => enable_registers_56,
151                     entrada => result_M66,
152                     salida => sr56
153             );
154
155     result_M6 <= result_M66;
156
157     end Behavioral;
```

G.16 Sección M7.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    16:00:42 08/17/2010
```

```
6  -- Design Name:
7  -- Module Name:    Seccion_M7 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M7 is
31     Port ( clk,reset : in  STD_LOGIC;
32 mux3_6_a,mux3_6_b,mux3_6_c : in std_logic_vector(31 downto 0);
33 mux2_34_a,mux2_34_b : in std_logic_vector(31 downto 0);
34 sr57            : out std_logic_vector(31 downto 0);
35 enable_registers_57  : IN std_logic;
36 enable_mux3_6 : in std_logic_vector(1 downto 0);
37 enable_mux2_34 : IN std_logic;
38 operation_nd_M7 : in std_logic;
39 operation_rfd_M7 : out std_logic;
40 sclr_M7 : in std_logic;
41 ce_M7 : in std_logic;
42 result_M7 : OUT std_logic_vector(31 downto 0);
43 underflow_M7 : out std_logic;
44 overflow_M7 : out std_logic;
45 invalid_op_M7 : out std_logic;
46 rdy_M7 : out std_logic
47 );
48 end Seccion_M7;
49
50 architecture Behavioral of Seccion_M7 is
51
52 COMPONENT multiplicacion
53         PORT(
54                 a : IN std_logic_vector(31 downto 0);
55                 b : IN std_logic_vector(31 downto 0);
56                 operation_nd : IN std_logic;
57                 clk : IN std_logic;
58                 sclr : IN std_logic;
```

```
59                  ce : IN std_logic;
60                  operation_rfd : OUT std_logic;
61                  result : OUT std_logic_vector(31 downto 0);
62                  underflow : OUT std_logic;
63                  overflow : OUT std_logic;
64                  invalid_op : OUT std_logic;
65                  rdy : OUT std_logic
66                  );
67          END COMPONENT;
68
69  COMPONENT mux2
70          PORT(
71                  a : IN std_logic_vector(31 downto 0);
72                  b : IN std_logic_vector(31 downto 0);
73                  control : IN std_logic;
74                  salida : OUT std_logic_vector(31 downto 0)
75                  );
76          END COMPONENT;
77
78  COMPONENT mux3
79          PORT(
80                  a : IN std_logic_vector(31 downto 0);
81                  b : IN std_logic_vector(31 downto 0);
82                  c : IN std_logic_vector(31 downto 0);
83                  control : IN std_logic_vector(1 downto 0);
84                  salida : OUT std_logic_vector(31 downto 0)
85                  );
86          END COMPONENT;
87
88  COMPONENT registro
89      generic(
90      valor_inicial : std_logic_vector(31 downto 0)
91      );
92          PORT(
93                  clk : IN std_logic;
94                  reset : IN std_logic;
95                  enable : IN std_logic;
96                  entrada : IN std_logic_vector(31 downto 0);
97                  salida : OUT std_logic_vector(31 downto 0)
98                  );
99          END COMPONENT;
100
101 signal result_M77 : std_logic_vector(31 downto 0);
102 signal sm3_6,sm2_34 : std_logic_vector(31 downto 0);
103
104 begin
105
106 Inst_multiplicacion_M7: multiplicacion PORT MAP(
107                 a => sm2_34,
108                 b => sm3_6,
109                 operation_nd => operation_nd_M7,
110                 operation_rfd => operation_rfd_M7,
111                 clk => clk,
```

```
112                     sclr => sclr_M7,
113                     ce => ce_M7,
114                     result => result_M77,
115                     underflow => underflow_M7,
116                     overflow => overflow_M7,
117                     invalid_op => invalid_op_M7,
118                     rdy => rdy_M7
119             );
120
121     mux2_34: mux2 PORT MAP(
122                     a => mux2_34_a,
123                     b => mux2_34_b,
124                     control => enable_mux2_34,
125                     salida => sm2_34
126             );
127
128     mux3_6: mux3 PORT MAP(
129                     a => mux3_6_a,
130                     b => mux3_6_b,
131                     c => mux3_6_c,
132                     control => enable_mux3_6,
133                     salida => sm3_6
134             );
135
136     registro_57: registro
137
138             generic map(
139             valor_inicial=>x"41400000"--valor decimal de 12
140             )
141
142             PORT MAP(
143                     clk => clk,
144                     reset => reset,
145                     enable => enable_registers_57,
146                     entrada => result_M77,
147                     salida => sr57
148             );
149
150     result_M7 <= result_M77;
151
152     end Behavioral;
```

G.17 Sección M8.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:     16:50:01 08/17/2010
6   -- Design Name:
7   -- Module Name:     Seccion_M8 - Behavioral
```

```
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Seccion_M8 is
31     Port (  clk,reset : in  STD_LOGIC;
32 mux4_6_a,mux4_6_b,mux4_6_c,mux4_6_d : in std_logic_vector(31 downto 0);
33 mux3_7_a,mux3_7_b,mux3_7_c : in std_logic_vector(31 downto 0);
34 sr54 : out std_logic_vector(31 downto 0);
35 enable_registers_54  : IN std_logic;
36 enable_mux4_6 : in std_logic_vector(1 downto 0);
37 enable_mux3_7 : in std_logic_vector(1 downto 0);
38 operation_nd_M8 : in std_logic;
39 operation_rfd_M8 : out std_logic;
40 sclr_M8 : in std_logic;
41 ce_M8 : in std_logic;
42 underflow_M8 : out std_logic;
43 overflow_M8 : out std_logic;
44 invalid_op_M8 : out std_logic;
45 rdy_M8 : out std_logic
46 );
47 end Seccion_M8;
48
49 architecture Behavioral of Seccion_M8 is
50
51 COMPONENT multiplicacion
52         PORT(
53                 a : IN std_logic_vector(31 downto 0);
54                 b : IN std_logic_vector(31 downto 0);
55                 operation_nd : IN std_logic;
56                 clk : IN std_logic;
57                 sclr : IN std_logic;
58                 ce : IN std_logic;
59                 operation_rfd : OUT std_logic;
60                 result : OUT std_logic_vector(31 downto 0);
```

```
61                  underflow : OUT std_logic;
62                  overflow : OUT std_logic;
63                  invalid_op : OUT std_logic;
64                  rdy : OUT std_logic
65                  );
66          END COMPONENT;
67
68  COMPONENT mux4
69          PORT(
70                  a : IN std_logic_vector(31 downto 0);
71                  b : IN std_logic_vector(31 downto 0);
72                  c : IN std_logic_vector(31 downto 0);
73                  d : IN std_logic_vector(31 downto 0);
74                  control : IN std_logic_vector(1 downto 0);
75                  salida : OUT std_logic_vector(31 downto 0)
76                  );
77          END COMPONENT;
78
79  COMPONENT mux3
80          PORT(
81                  a : IN std_logic_vector(31 downto 0);
82                  b : IN std_logic_vector(31 downto 0);
83                  c : IN std_logic_vector(31 downto 0);
84                  control : IN std_logic_vector(1 downto 0);
85                  salida : OUT std_logic_vector(31 downto 0)
86                  );
87          END COMPONENT;
88
89  COMPONENT registro
90      generic(
91      valor_inicial : std_logic_vector(31 downto 0)
92      );
93          PORT(
94                  clk : IN std_logic;
95                  reset : IN std_logic;
96                  enable : IN std_logic;
97                  entrada : IN std_logic_vector(31 downto 0);
98                  salida : OUT std_logic_vector(31 downto 0)
99                  );
100         END COMPONENT;
101
102
103 signal result_M88 : std_logic_vector(31 downto 0);
104 signal sm3_7,sm4_6 : std_logic_vector(31 downto 0);
105
106 begin
107
108 Inst_multiplicacion_M8: multiplicacion PORT MAP(
109                 a => sm4_6,
110                 b => sm3_7,
111                 operation_nd => operation_nd_M8,
112                 operation_rfd => operation_rfd_M8,
113                 clk => clk,
```

```
114                     sclr => sclr_M8,
115                     ce => ce_M8,
116                     result => result_M88,
117                     underflow => underflow_M8,
118                     overflow => overflow_M8,
119                     invalid_op => invalid_op_M8,
120                     rdy => rdy_M8
121             );
122
123 Inst_mux4_6: mux4 PORT MAP(
124                     a => mux4_6_a,
125                     b => mux4_6_b,
126                     c => mux4_6_c,
127                     d => mux4_6_d,
128                     control => enable_mux4_6,
129                     salida => sm4_6
130             );
131
132 mux3_7: mux3 PORT MAP(
133                     a => mux3_7_a,
134                     b => mux3_7_b,
135                     c => mux3_7_c,
136                     control => enable_mux3_7,
137                     salida => sm3_7
138             );
139
140 registro_54: registro
141
142             generic map(
143             valor_inicial=>x"41400000"--valor decimal de 12
144             )
145
146             PORT MAP(
147                     clk => clk,
148                     reset => reset,
149                     enable => enable_registers_54,
150                     entrada => result_M88,
151                     salida => sr54
152             );
153
154 end Behavioral;
```

G.18 Sección FSM7.vhd

```
1
2 ----------------------------------------------------------------------------------
3 -- Company:
4 -- Engineer:
5 --
6 -- Create Date:    11:32:06 08/19/2010
7 -- Design Name:
```

```
8  -- Module Name:    FSM_7 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity FSM_7 is
32     Port (  clk,inicio : in  STD_LOGIC;
33     m : in  STD_LOGIC_VECTOR (4 downto 0);
34     reset : in  STD_LOGIC;
35     banderas1,banderas2 : in STD_LOGIC_VECTOR (15 downto 0);
36     estado_sig : out std_logic_vector(4 downto 0);
37     senales_de_control : out STD_LOGIC_VECTOR (272 downto 0));
38     attribute FSM_EXTRACT : string;
39     attribute FSM_EXTRACT of FSM_7: entity is "NO";
40 end FSM_7;
41
42 architecture Behavioral of FSM_7 is
43
44 constant N : integer := 224;
45 signal estado_presente,estado_siguiente : STD_LOGIC_VECTOR (N downto 0) ;
46 signal L : STD_LOGIC_VECTOR (4 downto 0):="00000";
47 signal flag1,flag2,flag3 : STD_LOGIC :='0';
48 signal m1 : STD_LOGIC_VECTOR (4 downto 0):="00000";
49
50 ----------registros--------------------
51 constant R2_ON  : std_logic := '1';
52 constant R2_OFF : std_logic := '0';
53 constant R3_ON  : std_logic := '1';
54 constant R3_OFF : std_logic := '0';
55 constant R4_ON  : std_logic := '1';
56 constant R4_OFF : std_logic := '0';
57 constant R5_ON  : std_logic := '1';
58 constant R5_OFF : std_logic := '0';
59 constant R6_ON  : std_logic := '1';
60 constant R6_OFF : std_logic := '0';
```

```
61  constant R7_ON   : std_logic := '1';
62  constant R7_OFF  : std_logic := '0';
63  constant R8_ON   : std_logic := '1';
64  constant R8_OFF  : std_logic := '0';
65  constant R9_ON   : std_logic := '1';
66  constant R9_OFF  : std_logic := '0';
67  constant R10_ON   : std_logic := '1';
68  constant R10_OFF  : std_logic := '0';
69  constant R11_ON   : std_logic := '1';
70  constant R11_OFF  : std_logic := '0';
71  constant R12_ON   : std_logic := '1';
72  constant R12_OFF  : std_logic := '0';
73  constant R13_ON   : std_logic := '1';
74  constant R13_OFF  : std_logic := '0';
75  constant R14_ON   : std_logic := '1';
76  constant R14_OFF  : std_logic := '0';
77  constant R15_ON   : std_logic := '1';
78  constant R15_OFF  : std_logic := '0';
79  constant R16_ON   : std_logic := '1';
80  constant R16_OFF  : std_logic := '0';
81  constant R17_ON   : std_logic := '1';
82  constant R17_OFF  : std_logic := '0';
83  constant R18_ON   : std_logic := '1';
84  constant R18_OFF  : std_logic := '0';
85  constant R19_ON   : std_logic := '1';
86  constant R19_OFF  : std_logic := '0';
87  constant R20_ON   : std_logic := '1';
88  constant R20_OFF  : std_logic := '0';
89  constant R21_ON   : std_logic := '1';
90  constant R21_OFF  : std_logic := '0';
91  constant R28_ON   : std_logic := '1';
92  constant R28_OFF  : std_logic := '0';
93  constant R29_ON   : std_logic := '1';
94  constant R29_OFF  : std_logic := '0';
95  constant R30_ON   : std_logic := '1';
96  constant R30_OFF  : std_logic := '0';
97  constant R31_ON   : std_logic := '1';
98  constant R31_OFF  : std_logic := '0';
99  constant R32_ON   : std_logic := '1';
100 constant R32_OFF  : std_logic := '0';
101 constant R33_ON   : std_logic := '1';
102 constant R33_OFF  : std_logic := '0';
103 constant R34_ON   : std_logic := '1';
104 constant R34_OFF  : std_logic := '0';
105 constant R35_ON   : std_logic := '1';
106 constant R35_OFF  : std_logic := '0';
107 constant R36_ON   : std_logic := '1';
108 constant R36_OFF  : std_logic := '0';
109 constant R37_ON   : std_logic := '1';
110 constant R37_OFF  : std_logic := '0';
111 constant R38_ON   : std_logic := '1';
112 constant R38_OFF  : std_logic := '0';
113 constant R39_ON   : std_logic := '1';
```

```
114 constant R39_OFF : std_logic := '0';
115 constant R40_ON  : std_logic := '1';
116 constant R40_OFF : std_logic := '0';
117 constant R41_ON  : std_logic := '1';
118 constant R41_OFF : std_logic := '0';
119 constant R42_ON  : std_logic := '1';
120 constant R42_OFF : std_logic := '0';
121 constant R43_ON  : std_logic := '1';
122 constant R43_OFF : std_logic := '0';
123 constant R44_ON  : std_logic := '1';
124 constant R44_OFF : std_logic := '0';
125 constant R45_ON  : std_logic := '1';
126 constant R45_OFF : std_logic := '0';
127 constant R46_ON  : std_logic := '1';
128 constant R46_OFF : std_logic := '0';
129 constant R47_ON  : std_logic := '1';
130 constant R47_OFF : std_logic := '0';
131 constant R48_ON  : std_logic := '1';
132 constant R48_OFF : std_logic := '0';
133 constant R49_ON  : std_logic := '1';
134 constant R49_OFF : std_logic := '0';
135 constant R50_ON  : std_logic := '1';
136 constant R50_OFF : std_logic := '0';
137 constant R51_ON  : std_logic := '1';
138 constant R51_OFF : std_logic := '0';
139 constant R52_ON  : std_logic := '1';
140 constant R52_OFF : std_logic := '0';
141 constant R53_ON  : std_logic := '1';
142 constant R53_OFF : std_logic := '0';
143 constant R54_ON  : std_logic := '1';
144 constant R54_OFF : std_logic := '0';
145 constant R55_ON  : std_logic := '1';
146 constant R55_OFF : std_logic := '0';
147 constant R56_ON  : std_logic := '1';
148 constant R56_OFF : std_logic := '0';
149 constant R57_ON  : std_logic := '1';
150 constant R57_OFF : std_logic := '0';
151 constant R58_ON  : std_logic := '1';
152 constant R58_OFF : std_logic := '0';
153 constant R59_ON  : std_logic := '1';
154 constant R59_OFF : std_logic := '0';
155
156 ------multiplexores------
157
158 constant mux_2_1_a : std_logic := '0';
159 constant mux_2_1_b : std_logic := '1';
160 constant mux_2_2_a : std_logic := '0';
161 constant mux_2_2_b : std_logic := '1';
162 constant mux_2_3_a : std_logic := '0';
163 constant mux_2_3_b : std_logic := '1';
164 constant mux_2_4_a : std_logic := '0';
165 constant mux_2_4_b : std_logic := '1';
166 constant mux_2_5_a : std_logic := '0';
```

```
167  constant mux_2_5_b : std_logic := '1';
168  constant mux_2_6_a : std_logic := '0';
169  constant mux_2_6_b : std_logic := '1';
170  constant mux_2_7_a : std_logic := '0';
171  constant mux_2_7_b : std_logic := '1';
172  constant mux_2_8_a : std_logic := '0';
173  constant mux_2_8_b : std_logic := '1';
174  constant mux_2_9_a : std_logic := '0';
175  constant mux_2_9_b : std_logic := '1';
176  constant mux_2_10_a : std_logic := '0';
177  constant mux_2_10_b : std_logic := '1';
178  constant mux_2_11_a : std_logic := '0';
179  constant mux_2_11_b : std_logic := '1';
180  constant mux_2_12_a : std_logic := '0';
181  constant mux_2_12_b : std_logic := '1';
182  constant mux_2_13_a : std_logic := '0';
183  constant mux_2_13_b : std_logic := '1';
184  constant mux_2_14_a : std_logic := '0';
185  constant mux_2_14_b : std_logic := '1';
186  constant mux_2_15_a : std_logic := '0';
187  constant mux_2_15_b : std_logic := '1';
188  constant mux_2_16_a : std_logic := '0';
189  constant mux_2_16_b : std_logic := '1';
190  constant mux_2_17_a : std_logic := '0';
191  constant mux_2_17_b : std_logic := '1';
192  constant mux_2_18_a : std_logic := '0';
193  constant mux_2_18_b : std_logic := '1';
194  constant mux_2_19_a : std_logic := '0';
195  constant mux_2_19_b : std_logic := '1';
196  constant mux_2_20_a : std_logic := '0';
197  constant mux_2_20_b : std_logic := '1';
198  constant mux_2_21_a : std_logic := '0';
199  constant mux_2_21_b : std_logic := '1';
200  constant mux_2_22_a : std_logic := '0';
201  constant mux_2_22_b : std_logic := '1';
202  constant mux_2_23_a : std_logic := '0';
203  constant mux_2_23_b : std_logic := '1';
204  constant mux_2_24_a : std_logic := '0';
205  constant mux_2_24_b : std_logic := '1';
206  constant mux_2_25_a : std_logic := '0';
207  constant mux_2_25_b : std_logic := '1';
208  constant mux_2_26_a : std_logic := '0';
209  constant mux_2_26_b : std_logic := '1';
210  constant mux_2_27_a : std_logic := '0';
211  constant mux_2_27_b : std_logic := '1';
212  constant mux_2_28_a : std_logic := '0';
213  constant mux_2_28_b : std_logic := '1';
214  constant mux_2_29_a : std_logic := '0';
215  constant mux_2_29_b : std_logic := '1';
216  constant mux_2_30_a : std_logic := '0';
217  constant mux_2_30_b : std_logic := '1';
218  constant mux_2_31_a : std_logic := '0';
219  constant mux_2_31_b : std_logic := '1';
```

```
220 constant mux_2_32_a : std_logic := '0';
221 constant mux_2_32_b : std_logic := '1';
222 constant mux_2_33_a : std_logic := '0';
223 constant mux_2_33_b : std_logic := '1';
224 constant mux_2_34_a : std_logic := '0';
225 constant mux_2_34_b : std_logic := '1';
226
227 constant mux_3_1_a : std_logic_vector(1 downto 0) := "00";
228 constant mux_3_1_b : std_logic_vector(1 downto 0) := "01";
229 constant mux_3_1_c : std_logic_vector(1 downto 0) := "10";
230 constant mux_3_2_a : std_logic_vector(1 downto 0) := "00";
231 constant mux_3_2_b : std_logic_vector(1 downto 0) := "01";
232 constant mux_3_2_c : std_logic_vector(1 downto 0) := "10";
233 constant mux_3_3_a : std_logic_vector(1 downto 0) := "00";
234 constant mux_3_3_b : std_logic_vector(1 downto 0) := "01";
235 constant mux_3_3_c : std_logic_vector(1 downto 0) := "10";
236 constant mux_3_4_a : std_logic_vector(1 downto 0) := "00";
237 constant mux_3_4_b : std_logic_vector(1 downto 0) := "01";
238 constant mux_3_4_c : std_logic_vector(1 downto 0) := "10";
239 constant mux_3_5_a : std_logic_vector(1 downto 0) := "00";
240 constant mux_3_5_b : std_logic_vector(1 downto 0) := "01";
241 constant mux_3_5_c : std_logic_vector(1 downto 0) := "10";
242 constant mux_3_6_a : std_logic_vector(1 downto 0) := "00";
243 constant mux_3_6_b : std_logic_vector(1 downto 0) := "01";
244 constant mux_3_6_c : std_logic_vector(1 downto 0) := "10";
245 constant mux_3_7_a : std_logic_vector(1 downto 0) := "00";
246 constant mux_3_7_b : std_logic_vector(1 downto 0) := "01";
247 constant mux_3_7_c : std_logic_vector(1 downto 0) := "10";
248
249 constant mux_4_1_a : std_logic_vector(1 downto 0) := "00";
250 constant mux_4_1_b : std_logic_vector(1 downto 0) := "01";
251 constant mux_4_1_c : std_logic_vector(1 downto 0) := "10";
252 constant mux_4_1_d : std_logic_vector(1 downto 0) := "11";
253 constant mux_4_2_a : std_logic_vector(1 downto 0) := "00";
254 constant mux_4_2_b : std_logic_vector(1 downto 0) := "01";
255 constant mux_4_2_c : std_logic_vector(1 downto 0) := "10";
256 constant mux_4_2_d : std_logic_vector(1 downto 0) := "11";
257 constant mux_4_3_a : std_logic_vector(1 downto 0) := "00";
258 constant mux_4_3_b : std_logic_vector(1 downto 0) := "01";
259 constant mux_4_3_c : std_logic_vector(1 downto 0) := "10";
260 constant mux_4_3_d : std_logic_vector(1 downto 0) := "11";
261 constant mux_4_4_a : std_logic_vector(1 downto 0) := "00";
262 constant mux_4_4_b : std_logic_vector(1 downto 0) := "01";
263 constant mux_4_4_c : std_logic_vector(1 downto 0) := "10";
264 constant mux_4_4_d : std_logic_vector(1 downto 0) := "11";
265 constant mux_4_5_a : std_logic_vector(1 downto 0) := "00";
266 constant mux_4_5_b : std_logic_vector(1 downto 0) := "01";
267 constant mux_4_5_c : std_logic_vector(1 downto 0) := "10";
268 constant mux_4_5_d : std_logic_vector(1 downto 0) := "11";
269 constant mux_4_6_a : std_logic_vector(1 downto 0) := "00";
270 constant mux_4_6_b : std_logic_vector(1 downto 0) := "01";
271 constant mux_4_6_c : std_logic_vector(1 downto 0) := "10";
272 constant mux_4_6_d : std_logic_vector(1 downto 0) := "11";
```

```
273
274 constant mux_5_1_a : std_logic_vector(2 downto 0) := "000";
275 constant mux_5_1_b : std_logic_vector(2 downto 0) := "001";
276 constant mux_5_1_c : std_logic_vector(2 downto 0) := "010";
277 constant mux_5_1_d : std_logic_vector(2 downto 0) := "011";
278 constant mux_5_1_e : std_logic_vector(2 downto 0) := "100";
279 constant mux_5_2_a : std_logic_vector(2 downto 0) := "000";
280 constant mux_5_2_b : std_logic_vector(2 downto 0) := "001";
281 constant mux_5_2_c : std_logic_vector(2 downto 0) := "010";
282 constant mux_5_2_d : std_logic_vector(2 downto 0) := "011";
283 constant mux_5_2_e : std_logic_vector(2 downto 0) := "100";
284
285 constant mux_7_1_a : std_logic_vector(2 downto 0) := "000";
286 constant mux_7_1_b : std_logic_vector(2 downto 0) := "001";
287 constant mux_7_1_c : std_logic_vector(2 downto 0) := "010";
288 constant mux_7_1_d : std_logic_vector(2 downto 0) := "011";
289 constant mux_7_1_e : std_logic_vector(2 downto 0) := "100";
290 constant mux_7_1_f : std_logic_vector(2 downto 0) := "101";
291 constant mux_7_1_g : std_logic_vector(2 downto 0) := "110";
292 constant mux_7_2_a : std_logic_vector(2 downto 0) := "000";
293 constant mux_7_2_b : std_logic_vector(2 downto 0) := "001";
294 constant mux_7_2_c : std_logic_vector(2 downto 0) := "010";
295 constant mux_7_2_d : std_logic_vector(2 downto 0) := "011";
296 constant mux_7_2_e : std_logic_vector(2 downto 0) := "100";
297 constant mux_7_2_f : std_logic_vector(2 downto 0) := "101";
298 constant mux_7_2_g : std_logic_vector(2 downto 0) := "110";
299 constant mux_7_3_a : std_logic_vector(2 downto 0) := "000";
300 constant mux_7_3_b : std_logic_vector(2 downto 0) := "001";
301 constant mux_7_3_c : std_logic_vector(2 downto 0) := "010";
302 constant mux_7_3_d : std_logic_vector(2 downto 0) := "011";
303 constant mux_7_3_e : std_logic_vector(2 downto 0) := "100";
304 constant mux_7_3_f : std_logic_vector(2 downto 0) := "101";
305 constant mux_7_3_g : std_logic_vector(2 downto 0) := "110";
306 constant mux_7_4_a : std_logic_vector(2 downto 0) := "000";
307 constant mux_7_4_b : std_logic_vector(2 downto 0) := "001";
308 constant mux_7_4_c : std_logic_vector(2 downto 0) := "010";
309 constant mux_7_4_d : std_logic_vector(2 downto 0) := "011";
310 constant mux_7_4_e : std_logic_vector(2 downto 0) := "100";
311 constant mux_7_4_f : std_logic_vector(2 downto 0) := "101";
312 constant mux_7_4_g : std_logic_vector(2 downto 0) := "110";
313 constant mux_7_5_a : std_logic_vector(2 downto 0) := "000";
314 constant mux_7_5_b : std_logic_vector(2 downto 0) := "001";
315 constant mux_7_5_c : std_logic_vector(2 downto 0) := "010";
316 constant mux_7_5_d : std_logic_vector(2 downto 0) := "011";
317 constant mux_7_5_e : std_logic_vector(2 downto 0) := "100";
318 constant mux_7_5_f : std_logic_vector(2 downto 0) := "101";
319 constant mux_7_5_g : std_logic_vector(2 downto 0) := "110";
320 constant mux_7_6_a : std_logic_vector(2 downto 0) := "000";
321 constant mux_7_6_b : std_logic_vector(2 downto 0) := "001";
322 constant mux_7_6_c : std_logic_vector(2 downto 0) := "010";
323 constant mux_7_6_d : std_logic_vector(2 downto 0) := "011";
324 constant mux_7_6_e : std_logic_vector(2 downto 0) := "100";
325 constant mux_7_6_f : std_logic_vector(2 downto 0) := "101";
```

```
326 constant mux_7_6_g : std_logic_vector(2 downto 0) := "110";
327 constant mux_7_7_a : std_logic_vector(2 downto 0) := "000";
328 constant mux_7_7_b : std_logic_vector(2 downto 0) := "001";
329 constant mux_7_7_c : std_logic_vector(2 downto 0) := "010";
330 constant mux_7_7_d : std_logic_vector(2 downto 0) := "011";
331 constant mux_7_7_e : std_logic_vector(2 downto 0) := "100";
332 constant mux_7_7_f : std_logic_vector(2 downto 0) := "101";
333 constant mux_7_7_g : std_logic_vector(2 downto 0) := "110";
334
335 constant mux_8_1_a : std_logic_vector(2 downto 0) := "000";
336 constant mux_8_1_b : std_logic_vector(2 downto 0) := "001";
337 constant mux_8_1_c : std_logic_vector(2 downto 0) := "010";
338 constant mux_8_1_d : std_logic_vector(2 downto 0) := "011";
339 constant mux_8_1_e : std_logic_vector(2 downto 0) := "100";
340 constant mux_8_1_f : std_logic_vector(2 downto 0) := "101";
341 constant mux_8_1_g : std_logic_vector(2 downto 0) := "110";
342 constant mux_8_1_h : std_logic_vector(2 downto 0) := "111";
343 constant mux_8_2_a : std_logic_vector(2 downto 0) := "000";
344 constant mux_8_2_b : std_logic_vector(2 downto 0) := "001";
345 constant mux_8_2_c : std_logic_vector(2 downto 0) := "010";
346 constant mux_8_2_d : std_logic_vector(2 downto 0) := "011";
347 constant mux_8_2_e : std_logic_vector(2 downto 0) := "100";
348 constant mux_8_2_f : std_logic_vector(2 downto 0) := "101";
349 constant mux_8_2_g : std_logic_vector(2 downto 0) := "110";
350 constant mux_8_2_h : std_logic_vector(2 downto 0) := "111";
351 constant mux_8_3_a : std_logic_vector(2 downto 0) := "000";
352 constant mux_8_3_b : std_logic_vector(2 downto 0) := "001";
353 constant mux_8_3_c : std_logic_vector(2 downto 0) := "010";
354 constant mux_8_3_d : std_logic_vector(2 downto 0) := "011";
355 constant mux_8_3_e : std_logic_vector(2 downto 0) := "100";
356 constant mux_8_3_f : std_logic_vector(2 downto 0) := "101";
357 constant mux_8_3_g : std_logic_vector(2 downto 0) := "110";
358 constant mux_8_3_h : std_logic_vector(2 downto 0) := "111";
359 constant mux_8_4_a : std_logic_vector(2 downto 0) := "000";
360 constant mux_8_4_b : std_logic_vector(2 downto 0) := "001";
361 constant mux_8_4_c : std_logic_vector(2 downto 0) := "010";
362 constant mux_8_4_d : std_logic_vector(2 downto 0) := "011";
363 constant mux_8_4_e : std_logic_vector(2 downto 0) := "100";
364 constant mux_8_4_f : std_logic_vector(2 downto 0) := "101";
365 constant mux_8_4_g : std_logic_vector(2 downto 0) := "110";
366 constant mux_8_4_h : std_logic_vector(2 downto 0) := "111";
367
368
369 constant mux_9_1_a : std_logic_vector(3 downto 0) := "0000";
370 constant mux_9_1_b : std_logic_vector(3 downto 0) := "0001";
371 constant mux_9_1_c : std_logic_vector(3 downto 0) := "0010";
372 constant mux_9_1_d : std_logic_vector(3 downto 0) := "0011";
373 constant mux_9_1_e : std_logic_vector(3 downto 0) := "0100";
374 constant mux_9_1_f : std_logic_vector(3 downto 0) := "0101";
375 constant mux_9_1_g : std_logic_vector(3 downto 0) := "0110";
376 constant mux_9_1_h : std_logic_vector(3 downto 0) := "0111";
377 constant mux_9_1_i : std_logic_vector(3 downto 0) := "1000";
378 constant mux_9_2_a : std_logic_vector(3 downto 0) := "0000";
```

```
379 constant mux_9_2_b : std_logic_vector(3 downto 0) := "0001";
380 constant mux_9_2_c : std_logic_vector(3 downto 0) := "0010";
381 constant mux_9_2_d : std_logic_vector(3 downto 0) := "0011";
382 constant mux_9_2_e : std_logic_vector(3 downto 0) := "0100";
383 constant mux_9_2_f : std_logic_vector(3 downto 0) := "0101";
384 constant mux_9_2_g : std_logic_vector(3 downto 0) := "0110";
385 constant mux_9_2_h : std_logic_vector(3 downto 0) := "0111";
386 constant mux_9_2_i : std_logic_vector(3 downto 0) := "1000";
387 constant mux_9_3_a : std_logic_vector(3 downto 0) := "0000";
388 constant mux_9_3_b : std_logic_vector(3 downto 0) := "0001";
389 constant mux_9_3_c : std_logic_vector(3 downto 0) := "0010";
390 constant mux_9_3_d : std_logic_vector(3 downto 0) := "0011";
391 constant mux_9_3_e : std_logic_vector(3 downto 0) := "0100";
392 constant mux_9_3_f : std_logic_vector(3 downto 0) := "0101";
393 constant mux_9_3_g : std_logic_vector(3 downto 0) := "0110";
394 constant mux_9_3_h : std_logic_vector(3 downto 0) := "0111";
395 constant mux_9_3_i : std_logic_vector(3 downto 0) := "1000";
396
397 constant mux_11_1_a : std_logic_vector(3 downto 0) := "0000";
398 constant mux_11_1_b : std_logic_vector(3 downto 0) := "0001";
399 constant mux_11_1_c : std_logic_vector(3 downto 0) := "0010";
400 constant mux_11_1_d : std_logic_vector(3 downto 0) := "0011";
401 constant mux_11_1_e : std_logic_vector(3 downto 0) := "0100";
402 constant mux_11_1_f : std_logic_vector(3 downto 0) := "0101";
403 constant mux_11_1_g : std_logic_vector(3 downto 0) := "0110";
404 constant mux_11_1_h : std_logic_vector(3 downto 0) := "0111";
405 constant mux_11_1_i : std_logic_vector(3 downto 0) := "1000";
406 constant mux_11_1_j : std_logic_vector(3 downto 0) := "1001";
407 constant mux_11_1_k : std_logic_vector(3 downto 0) := "1010";
408 constant mux_11_2_a : std_logic_vector(3 downto 0) := "0000";
409 constant mux_11_2_b : std_logic_vector(3 downto 0) := "0001";
410 constant mux_11_2_c : std_logic_vector(3 downto 0) := "0010";
411 constant mux_11_2_d : std_logic_vector(3 downto 0) := "0011";
412 constant mux_11_2_e : std_logic_vector(3 downto 0) := "0100";
413 constant mux_11_2_f : std_logic_vector(3 downto 0) := "0101";
414 constant mux_11_2_g : std_logic_vector(3 downto 0) := "0110";
415 constant mux_11_2_h : std_logic_vector(3 downto 0) := "0111";
416 constant mux_11_2_i : std_logic_vector(3 downto 0) := "1000";
417 constant mux_11_2_j : std_logic_vector(3 downto 0) := "1001";
418 constant mux_11_2_k : std_logic_vector(3 downto 0) := "1010";
419
420 ---------SR1-8, M1-8--------------------
421 constant operation_suma_SR1  : std_logic := '0';
422 constant operation_suma_SR2  : std_logic := '0';
423 constant operation_suma_SR3  : std_logic := '0';
424 constant operation_suma_SR4  : std_logic := '0';
425 constant operation_suma_SR5  : std_logic := '0';
426 constant operation_suma_SR6  : std_logic := '0';
427 constant operation_suma_SR7  : std_logic := '0';
428 constant operation_suma_SR8  : std_logic := '0';
429 constant operation_resta_SR1 : std_logic := '1';
430 constant operation_resta_SR2 : std_logic := '1';
431 constant operation_resta_SR3 : std_logic := '1';
```

```
constant operation_resta_SR4 : std_logic := '1';
constant operation_resta_SR5 : std_logic := '1';
constant operation_resta_SR6 : std_logic := '1';
constant operation_resta_SR7 : std_logic := '1';
constant operation_resta_SR8 : std_logic := '1';

constant operation_nd_on_SR1  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR2  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR3  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR4  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR5  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR6  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR7  : std_logic := '1';---nuevo dato
constant operation_nd_on_SR8  : std_logic := '1';---nuevo dato
constant operation_nd_on_M1   : std_logic := '1';---nuevo dato
constant operation_nd_on_M2   : std_logic := '1';---nuevo dato
constant operation_nd_on_M3   : std_logic := '1';---nuevo dato
constant operation_nd_on_M4   : std_logic := '1';---nuevo dato
constant operation_nd_on_M5   : std_logic := '1';---nuevo dato
constant operation_nd_on_M6   : std_logic := '1';---nuevo dato
constant operation_nd_on_M7   : std_logic := '1';---nuevo dato
constant operation_nd_on_M8   : std_logic := '1';---nuevo dato
constant operation_nd_off_SR1 : std_logic := '0';
constant operation_nd_off_SR2 : std_logic := '0';
constant operation_nd_off_SR3 : std_logic := '0';
constant operation_nd_off_SR4 : std_logic := '0';
constant operation_nd_off_SR5 : std_logic := '0';
constant operation_nd_off_SR6 : std_logic := '0';
constant operation_nd_off_SR7 : std_logic := '0';
constant operation_nd_off_SR8 : std_logic := '0';
constant operation_nd_off_M1  : std_logic := '0';
constant operation_nd_off_M2  : std_logic := '0';
constant operation_nd_off_M3  : std_logic := '0';
constant operation_nd_off_M4  : std_logic := '0';
constant operation_nd_off_M5  : std_logic := '0';
constant operation_nd_off_M6  : std_logic := '0';
constant operation_nd_off_M7  : std_logic := '0';
constant operation_nd_off_M8  : std_logic := '0';

constant CE_on_SR1    : std_logic := '1';----clk enable
constant CE_on_SR2    : std_logic := '1';----clk enable
constant CE_on_SR3    : std_logic := '1';----clk enable
constant CE_on_SR4    : std_logic := '1';----clk enable
constant CE_on_SR5    : std_logic := '1';----clk enable
constant CE_on_SR6    : std_logic := '1';----clk enable
constant CE_on_SR7    : std_logic := '1';----clk enable
constant CE_on_SR8    : std_logic := '1';----clk enable
constant CE_on_M1     : std_logic := '1';----clk enable
constant CE_on_M2     : std_logic := '1';----clk enable
constant CE_on_M3     : std_logic := '1';----clk enable
constant CE_on_M4     : std_logic := '1';----clk enable
constant CE_on_M5     : std_logic := '1';----clk enable
constant CE_on_M6     : std_logic := '1';----clk enable
```

```
485 constant CE_on_M7     : std_logic := '1';----clk enable
486 constant CE_on_M8     : std_logic := '1';----clk enable
487 constant CE_off_M1    : std_logic := '0';----clk disable
488 constant CE_off_M2    : std_logic := '0';----clk disable
489 constant CE_off_M3    : std_logic := '0';----clk disable
490 constant CE_off_M4    : std_logic := '0';----clk disable
491 constant CE_off_M5    : std_logic := '0';----clk disable
492 constant CE_off_M6    : std_logic := '0';----clk disable
493 constant CE_off_M7    : std_logic := '0';----clk disable
494 constant CE_off_M8    : std_logic := '0';----clk disable
495
496 constant SCLR_on_SR1    : std_logic := '1';----Synchronous reset enable
497 constant SCLR_on_SR2    : std_logic := '1';----Synchronous reset enable
498 constant SCLR_on_SR3    : std_logic := '1';----Synchronous reset enable
499 constant SCLR_on_SR4    : std_logic := '1';----Synchronous reset enable
500 constant SCLR_on_SR5    : std_logic := '1';----Synchronous reset enable
501 constant SCLR_on_SR6    : std_logic := '1';----Synchronous reset enable
502 constant SCLR_on_SR7    : std_logic := '1';----Synchronous reset enable
503 constant SCLR_on_SR8    : std_logic := '1';----Synchronous reset enable
504 constant SCLR_on_M1     : std_logic := '1';----Synchronous reset enable
505 constant SCLR_on_M2     : std_logic := '1';----Synchronous reset enable
506 constant SCLR_on_M3     : std_logic := '1';----Synchronous reset enable
507 constant SCLR_on_M4     : std_logic := '1';----Synchronous reset enable
508 constant SCLR_on_M5     : std_logic := '1';----Synchronous reset enable
509 constant SCLR_on_M6     : std_logic := '1';----Synchronous reset enable
510 constant SCLR_on_M7     : std_logic := '1';----Synchronous reset enable
511 constant SCLR_on_M8     : std_logic := '1';----Synchronous reset enable
512 constant SCLR_off_SR1   : std_logic := '0';----Synchronous reset disable
513 constant SCLR_off_SR2   : std_logic := '0';----Synchronous reset disable
514 constant SCLR_off_SR3   : std_logic := '0';----Synchronous reset disable
515 constant SCLR_off_SR4   : std_logic := '0';----Synchronous reset disable
516 constant SCLR_off_SR5   : std_logic := '0';----Synchronous reset disable
517 constant SCLR_off_SR6   : std_logic := '0';----Synchronous reset disable
518 constant SCLR_off_SR7   : std_logic := '0';----Synchronous reset disable
519 constant SCLR_off_SR8   : std_logic := '0';----Synchronous reset disable
520 constant SCLR_off_M1    : std_logic := '0';----Synchronous reset disable
521 constant SCLR_off_M2    : std_logic := '0';----Synchronous reset disable
522 constant SCLR_off_M3    : std_logic := '0';----Synchronous reset disable
523 constant SCLR_off_M4    : std_logic := '0';----Synchronous reset disable
524 constant SCLR_off_M5    : std_logic := '0';----Synchronous reset disable
525 constant SCLR_off_M6    : std_logic := '0';----Synchronous reset disable
526 constant SCLR_off_M7    : std_logic := '0';----Synchronous reset disable
527 constant SCLR_off_M8    : std_logic := '0';----Synchronous reset disable
528 constant desocupado     : std_logic := '0';
529 constant ocupado        : std_logic := '1';
530
531 constant s0 : std_logic_vector(N downto 0) := "00000" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
532 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
533 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
534 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
535 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
536 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
537 R58_OFF & R59_OFF & mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_a & mux_8_1_a &
```

```
538 mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_a & mux_7_5_a
539 & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a
540 & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a
541 & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a
542 & mux_2_8_a & mux_2_9_a & mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a &
543 mux_2_15_a & mux_2_16_a & mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a &
544 mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a &
545 mux_2_29_a & mux_2_30_a & mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_b & operation_suma_SR1
546 & operation_suma_SR2 & operation_suma_SR3 & operation_suma_SR4 & operation_suma_SR5 &
547 operation_suma_SR6 & operation_suma_SR7 & operation_suma_SR8 & SCLR_on_SR1 & SCLR_on_SR2 &
548 SCLR_on_SR3 & SCLR_on_SR4 & SCLR_on_SR5 & SCLR_on_SR6 & SCLR_on_SR7 & SCLR_on_SR8 &
549 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 &
550 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 &
551 SCLR_on_M1 & SCLR_on_M2 & SCLR_on_M3 & SCLR_on_M4 & SCLR_on_M5 & SCLR_on_M6 & SCLR_on_M7 &
552 SCLR_on_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 &
553 operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
554 operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6 &
555 CE_on_M7 & CE_on_M8 & desocupado ;
556 constant s1 : std_logic_vector(N downto 0) := "00001" & R2_ON & R3_ON & R4_ON & R5_ON &
557 R6_ON & R7_ON & R8_ON & R9_ON & R10_ON & R11_ON & R12_ON & R13_ON & R14_ON & R15_ON & R16_ON
558 & R17_ON & R18_ON & R19_ON & R20_ON & R21_ON & R28_ON & R29_OFF & R30_OFF & R31_OFF & R32_OFF
559 & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF & R40_OFF & R41_OFF &
560 R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF & R49_OFF & R50_OFF &
561 R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF & R58_OFF & R59_OFF &
562 mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_a & mux_8_1_a & mux_8_2_a & mux_8_3_a
563 & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_a & mux_7_5_a & mux_7_6_a & mux_7_7_a
564 & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a
565 & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a
566 & mux_2_2_b & mux_2_3_b & mux_2_4_b & mux_2_5_a & mux_2_6_a & mux_2_7_b & mux_2_8_b & mux_2_9_b
567 & mux_2_10_a & mux_2_11_b & mux_2_12_b & mux_2_13_b & mux_2_14_a & mux_2_15_a & mux_2_16_a
568 & mux_2_17_b & mux_2_18_b & mux_2_19_b & mux_2_20_a & mux_2_21_a & mux_2_22_b & mux_2_23_b
569 & mux_2_24_b & mux_2_25_b & mux_2_26_a & mux_2_27_a & mux_2_28_b & mux_2_29_b & mux_2_30_b
570 & mux_2_31_a & mux_2_32_b & mux_2_33_b & mux_2_34_b & operation_suma_SR1  & operation_suma_SR2
571 & operation_suma_SR3 & operation_suma_SR4 & operation_suma_SR5 & operation_suma_SR6 &
572 operation_suma_SR7 & operation_suma_SR8 & SCLR_on_SR1 & SCLR_on_SR2 & SCLR_on_SR3 &
573 SCLR_on_SR4 & SCLR_on_SR5 & SCLR_on_SR6 & SCLR_on_SR7 & SCLR_on_SR8 & operation_nd_off_SR1
574 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 & operation_nd_off_SR5
575 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 & SCLR_on_M1 & SCLR_on_M2
576 & SCLR_on_M3 & SCLR_on_M4 & SCLR_on_M5 & SCLR_on_M6 & SCLR_on_M7 & SCLR_on_M8 &
577 operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 & operation_nd_off_M4 &
578 operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 & operation_nd_off_M8 &
579 CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6 & CE_on_M7 & CE_on_M8 & ocupado;
580 constant s2 : std_logic_vector(N downto 0) := "00010" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
581 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
582 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
583 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
584 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
585 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
586 R58_OFF & R59_OFF & mux_11_1_i & mux_11_2_i & mux_9_1_i & mux_9_2_i & mux_9_3_h & mux_8_1_h
587 & mux_8_2_h & mux_8_3_a & mux_8_4_a & mux_7_1_g & mux_7_2_g & mux_7_3_g & mux_7_4_g & mux_7_5_g
588 & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a
589 & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a
590 & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a
```

```
591 & mux_2_8_a & mux_2_9_a & mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a &
592 mux_2_15_a & mux_2_16_a & mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a &
593 mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a &
594 mux_2_29_a & mux_2_30_a & mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1
595 & operation_suma_SR2 & operation_suma_SR3 & operation_suma_SR4 & operation_suma_SR5 &
596 operation_suma_SR6 & operation_suma_SR7 & operation_suma_SR8 & SCLR_off_SR1 & SCLR_off_SR2
597 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
598 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 &
599 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 &
600 SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 & SCLR_off_M7
601 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 &
602 operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
603 operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6 &
604 CE_on_M7 & CE_on_M8 & ocupado ;
605 constant s3 : std_logic_vector(N downto 0) := "00011" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
606 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
607 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
608 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
609 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
610 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
611 R58_OFF & R59_OFF & mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_a & mux_8_1_a &
612 mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_a & mux_7_5_a
613 & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a
614 & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a
615 & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a
616 & mux_2_8_a & mux_2_9_a & mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a &
617 mux_2_15_a & mux_2_16_a & mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a &
618 mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a &
619 mux_2_29_a & mux_2_30_a & mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a &
620 operation_suma_SR1 & operation_suma_SR2 & operation_suma_SR3 & operation_suma_SR4 &
621 operation_suma_SR5 & operation_suma_SR6 & operation_resta_SR7 & operation_resta_SR8 &
622 SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 &
623 SCLR_off_SR7 & SCLR_off_SR8 & operation_nd_on_SR1 & operation_nd_on_SR2 & operation_nd_on_SR3
624 & operation_nd_on_SR4 & operation_nd_on_SR5 & operation_nd_on_SR6 & operation_nd_on_SR7 &
625 operation_nd_on_SR8 & SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 &
626 SCLR_off_M6 & SCLR_off_M7 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 &
627 operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
628 operation_nd_off_M7 & operation_nd_off_M8 & CE_off_M1 & CE_off_M2 & CE_off_M3 & CE_off_M4 &
629 CE_off_M5 & CE_off_M6 & CE_off_M7 & CE_off_M8 & ocupado ;
630 constant s3_w : std_logic_vector(N downto 0) := "00100" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
631 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
632 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_ON  & R30_ON  &
633 R31_ON  & R32_ON  & R33_ON  & R34_ON  & R35_ON  & R36_ON  & R37_OFF & R38_OFF & R39_OFF &
634 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
635 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
636 R58_OFF & R59_OFF & mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_a & mux_8_1_a
637 & mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_a &
638 mux_7_5_a & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a &
639 mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a &
640 mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a &
641 mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a &
642 mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a & mux_2_17_a &
643 mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a &
```

```
644 mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a &
645 mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1 & operation_suma_SR2 &
646 operation_suma_SR3 & operation_suma_SR4 & operation_suma_SR5 & operation_suma_SR6 &
647 operation_resta_SR7 & operation_resta_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
648 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
649 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4
650 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8
651 & SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
652 SCLR_off_M7 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
653 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7
654 & operation_nd_off_M8 & CE_off_M1 & CE_off_M2 & CE_off_M3 & CE_off_M4 & CE_off_M5 & CE_off_M6
655 & CE_off_M7 & CE_off_M8 & ocupado ;
656 constant s4 : std_logic_vector(N downto 0) := "00101" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
657 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
658 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
659 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
660 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
661 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
662 R58_OFF & R59_OFF & mux_11_1_d & mux_11_2_d & mux_9_1_d & mux_9_2_d & mux_9_3_d & mux_8_1_c &
663 mux_8_2_d & mux_8_3_d & mux_8_4_d & mux_7_1_c & mux_7_2_c & mux_7_3_c & mux_7_4_b & mux_7_5_b
664 & mux_7_6_d & mux_7_7_d & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a & mux_4_3_a &
665 mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a &
666 mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a &
667 mux_2_5_a & mux_2_6_b & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a & mux_2_11_a &
668 mux_2_12_a & mux_2_13_a & mux_2_14_b & mux_2_15_a & mux_2_16_a & mux_2_17_a & mux_2_18_a &
669 mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a &
670 mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a & mux_2_32_a &
671 mux_2_33_a & mux_2_34_a & operation_resta_SR1 & operation_resta_SR2 & operation_resta_SR3 &
672 operation_resta_SR4 & operation_resta_SR5 & operation_resta_SR6 & operation_resta_SR7 &
673 operation_resta_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 &
674 SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 & operation_nd_on_SR1 &
675 operation_nd_on_SR2 & operation_nd_on_SR3 & operation_nd_on_SR4 & operation_nd_on_SR5 &
676 operation_nd_on_SR6 & operation_nd_on_SR7 & operation_nd_on_SR8 & SCLR_off_M1 &
677 SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 & SCLR_off_M7 &
678 SCLR_off_M8 & operation_nd_on_M1 & operation_nd_on_M2 & operation_nd_off_M3 &
679 operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
680 operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6
681 & CE_on_M7 & CE_on_M8 & ocupado ;
682 constant s4_w : std_logic_vector(N downto 0) := "00110" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
683 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
684 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
685 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_ON & R38_ON &
686 R39_ON & R40_ON & R41_ON & R42_ON & R43_ON & R44_ON & R45_ON & R46_ON & R47_OFF &
687 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
688 R57_OFF & R58_OFF & R59_OFF & mux_11_1_d & mux_11_2_d & mux_9_1_d & mux_9_2_d & mux_9_3_d &
689 mux_8_1_c & mux_8_2_d & mux_8_3_d & mux_8_4_d & mux_7_1_c & mux_7_2_c & mux_7_3_c & mux_7_4_b
690 & mux_7_5_b & mux_7_6_d & mux_7_7_d & mux_5_1_a & mux_5_2_a & mux_4_1_a & mux_4_2_a &
691 mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a &
692 mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a &
693 mux_2_4_a & mux_2_5_a & mux_2_6_b & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a &
694 mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_b & mux_2_15_a & mux_2_16_a & mux_2_17_a &
695 mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a
696 & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a
```

```
697 & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_resta_SR1 & operation_resta_SR2 &
698 operation_resta_SR3 & operation_resta_SR4 & operation_resta_SR5 & operation_resta_SR6 &
699 operation_resta_SR7 & operation_resta_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
700 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
701 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4
702 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8
703 & SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
704 SCLR_off_M7 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
705 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
706 operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6
707 & CE_on_M7 & CE_on_M8 & ocupado ;
708 constant s5 : std_logic_vector(N downto 0) := "00111" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
709 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
710 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
711 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
712 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
713 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
714 R58_OFF & R59_OFF & mux_11_1_j & mux_11_2_j & mux_9_1_a & mux_9_2_a & mux_9_3_a & mux_8_1_e
715 & mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_e & mux_7_4_a &
716 mux_7_5_a & mux_7_6_a & mux_7_7_a & mux_5_1_b & mux_5_2_b & mux_4_1_b & mux_4_2_b & mux_4_3_a
717 & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a &
718 mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a
719 & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_b & mux_2_11_a & mux_2_12_a &
720 mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_b & mux_2_17_a & mux_2_18_a & mux_2_19_a
721 & mux_2_20_a & mux_2_21_b & mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a
722 & mux_2_27_b & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_b & mux_2_32_a & mux_2_33_a
723 & mux_2_34_a & operation_suma_SR1 & operation_resta_SR2 & operation_resta_SR3 &
724 operation_suma_SR4 & operation_suma_SR5 & operation_suma_SR6 & operation_suma_SR7 &
725 operation_suma_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 &
726 SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 & operation_nd_off_SR1 &
727 operation_nd_on_SR2 & operation_nd_on_SR3 & operation_nd_off_SR4 & operation_nd_off_SR5 &
728 operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 & SCLR_off_M1 &
729 SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 & SCLR_off_M7 &
730 SCLR_off_M8 & operation_nd_on_M1 & operation_nd_on_M2 & operation_nd_on_M3 &
731 operation_nd_on_M4 & operation_nd_on_M5 & operation_nd_on_M6 & operation_nd_on_M7
732 & operation_nd_on_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6
733 & CE_on_M7 & CE_on_M8 & ocupado ;
734 constant s5_w : std_logic_vector(N downto 0) := "01000" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
735 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
736 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
737 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
738 R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_ON & R46_ON & R47_ON &
739 R48_ON & R49_ON & R50_ON & R51_ON & R52_ON & R53_ON & R54_ON & R55_OFF & R56_OFF &
740 R57_OFF & R58_OFF & R59_OFF & mux_11_1_j & mux_11_2_j & mux_9_1_a & mux_9_2_a & mux_9_3_a &
741 mux_8_1_e & mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_e & mux_7_4_a
742 & mux_7_5_a & mux_7_6_a & mux_7_7_a & mux_5_1_b & mux_5_2_b & mux_4_1_b & mux_4_2_b &
743 mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a & mux_3_3_a & mux_3_4_a
744 & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a & mux_2_3_a & mux_2_4_a &
745 mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_b & mux_2_11_a &
746 mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_b & mux_2_17_a & mux_2_18_a
747 & mux_2_19_a & mux_2_20_a & mux_2_21_b & mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a
748 & mux_2_26_a & mux_2_27_b & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_b & mux_2_32_a
749 & mux_2_33_a & mux_2_34_a & operation_suma_SR1 & operation_resta_SR2 & operation_resta_SR3
```

```
750 & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  & operation_suma_SR7  &
751 operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 &
752 SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 & operation_nd_off_SR1 &
753 operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 & operation_nd_off_SR5 &
754 operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 & SCLR_off_M1 &
755 SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 & SCLR_off_M7 &
756 SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 &
757 operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
758 operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6
759 & CE_on_M7  & CE_on_M8  & ocupado ;
760 constant s6 : std_logic_vector(N downto 0) := "01001" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
761 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
762 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
763 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
764 R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
765 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
766 R57_OFF & R58_OFF & R59_OFF & mux_11_1_f & mux_11_2_f & mux_9_1_f & mux_9_2_f & mux_9_3_a
767 & mux_8_1_a & mux_8_2_a & mux_8_3_e & mux_8_4_e & mux_7_1_a & mux_7_2_a & mux_7_3_a &
768 mux_7_4_a & mux_7_5_a & mux_7_6_e & mux_7_7_e & mux_5_1_c & mux_5_2_c & mux_4_1_c &
769 mux_4_2_c & mux_4_3_b & mux_4_4_b & mux_4_5_b & mux_4_6_b & mux_3_1_b & mux_3_2_b &
770 mux_3_3_b & mux_3_4_b & mux_3_5_b & mux_3_6_b & mux_3_7_b & mux_2_1_a & mux_2_2_a &
771 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
772 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
773 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_b & mux_2_21_b & mux_2_22_a & mux_2_23_a &
774 mux_2_24_a & mux_2_25_a & mux_2_26_b & mux_2_27_b & mux_2_28_a & mux_2_29_a & mux_2_30_a &
775 mux_2_31_b & mux_2_32_a & mux_2_33_a & mux_2_34_b & operation_suma_SR1  & operation_suma_SR2
776 & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  &
777 operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
778 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
779 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_on_SR3  & operation_nd_on_SR4  &
780 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_on_SR7  & operation_nd_on_SR8 &
781 SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
782 SCLR_off_M7 & SCLR_off_M8 & operation_nd_on_M1  & operation_nd_on_M2  & operation_nd_on_M3
783 & operation_nd_on_M4  & operation_nd_on_M5  & operation_nd_on_M6  & operation_nd_on_M7  &
784 operation_nd_on_M8  & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6
785 & CE_on_M7  & CE_on_M8  & ocupado ;
786 constant s6_w : std_logic_vector(N downto 0) := "01010" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
787 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
788 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
789 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
790 R39_OFF & R40_OFF & R41_ON  & R42_ON  & R43_ON  & R44_ON  & R45_OFF & R46_OFF & R47_ON  &
791 R48_ON  & R49_ON  & R50_ON  & R51_ON  & R52_ON  & R53_ON  & R54_ON  & R55_OFF & R56_OFF &
792 R57_OFF & R58_OFF & R59_OFF & mux_11_1_f & mux_11_2_f & mux_9_1_f & mux_9_2_f & mux_9_3_a &
793 mux_8_1_a & mux_8_2_a & mux_8_3_e & mux_8_4_e & mux_7_1_a & mux_7_2_a & mux_7_3_a &
794 mux_7_4_a & mux_7_5_a & mux_7_6_e & mux_7_7_e & mux_5_1_c & mux_5_2_c & mux_4_1_c &
795 mux_4_2_c & mux_4_3_b & mux_4_4_b & mux_4_5_b & mux_4_6_b & mux_3_1_b & mux_3_2_b &
796 mux_3_3_b & mux_3_4_b & mux_3_5_b & mux_3_6_b & mux_3_7_b & mux_2_1_a & mux_2_2_a &
797 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
798 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
799 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_b & mux_2_21_b & mux_2_22_a & mux_2_23_a &
800 mux_2_24_a & mux_2_25_a & mux_2_26_b & mux_2_27_b & mux_2_28_a & mux_2_29_a & mux_2_30_a &
801 mux_2_31_b & mux_2_32_a & mux_2_33_a & mux_2_34_b & operation_suma_SR1  & operation_suma_SR2
802 & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  &
```

```
803 operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
804 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
805 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 &
806 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 &
807 SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
808 SCLR_off_M7 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
809 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
810 operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6 &
811 CE_on_M7  & CE_on_M8  & ocupado ;
812 constant s7 : std_logic_vector(N downto 0) := "01011" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
813 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
814 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
815 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
816 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
817 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
818 R58_OFF & R59_OFF & mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_e & mux_8_1_a
819 & mux_8_2_e & mux_8_3_g & mux_8_4_g & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_d &
820 mux_7_5_d & mux_7_6_a & mux_7_7_a & mux_5_1_d & mux_5_2_e & mux_4_1_d & mux_4_2_c &
821 mux_4_3_c & mux_4_4_c & mux_4_5_c & mux_4_6_c & mux_3_1_c & mux_3_2_c & mux_3_3_c &
822 mux_3_4_c & mux_3_5_c & mux_3_6_c & mux_3_7_c & mux_2_1_a & mux_2_2_a & mux_2_3_a &
823 mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a &
824 mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a & mux_2_17_a &
825 mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a &
826 mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a &
827 mux_2_32_a & mux_2_33_a & mux_2_34_b & operation_suma_SR1  & operation_suma_SR2  &
828 operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  &
829 operation_resta_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
830 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
831 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4
832 & operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_on_SR7  & operation_nd_off_SR8
833 & SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
834 SCLR_off_M7 & SCLR_off_M8 & operation_nd_on_M1  & operation_nd_on_M2  & operation_nd_on_M3 &
835 operation_nd_on_M4  & operation_nd_on_M5  & operation_nd_on_M6  & operation_nd_on_M7 &
836 operation_nd_on_M8  & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6 &
837 CE_on_M7  & CE_on_M8  & ocupado ;
838 constant s7_w : std_logic_vector(N downto 0) := "01100" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
839 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
840 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
841 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
842 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
843 R49_ON  & R50_ON  & R51_ON  & R52_ON  & R53_ON  & R54_ON  & R55_ON  & R56_ON  & R57_ON  &
844 R58_ON  & R59_ON  & mux_11_1_a & mux_11_2_a & mux_9_1_a & mux_9_2_a & mux_9_3_e & mux_8_1_a
845 & mux_8_2_e & mux_8_3_g & mux_8_4_g & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_d &
846 mux_7_5_d & mux_7_6_a & mux_7_7_a & mux_5_1_d & mux_5_2_e & mux_4_1_d & mux_4_2_c &
847 mux_4_3_c & mux_4_4_c & mux_4_5_c & mux_4_6_c & mux_3_1_c & mux_3_2_c & mux_3_3_c &
848 mux_3_4_c & mux_3_5_c & mux_3_6_c & mux_3_7_c & mux_2_1_a & mux_2_2_a & mux_2_3_a &
849 mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a &
850 mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a & mux_2_17_a &
851 mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a &
852 mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a &
853 mux_2_32_a & mux_2_33_a & mux_2_34_b & operation_suma_SR1  & operation_suma_SR2  &
854 operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  &
855 operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
```

```
856 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
857 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 &
858 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 &
859 SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_off_M5 & SCLR_off_M6 &
860 SCLR_off_M7 & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
861 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
862 operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6 &
863 CE_on_M7 & CE_on_M8 & ocupado ;
864 constant s8 : std_logic_vector(N downto 0) := "01101" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
865 R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF & R15_OFF
866 & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF & R30_OFF &
867 R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF & R39_OFF &
868 R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF & R48_OFF &
869 R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF & R57_OFF &
870 R58_OFF & R59_OFF & mux_11_1_e & mux_11_2_e & mux_9_1_a & mux_9_2_a & mux_9_3_e & mux_8_1_a
871 & mux_8_2_e & mux_8_3_b & mux_8_4_b & mux_7_1_a & mux_7_2_a & mux_7_3_a & mux_7_4_d &
872 mux_7_5_d & mux_7_6_b & mux_7_7_b & mux_5_1_e & mux_5_2_d & mux_4_1_d & mux_4_2_d &
873 mux_4_3_d & mux_4_4_d & mux_4_5_d & mux_4_6_d & mux_3_1_c & mux_3_2_c & mux_3_3_a &
874 mux_3_4_a & mux_3_5_c & mux_3_6_a & mux_3_7_b & mux_2_1_b & mux_2_2_a & mux_2_3_a &
875 mux_2_4_a & mux_2_5_b & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a & mux_2_10_a &
876 mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_b & mux_2_16_a & mux_2_17_a &
877 mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a & mux_2_24_a &
878 mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a &
879 mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1 & operation_suma_SR2 &
880 operation_suma_SR3 & operation_suma_SR4 & operation_suma_SR5 & operation_suma_SR6 &
881 operation_resta_SR7 & operation_suma_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
882 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
883 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_on_SR3 & operation_nd_off_SR4 &
884 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_on_SR7 & operation_nd_on_SR8 &
885 SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_on_M5 & SCLR_off_M6 &
886 SCLR_on_M7 & SCLR_off_M8 & operation_nd_on_M1 & operation_nd_on_M2 & operation_nd_on_M3 &
887 operation_nd_on_M4 & operation_nd_off_M5 & operation_nd_on_M6 & operation_nd_off_M7 &
888 operation_nd_on_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 & CE_on_M5 & CE_on_M6 &
889 CE_on_M7 & CE_on_M8 & ocupado ;
890 constant s8_w : std_logic_vector(N downto 0) := "01110" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
891 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
892 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
893 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_ON & R36_ON & R37_ON & R38_ON &
894 R39_ON & R40_ON & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
895 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_ON & R55_ON & R56_ON &
896 R57_OFF & R58_OFF & R59_OFF & mux_11_1_e & mux_11_2_e & mux_9_1_a & mux_9_2_a & mux_9_3_e &
897 mux_8_1_a & mux_8_2_e & mux_8_3_b & mux_8_4_b & mux_7_1_a & mux_7_2_a & mux_7_3_a &
898 mux_7_4_d & mux_7_5_d & mux_7_6_b & mux_7_7_b & mux_5_1_e & mux_5_2_d & mux_4_1_d &
899 mux_4_2_d & mux_4_3_d & mux_4_4_d & mux_4_5_d & mux_4_6_d & mux_3_1_c & mux_3_2_c &
900 mux_3_3_a & mux_3_4_a & mux_3_5_c & mux_3_6_a & mux_3_7_b & mux_2_1_b & mux_2_2_a &
901 mux_2_3_a & mux_2_4_a & mux_2_5_b & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
902 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_b & mux_2_16_a &
903 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
904 mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
905 mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1 &
906 operation_suma_SR2 & operation_resta_SR3 & operation_suma_SR4 & operation_suma_SR5 &
907 operation_suma_SR6 & operation_suma_SR7 & operation_resta_SR8 & SCLR_off_SR1 &
908 SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
```

```
909  SCLR_off_SR8 & operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 &
910  operation_nd_off_SR4 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 &
911  operation_nd_off_SR8 & SCLR_off_M1 & SCLR_off_M2 & SCLR_off_M3 & SCLR_off_M4 & SCLR_on_M5 &
912  SCLR_off_M6 & SCLR_on_M7  & SCLR_off_M8 & operation_nd_off_M1 & operation_nd_off_M2 &
913  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
914  operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
915  CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
916  constant s9 : std_logic_vector(N downto 0) := "01111" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
917  R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
918  R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
919  R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
920  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
921  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
922  R57_OFF & R58_OFF & R59_OFF & mux_11_1_b & mux_11_2_b & mux_9_1_b & mux_9_2_b & mux_9_3_b &
923  mux_8_1_a & mux_8_2_b & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a &
924  mux_7_4_b & mux_7_5_b & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a &
925  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
926  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
927  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
928  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
929  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
930  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
931  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1 & operation_suma_SR2 &
932  operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  &
933  operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
934  SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
935  operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_on_SR3  & operation_nd_on_SR4  &
936  operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_off_SR7 & operation_nd_off_SR8 &
937  SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4 & SCLR_on_M5 & SCLR_on_M6  & SCLR_on_M7
938 & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 &
939  operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
940  operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6  &
941  CE_on_M7  & CE_on_M8  & ocupado ;
942  constant s9_w : std_logic_vector(N downto 0) := "10000" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
943 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
944  R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
945  R30_OFF & R31_ON  & R32_ON  & R33_ON  & R34_ON  & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
946  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
947  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
948  R57_OFF & R58_OFF & R59_OFF & mux_11_1_b & mux_11_2_b & mux_9_1_b & mux_9_2_b & mux_9_3_b &
949  mux_8_1_a & mux_8_2_b & mux_8_3_a & mux_8_4_a & mux_7_1_a & mux_7_2_a & mux_7_3_a &
950  mux_7_4_b & mux_7_5_b & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a &
951  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
952  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
953  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
954  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
955  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
956  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
957  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1  &
958  operation_suma_SR2 & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  &
959  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
960  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
961  SCLR_off_SR8 & operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 &
```

```
962  operation_nd_off_SR4 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 &
963  operation_nd_off_SR8 & SCLR_on_M1  & SCLR_on_M2 & SCLR_on_M3 & SCLR_on_M4 & SCLR_on_M5 &
964  SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
965  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
966  operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
967  CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
968  constant s10 : std_logic_vector(N downto 0) := "10001" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
969  & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
970  R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
971  R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
972  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
973  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
974  R57_OFF & R58_OFF & R59_OFF & mux_11_1_k & mux_11_2_k & mux_9_1_e & mux_9_2_e & mux_9_3_c &
975  mux_8_1_b & mux_8_2_c & mux_8_3_f & mux_8_4_f & mux_7_1_b & mux_7_2_b & mux_7_3_b &
976  mux_7_4_c & mux_7_5_c & mux_7_6_f & mux_7_7_f & mux_5_1_a & mux_5_2_a & mux_4_1_a &
977  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
978  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
979  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
980  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
981  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
982  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
983  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1  &
984  operation_suma_SR2  & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  &
985  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
986  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
987  SCLR_off_SR8 & operation_nd_on_SR1  & operation_nd_on_SR2  & operation_nd_on_SR3  &
988  operation_nd_on_SR4  & operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_on_SR7 &
989  operation_nd_on_SR8  & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3 & SCLR_on_M4 & SCLR_on_M5 &
990  SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
991  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
992  operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
993  CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
994  constant s10_w : std_logic_vector(N downto 0) := "10010" & R2_OFF & R3_OFF & R4_OFF &
995  R5_OFF & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF
996  & R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_ON  &
997  R30_ON  & R31_OFF & R32_OFF & R33_ON  & R34_ON  & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
998  R39_OFF & R40_ON  & R41_OFF & R42_OFF & R43_ON  & R44_ON  & R45_ON  & R46_OFF & R47_OFF &
999  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1000 R57_OFF & R58_OFF & R59_OFF & mux_11_1_k & mux_11_2_k & mux_9_1_e & mux_9_2_e & mux_9_3_c &
1001 mux_8_1_b & mux_8_2_c & mux_8_3_f & mux_8_4_f & mux_7_1_b & mux_7_2_b & mux_7_3_b &
1002 mux_7_4_c & mux_7_5_c & mux_7_6_f & mux_7_7_f & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1003 mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1004 mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1005 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1006 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1007 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1008 mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1009 mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1 & operation_suma_SR2
1010 & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6
1011 & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
1012 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
1013 operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 &
1014 operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 &
```

```
1015  SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5  & SCLR_on_M6  &
1016  SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
1017  & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
1018  operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4 & CE_on_M5 & CE_on_M6 &
1019  CE_on_M7  & CE_on_M8  & ocupado ;
1020  constant s11 : std_logic_vector(N downto 0) := "10011" & R2_OFF & R3_OFF & R4_OFF & R5_OFF &
1021  R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
1022  R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
1023  R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1024  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1025  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1026  R57_OFF & R58_OFF & R59_OFF & mux_11_1_c & mux_11_2_c & mux_9_1_c & mux_9_2_c & mux_9_3_f &
1027  mux_8_1_d & mux_8_2_f & mux_8_3_c & mux_8_4_c & mux_7_1_d & mux_7_2_d & mux_7_3_d &
1028  mux_7_4_e & mux_7_5_e & mux_7_6_c & mux_7_7_c & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1029  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1030  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1031  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1032  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1033  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1034  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1035  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_resta_SR1 &
1036  operation_resta_SR2 & operation_resta_SR3 & operation_resta_SR4 & operation_suma_SR5  &
1037  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
1038  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1039  SCLR_off_SR8 & operation_nd_on_SR1  & operation_nd_on_SR2  & operation_nd_on_SR3  &
1040  operation_nd_on_SR4  & operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_on_SR7 &
1041  operation_nd_on_SR8  & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5 &
1042  SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
1043  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
1044  operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1 & CE_on_M2 & CE_on_M3 & CE_on_M4 &
1045  CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1046  constant s11_w : std_logic_vector(N downto 0) := "10100" & R2_ON  & R3_ON  & R4_OFF &
1047  R5_OFF & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF &
1048  R14_OFF & R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF &
1049  R29_OFF & R30_OFF & R31_ON  & R32_ON  & R33_OFF & R34_OFF & R35_ON  & R36_ON  & R37_ON  &
1050  R38_ON  & R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF &
1051  R47_OFF & R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF &
1052  R56_OFF & R57_OFF & R58_OFF & R59_OFF & mux_11_1_c & mux_11_2_c & mux_9_1_c & mux_9_2_c &
1053  mux_9_3_f & mux_8_1_d & mux_8_2_f & mux_8_3_c & mux_8_4_c & mux_7_1_d & mux_7_2_d &
1054  mux_7_3_d & mux_7_4_e & mux_7_5_e & mux_7_6_c & mux_7_7_c & mux_5_1_a & mux_5_2_a &
1055  mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a &
1056  mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a &
1057  mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a &
1058  mux_2_9_a & mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a &
1059  mux_2_16_a & mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a &
1060  mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a &
1061  mux_2_30_a & mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_resta_SR1 &
1062  operation_resta_SR2 & operation_resta_SR3 & operation_resta_SR4 & operation_suma_SR5  &
1063  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
1064  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1065  SCLR_off_SR8 & operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 &
1066  operation_nd_off_SR4 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7
1067  & operation_nd_off_SR8 & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5
```

```
1068 & SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
1069 operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
1070 operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
1071 CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1072 constant s12 : std_logic_vector(N downto 0) := "10101" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
1073 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
1074 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
1075 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1076 R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1077 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1078 R57_OFF & R58_OFF & R59_OFF & mux_11_1_g & mux_11_2_g & mux_9_1_g & mux_9_2_g & mux_9_3_g &
1079 mux_8_1_f & mux_8_2_g & mux_8_3_h & mux_8_4_h & mux_7_1_e & mux_7_2_e & mux_7_3_d &
1080 mux_7_4_f & mux_7_5_f & mux_7_6_g & mux_7_7_g & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1081 mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1082 mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1083 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1084 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1085 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1086 mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1087 mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_resta_SR1 & operation_suma_SR2
1088 & operation_resta_SR3 & operation_suma_SR4  & operation_resta_SR5 & operation_suma_SR6  &
1089 operation_suma_SR7  & operation_resta_SR8 & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 &
1090 SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 &
1091 operation_nd_on_SR1  & operation_nd_on_SR2  & operation_nd_on_SR3  & operation_nd_on_SR4  &
1092 operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_on_SR7  & operation_nd_on_SR8  &
1093 SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5  & SCLR_on_M6  &
1094 SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3
1095 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
1096 operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6
1097 & CE_on_M7  & CE_on_M8  & ocupado ;
1098 constant s12_w : std_logic_vector(N downto 0) := "10110" & R2_OFF & R3_OFF & R4_ON  & R5_ON
1099 & R6_ON  & R7_ON  & R8_ON  & R9_ON  & R10_ON  & R11_ON  & R12_OFF & R13_OFF & R14_OFF &
1100 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
1101 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1102 R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1103 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1104 R57_OFF & R58_OFF & R59_OFF & mux_11_1_g & mux_11_2_g & mux_9_1_g & mux_9_2_g & mux_9_3_g &
1105 mux_8_1_f & mux_8_2_g & mux_8_3_h & mux_8_4_h & mux_7_1_e & mux_7_2_e & mux_7_3_d &
1106 mux_7_4_f & mux_7_5_f & mux_7_6_g & mux_7_7_g & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1107 mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1108 mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1109 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1110 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1111 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1112 mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1113 mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_resta_SR1 &
1114 operation_suma_SR2  & operation_resta_SR3 & operation_suma_SR4  & operation_resta_SR5 &
1115 operation_suma_SR6  & operation_suma_SR7  & operation_resta_SR8 & SCLR_off_SR1 &
1116 SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1117 SCLR_off_SR8 & operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 &
1118 operation_nd_off_SR4 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7
1119 & operation_nd_off_SR8 & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  &
1120 SCLR_on_M5  & SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 &
```

```
1121  operation_nd_off_M2 & operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 &
1122  operation_nd_off_M6 & operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  &
1123  CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1124  constant s13 : std_logic_vector(N downto 0) := "10111" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
1125  & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
1126  R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
1127  R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1128  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1129  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1130  R57_OFF & R58_OFF & R59_OFF & mux_11_1_h & mux_11_2_h & mux_9_1_h & mux_9_2_h & mux_9_3_i
1131  & mux_8_1_g & mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_f & mux_7_2_f & mux_7_3_f &
1132  mux_7_4_g & mux_7_5_a & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1133  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1134  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1135  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1136  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1137  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1138  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1139  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1  &
1140  operation_resta_SR2 & operation_suma_SR3  & operation_resta_SR4 & operation_suma_SR5  &
1141  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
1142  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1143  SCLR_off_SR8 & operation_nd_on_SR1  & operation_nd_on_SR2  & operation_nd_on_SR3  &
1144  operation_nd_on_SR4  & operation_nd_on_SR5  & operation_nd_off_SR6 & operation_nd_off_SR7 &
1145  operation_nd_off_SR8 & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5 &
1146  SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
1147  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
1148  operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
1149  CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1150  constant s13_w : std_logic_vector(N downto 0) := "11000" & R2_OFF & R3_OFF & R4_OFF &
1151  R5_OFF & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_ON  & R13_ON  & R14_ON
1152  & R15_ON  & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_ON  & R29_OFF &
1153  R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1154  R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1155  R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1156  R57_OFF & R58_OFF & R59_OFF & mux_11_1_h & mux_11_2_h & mux_9_1_h & mux_9_2_h & mux_9_3_i &
1157  mux_8_1_g & mux_8_2_a & mux_8_3_a & mux_8_4_a & mux_7_1_f & mux_7_2_f & mux_7_3_f &
1158  mux_7_4_g & mux_7_5_a & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1159  mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1160  mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1161  mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1162  mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1163  mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1164  mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1165  mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1  &
1166  operation_resta_SR2 & operation_suma_SR3  & operation_resta_SR4 & operation_suma_SR5  &
1167  operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
1168  SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1169  SCLR_off_SR8 & operation_nd_off_SR1 & operation_nd_off_SR2 & operation_nd_off_SR3 &
1170  operation_nd_off_SR4 & operation_nd_off_SR5 & operation_nd_off_SR6 & operation_nd_off_SR7 &
1171  operation_nd_off_SR8 & SCLR_on_M1 & SCLR_on_M2 & SCLR_on_M3 & SCLR_on_M4 & SCLR_on_M5 &
1172  SCLR_on_M6 & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 &
1173  operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 &
```

```
1174 operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  &
1175 CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1176 constant s14 : std_logic_vector(N downto 0) := "11001" & R2_OFF & R3_OFF & R4_OFF & R5_OFF
1177 & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF & R14_OFF &
1178 R15_OFF & R16_OFF & R17_OFF & R18_OFF & R19_OFF & R20_OFF & R21_OFF & R28_OFF & R29_OFF &
1179 R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF & R38_OFF &
1180 R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF & R47_OFF &
1181 R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF & R56_OFF &
1182 R57_OFF & R58_OFF & R59_OFF & mux_11_1_i & mux_11_2_i & mux_9_1_i & mux_9_2_i & mux_9_3_h &
1183 mux_8_1_h & mux_8_2_h & mux_8_3_a & mux_8_4_a & mux_7_1_g & mux_7_2_g & mux_7_3_g &
1184 mux_7_4_g & mux_7_5_g & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a & mux_4_1_a &
1185 mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a & mux_3_2_a &
1186 mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a & mux_2_2_a &
1187 mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a & mux_2_9_a &
1188 mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a & mux_2_16_a &
1189 mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a & mux_2_22_a & mux_2_23_a &
1190 mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a & mux_2_28_a & mux_2_29_a & mux_2_30_a &
1191 mux_2_31_a & mux_2_32_a & mux_2_33_a & mux_2_34_a & operation_suma_SR1  &
1192 operation_suma_SR2  & operation_suma_SR3  & operation_suma_SR4  & operation_suma_SR5  &
1193 operation_suma_SR6  & operation_suma_SR7  & operation_suma_SR8  & SCLR_off_SR1 &
1194 SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 & SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 &
1195 SCLR_off_SR8 & operation_nd_on_SR1  & operation_nd_on_SR2  & operation_nd_on_SR3  &
1196 operation_nd_on_SR4  & operation_nd_on_SR5  & operation_nd_on_SR6  & operation_nd_off_SR7
1197 & operation_nd_off_SR8 & SCLR_on_M1  & SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  &
1198 SCLR_on_M5  & SCLR_on_M6  & SCLR_on_M7  & SCLR_on_M8  & operation_nd_off_M1 &
1199 operation_nd_off_M2 & operation_nd_off_M3 & operation_nd_off_M4 & operation_nd_off_M5 &
1200 operation_nd_off_M6 & operation_nd_off_M7 & operation_nd_off_M8 & CE_on_M1  & CE_on_M2  &
1201 CE_on_M3  & CE_on_M4  & CE_on_M5  & CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1202 constant s14_w : std_logic_vector(N downto 0) := "11010" & R2_OFF & R3_OFF & R4_OFF &
1203 R5_OFF & R6_OFF & R7_OFF & R8_OFF & R9_OFF & R10_OFF & R11_OFF & R12_OFF & R13_OFF &
1204 R14_OFF & R15_OFF & R16_ON  & R17_ON  & R18_ON  & R19_ON  & R20_ON  & R21_ON  & R28_ON  &
1205 R29_OFF & R30_OFF & R31_OFF & R32_OFF & R33_OFF & R34_OFF & R35_OFF & R36_OFF & R37_OFF &
1206 R38_OFF & R39_OFF & R40_OFF & R41_OFF & R42_OFF & R43_OFF & R44_OFF & R45_OFF & R46_OFF &
1207 R47_OFF & R48_OFF & R49_OFF & R50_OFF & R51_OFF & R52_OFF & R53_OFF & R54_OFF & R55_OFF &
1208 R56_OFF & R57_OFF & R58_OFF & R59_OFF & mux_11_1_i & mux_11_2_i & mux_9_1_i & mux_9_2_i &
1209 mux_9_3_h & mux_8_1_h & mux_8_2_h & mux_8_3_a & mux_8_4_a & mux_7_1_g & mux_7_2_g &
1210 mux_7_3_g & mux_7_4_g & mux_7_5_g & mux_7_6_a & mux_7_7_a & mux_5_1_a & mux_5_2_a &
1211 mux_4_1_a & mux_4_2_a & mux_4_3_a & mux_4_4_a & mux_4_5_a & mux_4_6_a & mux_3_1_a &
1212 mux_3_2_a & mux_3_3_a & mux_3_4_a & mux_3_5_a & mux_3_6_a & mux_3_7_a & mux_2_1_a &
1213 mux_2_2_a & mux_2_3_a & mux_2_4_a & mux_2_5_a & mux_2_6_a & mux_2_7_a & mux_2_8_a &
1214 mux_2_9_a & mux_2_10_a & mux_2_11_a & mux_2_12_a & mux_2_13_a & mux_2_14_a & mux_2_15_a &
1215 mux_2_16_a & mux_2_17_a & mux_2_18_a & mux_2_19_a & mux_2_20_a & mux_2_21_a &
1216 mux_2_22_a & mux_2_23_a & mux_2_24_a & mux_2_25_a & mux_2_26_a & mux_2_27_a &
1217 mux_2_28_a & mux_2_29_a & mux_2_30_a & mux_2_31_a & mux_2_32_a & mux_2_33_a &
1218 mux_2_34_a & operation_suma_SR1  & operation_suma_SR2  & operation_suma_SR3  &
1219 operation_suma_SR4  & operation_suma_SR5  & operation_suma_SR6  & operation_suma_SR7  &
1220 operation_suma_SR8  & SCLR_off_SR1 & SCLR_off_SR2 & SCLR_off_SR3 & SCLR_off_SR4 &
1221 SCLR_off_SR5 & SCLR_off_SR6 & SCLR_off_SR7 & SCLR_off_SR8 & operation_nd_off_SR1 &
1222 operation_nd_off_SR2 & operation_nd_off_SR3 & operation_nd_off_SR4 & operation_nd_off_SR5 &
1223 operation_nd_off_SR6 & operation_nd_off_SR7 & operation_nd_off_SR8 & SCLR_on_M1  &
1224 SCLR_on_M2  & SCLR_on_M3  & SCLR_on_M4  & SCLR_on_M5  & SCLR_on_M6  & SCLR_on_M7  &
1225 SCLR_on_M8  & operation_nd_off_M1 & operation_nd_off_M2 & operation_nd_off_M3 &
1226 operation_nd_off_M4 & operation_nd_off_M5 & operation_nd_off_M6 & operation_nd_off_M7 &
```

```
1227 operation_nd_off_M8 & CE_on_M1  & CE_on_M2  & CE_on_M3  & CE_on_M4  & CE_on_M5  &
1228 CE_on_M6  & CE_on_M7  & CE_on_M8  & ocupado ;
1229
1230 begin
1231
1232 ----------Registro de Estado----------
1233 process (clk,reset)
1234 begin
1235         if reset='1' then
1236                 estado_presente<=s0;
1237         elsif (clk'event and clk='1') then
1238                 estado_presente<=estado_siguiente;
1239         end if ;
1240 end process ;
1241
1242 ----------Lógica del estado siguiente----------
1243 process (estado_presente,inicio,banderas1,banderas2,flag1)
1244 begin
1245         case estado_presente is
1246                 when s0=>
1247                         if inicio='1' then
1248                                 estado_siguiente<=s1;
1249                         else
1250                                 estado_siguiente<=s0;
1251                         end if ;
1252                 when s1=>
1253                         estado_siguiente<=s2;
1254                 when s2=>
1255                         if flag1='1' then
1256                                 estado_siguiente<=s3;
1257                         else
1258                                 estado_siguiente<=s0;
1259                         end if ;
1260                 when s3=>
1261                         estado_siguiente<=s3_w;
1262                 when s3_w=>
1263                         if ((banderas1(15 downto 8)=x"FF") and
1264                         (banderas2(15 downto 8)=x"FF")) then---condiciòn de los mòdulos de punto
1265                         --flotante sr1 sr2 sr3 sr4 sr5 sr6 sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1266                                 estado_siguiente<=s4;
1267                         else
1268                                 estado_siguiente<=s3_w;
1269                         end if ;
1270                 when s4=>
1271                         estado_siguiente<=s4_w;
1272                 when s4_w=>
1273                         if ((banderas1(15 downto 6)= x"FF" & "11") and
1274                         (banderas2(15 downto 6)= x"FF" & "11")) then---condiciòn de los mòdulos
1275 --de punto flotante sr1 sr2 sr3 sr4 sr5 sr6 sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1276                                 estado_siguiente<=s5;
1277                         else
1278                                 estado_siguiente<=s4_w;
1279                         end if ;
```

```
1280                when s5=>
1281                        estado_siguiente<=s5_w;
1282                when s5_w=>
1283                        if banderas1(14 downto 13)="11" and banderas1(7 downto 0)=x"FF" and
1284                        banderas2(14 downto 13)="11" and banderas2(7 downto 0)=x"FF" then
1285                        ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1286                        --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1287                                estado_siguiente<=s6;
1288                        else
1289                                estado_siguiente<=s5_w;
1290                        end if ;
1291                when s6=>
1292                        estado_siguiente<=s6_w;
1293                when s6_w=>
1294                        if banderas1(13 downto 12)="11" and banderas1(9 downto 8)="11" and
1295                        banderas1(7 downto 0)=x"FF" and banderas2(13 downto 12)="11" and
1296                        banderas2(9 downto 8)="11" and banderas2(7 downto 0)=x"FF" then
1297                        ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1298                        --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1299                                estado_siguiente<=s7;
1300                        else
1301                                estado_siguiente<=s6_w;
1302                        end if ;
1303                when s7=>
1304                        estado_siguiente<=s7_w;
1305                when s7_w=>
1306                        if banderas1(11 downto 9)="111" and banderas1(7 downto 0)=x"FF" and
1307                        banderas2(11 downto 9)="111" and banderas2(7 downto 0)=x"FF" then
1308                        ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1309                        --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1310                                estado_siguiente<=s8;
1311                        else
1312                                estado_siguiente<=s7_w;
1313                        end if ;
1314                when s8=>
1315                        estado_siguiente<=s8_w;
1316                when s8_w=>
1317                        if banderas1(13)='1' and banderas1(9 downto 8)="11" and
1318                        banderas1(7 downto 4)=x"F" and banderas1(2)='1' and banderas1(0)='1' and
1319                        banderas2(13)='1' and banderas2(9 downto 8)="11" and
1320                        banderas2(7 downto 4)=x"F" and banderas2(2)='1' and banderas2(0)='1' then
1321                        ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1322                        --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1323                                estado_siguiente<=s9;
1324                        else
1325                                estado_siguiente<=s8_w;
1326                        end if ;
1327                when s9=>
1328                        estado_siguiente<=s9_w;
1329                when s9_w=>
1330                        if banderas1(13 downto 10)="1111" and banderas2(13 downto 10)="1111" then
1331                        ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1332                        --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
```

```
1333                                    estado_siguiente<=s10;
1334                            else
1335                                    estado_siguiente<=s9_w;
1336                            end if ;
1337                    when s10=>
1338                            estado_siguiente<=s10_w;
1339                    when s10_w=>
1340                            if banderas1(15 downto 8)=x"FF" and banderas2(15 downto 8)=x"FF" then
1341                            ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1342                            --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1343                                    estado_siguiente<=s11;
1344                            else
1345                                    estado_siguiente<=s10_w;
1346                            end if ;
1347                    when s11=>
1348                            estado_siguiente<=s11_w;
1349                    when s11_w=>
1350                            if banderas1(15 downto 8)=x"FF" and banderas2(15 downto 8)=x"FF" then
1351                            ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1352                            --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1353                                    estado_siguiente<=s12;
1354                            else
1355                                    estado_siguiente<=s11_w;
1356                            end if ;
1357                    when s12=>
1358                            estado_siguiente<=s12_w;
1359            when s12_w=>
1360                            if banderas1(15 downto 8)=x"FF" and banderas2(15 downto 8)=x"FF" then
1361                            ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1362                            --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1363                                    estado_siguiente<=s13;
1364                            else
1365                                    estado_siguiente<=s12_w;
1366                            end if ;
1367                    when s13=>
1368                            estado_siguiente<=s13_w;
1369                when s13_w=>
1370                            if banderas1(15 downto 12)=x"F" and banderas1(11)='1' and
1371                            banderas2(15 downto 12)=x"F" and banderas2(11)='1' then
1372                            ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1373                            --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1374                                    estado_siguiente<=s14;
1375                            else
1376                                    estado_siguiente<=s13_w;
1377                            end if ;
1378                    when s14=>
1379                            estado_siguiente<=s14_w;
1380            when s14_w=>
1381                            if banderas1(15 downto 12)=x"F" and banderas1(11 downto 10)="11" and
1382                            banderas2(15 downto 12)=x"F" and banderas2(11 downto 10)="11" then
1383                            ---condiciòn de los mòdulos de punto flotante--- sr1 sr2 sr3 sr4 sr5 sr6
1384                            --sr7 sr8 m1 m2 m3 m4 m5 m6 m7 m8
1385                                    estado_siguiente<=s2;
```

```
1386                            else
1387                                    estado_siguiente<=s14_w;
1388                            end if ;
1389                    when others =>
1390                            estado_siguiente<=s0;
1391            end case ;
1392    end process ;
1393
1394    process(clk)
1395    begin
1396            if(clk'event and clk='1')then
1397                    if(estado_presente = s0)then
1398                            L <= "00000";
1399                    elsif( flag3 = '1' )then
1400                            L <= L + '1' ;
1401                    else
1402                            L <= L;
1403                    end if;
1404            else
1405                    L <= L ;
1406            end if;
1407    end process;
1408
1409    process(clk)
1410    begin
1411            if(clk'event and clk='1')then
1412                    if(estado_presente = s1)then
1413                            m1 <= m;
1414                    else
1415                            m1 <= m1;
1416                    end if;
1417            end if;
1418    end process;
1419
1420
1421    flag1 <= '1' when L < m1 else
1422                             '0';
1423
1424    flag2 <= '1' when estado_presente(N downto N-4) = "00010" else
1425                             '0';
1426
1427    flag3 <= flag1 and flag2;
1428
1429    senales_de_control <= estado_presente(N downto 49) & "00000" & estado_presente(48) & "00000" &
1430    estado_presente(47) & "00000" & estado_presente(46) & "00000" & estado_presente(45) & "00000" &
1431    estado_presente(44) & "00000" & estado_presente(43) & "00000" & estado_presente(42) & "00000" &
1432    estado_presente(41) & estado_presente(40 downto 0) & CE_on_SR1 & CE_on_SR2 & CE_on_SR3 &
1433    CE_on_SR4 & CE_on_SR5 & CE_on_SR6 & CE_on_SR7 & CE_on_SR8 ;
1434
1435    estado_sig<=estado_siguiente(N downto N-4);
1436
1437    end Behavioral;
```

G.19 Sección memoria-prueba.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    09:52:21 09/20/2010
7  -- Design Name:
8  -- Module Name:    memoria_prueba - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity memoria_prueba is
32          generic(INIT_00,INIT_01,INIT_02,INIT_03,INIT_04,INIT_05,INIT_06,INIT_07,INIT_08,
33          INIT_09, INIT_0A, INIT_0B, INIT_0C, INIT_0D, INIT_0E,
34          INIT_0F: std_logic_vector(31 downto 0);
35      ancho_dir : integer;
36          ancho_dat : integer);
37     Port (
38 clockA,clockB : in  STD_LOGIC;
39 ram_enableA,ram_enableB : in  STD_LOGIC;
40 write_enableA,write_enableB : in  STD_LOGIC;
41 input_dataA,input_dataB : in  std_logic_vector(31 downto 0);
42 ram_outputA,ram_outputB : out std_logic_vector(31 downto 0);
43 addressA,addressB : in std_logic_vector(3 downto 0));
44 attribute RAM_STYLE : string;
45 attribute RAM_STYLE of memoria_prueba: entity is "BLOCK";
46 end memoria_prueba;
47
48 architecture Behavioral of memoria_prueba is
49
50 constant ADDR_WIDTH : integer := ancho_dir;--4
51 constant DATA_WIDTH : integer := ancho_dat; --32
```

```
52
53 type ram_type is array (0 to 2**ADDR_WIDTH-1) of std_logic_vector (DATA_WIDTH-1 downto 0);
54 shared variable ram_name: ram_type:=(INIT_00, INIT_01, INIT_02, INIT_03, INIT_04, INIT_05,
55 INIT_06, INIT_07, INIT_08, INIT_09, INIT_0A, INIT_0B, INIT_0C, INIT_0D, INIT_0E, INIT_0F);
56
57
58 begin
59
60 process (clockA)
61 begin
62    if (clockA'event and clockA = '1') then
63       if (ram_enableA = '1') then
64          if (write_enableA = '1') then
65             ram_name(conv_integer(addressA)) := input_dataA;
66          end if;
67          ram_outputA <= ram_name(conv_integer(addressA));
68       end if;
69    end if;
70 end process;
71
72 process (clockB)
73 begin
74    if (clockB'event and clockB = '1') then
75       if (ram_enableB = '1') then
76          if (write_enableB = '1') then
77             ram_name(conv_integer(addressB)) := input_dataB;
78          end if;
79          ram_outputB <= ram_name(conv_integer(addressB));
80       end if;
81    end if;
82 end process;
83
84 end Behavioral;
```

G.20 Registro.vhd

```
1
2 ----------------------------------------------------------------------------------
3 -- Company:
4 -- Engineer:
5 --
6 -- Create Date:    16:16:17 08/11/2010
7 -- Design Name:
8 -- Module Name:    registro - Behavioral
9 -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
```

```
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity registro is
32
33     generic(
34
35     valor_inicial : std_logic_vector(31 downto 0)
36
37     );
38     Port (
39 clk : in  STD_LOGIC;
40 reset : in  STD_LOGIC;
41 enable : in  STD_LOGIC;
42 entrada : in  STD_LOGIC_VECTOR (31 downto 0);
43 salida : out  STD_LOGIC_VECTOR (31 downto 0));
44 end registro;
45
46 architecture Behavioral of registro is
47 signal salida1 : std_logic_vector(31 downto 0):=valor_inicial;
48 begin
49
50 process (clk, reset)
51 begin
52    if reset='1' then
53       salida1 <= (others => '0');
54         elsif (clk'event and clk='1') then
55                 if enable = '1' then
56          salida1 <= entrada;
57       end if;
58    end if;
59 end process;
60
61
62 salida<=salida1;
63
64 end Behavioral;
```

G.21 mux2.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    11:44:04 08/11/2010
7  -- Design Name:
8  -- Module Name:    mux2 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux2 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            control : in  STD_LOGIC;
35            salida : out  STD_LOGIC_VECTOR (31 downto 0));
36 end mux2;
37
38 architecture Behavioral of mux2 is
39
40 begin
41
42 with control select
43       salida <= a when '0',
44                 b when others;
45
46
47 end Behavioral;
```

G.22 mux3.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    11:42:35 08/11/2010
7  -- Design Name:
8  -- Module Name:    mux3 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux3 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            control : in  STD_LOGIC_VECTOR (1 downto 0);
36            salida : out  STD_LOGIC_VECTOR (31 downto 0));
37 end mux3;
38
39 architecture Behavioral of mux3 is
40
41 begin
42
43 with control select
44       salida <= a when "00",
45                 b when "01",
46                 c when others;
47
48
49 end Behavioral;
```

G.23 mux4.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    11:40:56 08/11/2010
7  -- Design Name:
8  -- Module Name:    mux4 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux4 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            d : in  STD_LOGIC_VECTOR (31 downto 0);
36            control : in  STD_LOGIC_VECTOR (1 downto 0);
37            salida : out  STD_LOGIC_VECTOR (31 downto 0));
38 end mux4;
39
40 architecture Behavioral of mux4 is
41
42 begin
43
44 with control select
45       salida <= a when "00",
46                 b when "01",
47                 c when "10",
48                 d when others;
49
50
51 end Behavioral;
```

G.24 mux5.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    11:39:05 08/11/2010
7  -- Design Name:
8  -- Module Name:    mux5 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux5 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            d : in  STD_LOGIC_VECTOR (31 downto 0);
36            e : in  STD_LOGIC_VECTOR (31 downto 0);
37            control : in  STD_LOGIC_VECTOR (2 downto 0);
38            salida : out  STD_LOGIC_VECTOR (31 downto 0));
39 end mux5;
40
41 architecture Behavioral of mux5 is
42
43 begin
44
45
46 with control select
47       salida <= a when "000",
```

```
48                  b when "001",
49                  c when "010",
50                  d when "011",
51                  e when others;
52
53  end Behavioral;
```

G.25 mux7.vhd

```
1
2   ----------------------------------------------------------------------------------
3   -- Company:
4   -- Engineer:
5   --
6   -- Create Date:    11:37:06 08/11/2010
7   -- Design Name:
8   -- Module Name:    mux7 - Behavioral
9   -- Project Name:
10  -- Target Devices:
11  -- Tool versions:
12  -- Description:
13  --
14  -- Dependencies:
15  --
16  -- Revision:
17  -- Revision 0.01 - File Created
18  -- Additional Comments:
19  --
20  ----------------------------------------------------------------------------------
21  library IEEE;
22  use IEEE.STD_LOGIC_1164.ALL;
23  use IEEE.STD_LOGIC_ARITH.ALL;
24  use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26  ---- Uncomment the following library declaration if instantiating
27  ---- any Xilinx primitives in this code.
28  --library UNISIM;
29  --use UNISIM.VComponents.all;
30
31  entity mux7 is
32      Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33             b : in  STD_LOGIC_VECTOR (31 downto 0);
34             c : in  STD_LOGIC_VECTOR (31 downto 0);
35             d : in  STD_LOGIC_VECTOR (31 downto 0);
36             e : in  STD_LOGIC_VECTOR (31 downto 0);
37             f : in  STD_LOGIC_VECTOR (31 downto 0);
38             g : in  STD_LOGIC_VECTOR (31 downto 0);
39             control : in  STD_LOGIC_VECTOR (2 downto 0);
40             salida : out  STD_LOGIC_VECTOR (31 downto 0));
41  end mux7;
42
```

```
43  architecture Behavioral of mux7 is
44
45  begin
46
47
48  with control select
49        salida <= a when "000",
50                  b when "001",
51                  c when "010",
52                  d when "011",
53                  e when "100",
54                  f when "101",
55                  g when others;
56
57  end Behavioral;
```

G.26 mux8.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:     11:34:09 08/11/2010
7  -- Design Name:
8  -- Module Name:     mux8 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux8 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
```

```
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            d : in  STD_LOGIC_VECTOR (31 downto 0);
36            e : in  STD_LOGIC_VECTOR (31 downto 0);
37            f : in  STD_LOGIC_VECTOR (31 downto 0);
38            g : in  STD_LOGIC_VECTOR (31 downto 0);
39            h : in  STD_LOGIC_VECTOR (31 downto 0);
40            control : in  STD_LOGIC_VECTOR (2 downto 0);
41            salida : out  STD_LOGIC_VECTOR (31 downto 0));
42 end mux8;
43
44 architecture Behavioral of mux8 is
45
46 begin
47
48 with control select
49       salida <= a when "000",
50                 b when "001",
51                 c when "010",
52                 d when "011",
53                 e when "100",
54                 f when "101",
55                 g when "110",
56                 h when others;
57
58
59 end Behavioral;
```

G.27 mux9.vhd

```
1
2  ----------------------------------------------------------------------------------
3  -- Company:
4  -- Engineer:
5  --
6  -- Create Date:    11:27:25 08/11/2010
7  -- Design Name:
8  -- Module Name:    mux9 - Behavioral
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
```

```
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux9 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            d : in  STD_LOGIC_VECTOR (31 downto 0);
36            e : in  STD_LOGIC_VECTOR (31 downto 0);
37            f : in  STD_LOGIC_VECTOR (31 downto 0);
38            g : in  STD_LOGIC_VECTOR (31 downto 0);
39            h : in  STD_LOGIC_VECTOR (31 downto 0);
40            i : in  STD_LOGIC_VECTOR (31 downto 0);
41
42                          salida : out  STD_LOGIC_VECTOR (31 downto 0);
43            control : in  STD_LOGIC_VECTOR (3 downto 0));
44 end mux9;
45
46 architecture Behavioral of mux9 is
47
48 begin
49
50 with control select
51       salida <= a when "0000",
52                 b when "0001",
53                 c when "0010",
54                 d when "0011",
55                 e when "0100",
56                 f when "0101",
57                 g when "0110",
58                 h when "0111",
59                 i when others;
60
61
62 end Behavioral;
```

G.28 mux11.vhd

```
1
2 ----------------------------------------------------------------------------------
3 -- Company:
4 -- Engineer:
5 --
6 -- Create Date:    08:45:05 08/25/2010
7 -- Design Name:
8 -- Module Name:    mux11 - Behavioral
```

```
9  -- Project Name:
10 -- Target Devices:
11 -- Tool versions:
12 -- Description:
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31 entity mux11 is
32     Port ( a : in  STD_LOGIC_VECTOR (31 downto 0);
33            b : in  STD_LOGIC_VECTOR (31 downto 0);
34            c : in  STD_LOGIC_VECTOR (31 downto 0);
35            d : in  STD_LOGIC_VECTOR (31 downto 0);
36            e : in  STD_LOGIC_VECTOR (31 downto 0);
37            f : in  STD_LOGIC_VECTOR (31 downto 0);
38            g : in  STD_LOGIC_VECTOR (31 downto 0);
39            h : in  STD_LOGIC_VECTOR (31 downto 0);
40            i : in  STD_LOGIC_VECTOR (31 downto 0);
41            j : in  STD_LOGIC_VECTOR (31 downto 0);
42            k : in  STD_LOGIC_VECTOR (31 downto 0);
43            salida : out  STD_LOGIC_VECTOR (31 downto 0);
44            control : in  STD_LOGIC_VECTOR (3 downto 0));
45 end mux11;
46
47 architecture Behavioral of mux11 is
48
49 begin
50
51
52 with control select
53       salida <= a when "0000",
54                 b when "0001",
55                 c when "0010",
56                 d when "0011",
57                 e when "0100",
58                 f when "0101",
59                 g when "0110",
60                 h when "0111",
61                 i when "1000",
```

```
62                  j when "1001",
63                  k when others;
64
65  end Behavioral;
```

G.29 async_trap_and_reset3.v

```
1   module async_trap_and_reset3 (
2       async_sig, outclk, auto_reset, reset,
3       out_sync_sig     );
4       parameter retardo = 3'd4;             // number of bits in register
5
6   /*  This module traps an asyncronous signal async_sig and syncronizes it via 2
7       flip-flops to outclk. The resulting signal is named out_sync_sig.
8       auto_reset tells the module whether to do an auto-reset of out_sync_sig.
9       reset is an asynchronous reset signal. This reset signal is active LOW. */
10
11      input async_sig, outclk, auto_reset, reset;
12      output out_sync_sig;
13
14      reg sync1= 1'b0, sync2= 1'b0, async_trap= 1'b0;
15      reg actual_auto_reset_signal = 1'b1;
16      wire actual_async_sig_reset;
17
18      reg [retardo-1:0] registro;
19      wire auto_reset_signal =  auto_reset && registro[retardo-1];
20
21    always @ (posedge outclk or negedge reset)
22        if (~reset)
23            actual_auto_reset_signal <= 1'b0;
24        else
25            actual_auto_reset_signal <= auto_reset_signal;
26
27    assign actual_async_sig_reset = actual_auto_reset_signal || (!reset);
28
29    always @ (posedge async_sig or posedge actual_async_sig_reset)
30        if (actual_async_sig_reset)
31            async_trap <= 1'b0;
32        else
33            async_trap <= 1'b1;
34
35    always @ (negedge outclk or posedge actual_async_sig_reset)
36        if (actual_async_sig_reset)
37            sync1 <= 1'b0;
38        else
39            sync1 <= async_trap;
40
41    always @ (negedge outclk or negedge reset)
42        if (~reset)
43            sync2 <= 1'b0;
44        else
```

```
45            sync2 <= sync1;
46
47    assign out_sync_sig = sync2;
48  // ------------
49    always @ (posedge outclk or negedge reset)
50        if (~reset)
51            registro <= 0;
52        else
53            registro <= {registro[retardo-2:0], sync2};
54
55  endmodule
```

Apéndice H

Archivos de la interconexión

H.1 Archivos modificados

```
1  ##############################################################################
2  ## Filename:          C:/EDK_maestria_05/ml507_ppc440_emb_ref_mod/pcores/procesador_factor_7_v1_00_a
3  /data/procesador_factor_7_v2_1_0.pao
4  ## Description:       Peripheral Analysis Order
5  ## Date:              Mon Nov 29 06:57:31 2010 (by Create and Import Peripheral Wizard)
6  ##############################################################################
7
8  lib proc_common_v2_00_a proc_common_pkg vhdl
9  lib proc_common_v2_00_a ipif_pkg vhdl
10 lib proc_common_v2_00_a or_muxcy vhdl
11 lib proc_common_v2_00_a or_gate128 vhdl
12 lib proc_common_v2_00_a family_support vhdl
13 lib proc_common_v2_00_a pselect_f vhdl
14 lib proc_common_v2_00_a counter_f vhdl
15 lib plbv46_slave_single_v1_00_a plb_address_decoder vhdl
16 lib plbv46_slave_single_v1_00_a plb_slave_attachment vhdl
17 lib plbv46_slave_single_v1_00_a plbv46_slave_single vhdl
18 lib procesador_factor_7_v1_00_a user_logic vhdl
19 lib procesador_factor_7_v1_00_a procesador_factor_7 vhdl
20 lib procesador_factor_7_v1_00_a FSM_7 vhdl
21 lib procesador_factor_7_v1_00_a memoria_prueba vhdl
22 lib procesador_factor_7_v1_00_a multiplicacion vhdl
23 lib procesador_factor_7_v1_00_a mux2 vhdl
24 lib procesador_factor_7_v1_00_a mux3 vhdl
25 lib procesador_factor_7_v1_00_a mux4 vhdl
26 lib procesador_factor_7_v1_00_a mux5 vhdl
27 lib procesador_factor_7_v1_00_a mux7 vhdl
28 lib procesador_factor_7_v1_00_a mux8 vhdl
29 lib procesador_factor_7_v1_00_a mux9 vhdl
30 lib procesador_factor_7_v1_00_a mux11 vhdl
31 lib procesador_factor_7_v1_00_a principal vhdl
32 lib procesador_factor_7_v1_00_a registro vhdl
33 lib procesador_factor_7_v1_00_a Seccion_M1 vhdl
34 lib procesador_factor_7_v1_00_a Seccion_M2 vhdl
```

```
35 lib procesador_factor_7_v1_00_a Seccion_M3 vhdl
36 lib procesador_factor_7_v1_00_a Seccion_M4 vhdl
37 lib procesador_factor_7_v1_00_a Seccion_M5 vhdl
38 lib procesador_factor_7_v1_00_a Seccion_M6 vhdl
39 lib procesador_factor_7_v1_00_a Seccion_M7 vhdl
40 lib procesador_factor_7_v1_00_a Seccion_M8 vhdl
41 lib procesador_factor_7_v1_00_a Seccion_SR1 vhdl
42 lib procesador_factor_7_v1_00_a Seccion_SR2 vhdl
43 lib procesador_factor_7_v1_00_a Seccion_SR3 vhdl
44 lib procesador_factor_7_v1_00_a Seccion_SR4 vhdl
45 lib procesador_factor_7_v1_00_a Seccion_SR5 vhdl
46 lib procesador_factor_7_v1_00_a Seccion_SR6 vhdl
47 lib procesador_factor_7_v1_00_a Seccion_SR7 vhdl
48 lib procesador_factor_7_v1_00_a Seccion_SR8 vhdl
49 lib procesador_factor_7_v1_00_a suma_resta vhdl
50 LIB procesador_factor_7_v1_00_a async_trap_and_reset3 verilog
```

Script H.1: Versión final del archivo procesador_factor_7_v2_1_0.pao

```
1 ## Ports
2 PORT clk ="", DIR= I
3 PORT SPLB_Clk = "", DIR = I, SIGIS = CLK, BUS = SPLB
4 PORT SPLB_Rst = SPLB_Rst, DIR = I, SIGIS = RST, BUS = SPLB
5 PORT PLB_ABus = PLB_ABus, DIR = I, VEC = [0:31], BUS = SPLB
6 PORT PLB_UABus = PLB_UABus, DIR = I, VEC = [0:31], BUS = SPLB
7 PORT PLB_PAValid = PLB_PAValid, DIR = I, BUS = SPLB
```

Script H.2: Versión final del archivo procesador_factor_7_v2_1_0.mpd

```
1  port
2    (
3      -- ADD USER PORTS BELOW THIS LINE ------------------
4      --USER ports added here
5      clk : in  STD_LOGIC;
6      -- ADD USER PORTS ABOVE THIS LINE ------------------
7
8  port map
9      (
10       -- MAP USER PORTS BELOW THIS LINE ------------------
11       --USER ports mapped here
12       clk => clk,
13       -- MAP USER PORTS ABOVE THIS LINE ------------------
```

Script H.3: Versión final del archivo procesador_factor_7.vhd

```
1  port
2    (
3      -- ADD USER PORTS BELOW THIS LINE ------------------
4      --USER ports added here
5      clk : in  STD_LOGIC;--clk  periodo > 14 ns
6      -- ADD USER PORTS ABOVE THIS LINE ------------------
```

```
7
8  architecture IMP of user_logic is
9
10   --USER signal declarations added here, as needed for user logic
11
12   COMPONENT principal
13         PORT(
14                  mu : IN std_logic_vector(3 downto 0);
15                  inicio : IN std_logic;
16                  clk : IN std_logic;
17                  reset : IN std_logic;
18                  m : IN std_logic_vector(19 downto 0);
19                  clk_driver : IN std_logic;
20                  ram_enable_driver : IN std_logic;
21                  write_enable_driver : IN std_logic;
22                  input_data_driver : IN std_logic_vector(31 downto 0);
23                  address_driver : IN std_logic_vector(3 downto 0);
24                  bloque_ram_selection : IN std_logic_vector(3 downto 0);
25                  j0 : IN std_logic_vector(31 downto 0);
26                  j1 : IN std_logic_vector(31 downto 0);
27                  j2 : IN std_logic_vector(31 downto 0);
28                  j3 : IN std_logic_vector(31 downto 0);
29                  j4 : IN std_logic_vector(31 downto 0);
30                  j5 : IN std_logic_vector(31 downto 0);
31                  j6 : IN std_logic_vector(31 downto 0);
32                  underflow_sr1 : OUT std_logic;
33                  underflow_sr2 : OUT std_logic;
34                  underflow_sr3 : OUT std_logic;
35                  underflow_sr4 : OUT std_logic;
36                  underflow_sr5 : OUT std_logic;
37                  underflow_sr6 : OUT std_logic;
38                  underflow_sr7 : OUT std_logic;
39                  underflow_sr8 : OUT std_logic;
40                  underflow_M1 : OUT std_logic;
41                  underflow_M2 : OUT std_logic;
42                  underflow_M3 : OUT std_logic;
43                  underflow_M4 : OUT std_logic;
44                  underflow_M5 : OUT std_logic;
45                  underflow_M6 : OUT std_logic;
46                  underflow_M7 : OUT std_logic;
47                  underflow_M8 : OUT std_logic;
48                  overflow_sr1 : OUT std_logic;
49                  overflow_sr2 : OUT std_logic;
50                  overflow_sr3 : OUT std_logic;
51                  overflow_sr4 : OUT std_logic;
52                  overflow_sr5 : OUT std_logic;
53                  overflow_sr6 : OUT std_logic;
54                  overflow_sr7 : OUT std_logic;
55                  overflow_sr8 : OUT std_logic;
56                  overflow_M1 : OUT std_logic;
57                  overflow_M2 : OUT std_logic;
58                  overflow_M3 : OUT std_logic;
59                  overflow_M4 : OUT std_logic;
```

```
60                  overflow_M5 : OUT std_logic;
61                  overflow_M6 : OUT std_logic;
62                  overflow_M7 : OUT std_logic;
63                  overflow_M8 : OUT std_logic;
64                  invalid_op_sr1 : OUT std_logic;
65                  invalid_op_sr2 : OUT std_logic;
66                  invalid_op_sr3 : OUT std_logic;
67                  invalid_op_sr4 : OUT std_logic;
68                  invalid_op_sr5 : OUT std_logic;
69                  invalid_op_sr6 : OUT std_logic;
70                  invalid_op_sr7 : OUT std_logic;
71                  invalid_op_sr8 : OUT std_logic;
72                  invalid_op_M1 : OUT std_logic;
73                  invalid_op_M2 : OUT std_logic;
74                  invalid_op_M3 : OUT std_logic;
75                  invalid_op_M4 : OUT std_logic;
76                  invalid_op_M5 : OUT std_logic;
77                  invalid_op_M6 : OUT std_logic;
78                  invalid_op_M7 : OUT std_logic;
79                  invalid_op_M8 : OUT std_logic;
80                  ocupado : OUT std_logic;
81                  ram_output_driver : OUT std_logic_vector(31 downto 0);
82                  banderas_11 : OUT std_logic_vector(15 downto 0);
83                  banderas_22 : OUT std_logic_vector(15 downto 0);
84                  estado_sig_out : out std_logic_vector(4 downto 0);--salida de prueba
85                  j0_o : OUT std_logic_vector(31 downto 0);
86                  j1_o : OUT std_logic_vector(31 downto 0);
87                  j2_o : OUT std_logic_vector(31 downto 0);
88                  j3_o : OUT std_logic_vector(31 downto 0);
89                  j4_o : OUT std_logic_vector(31 downto 0);
90                  j5_o : OUT std_logic_vector(31 downto 0);
91                  j6_o : OUT std_logic_vector(31 downto 0)
92                  );
93          END COMPONENT;
94
95      signal ocupado : std_logic;
96                  signal ram_output_driver : std_logic_vector(31 downto 0);
97                  signal banderas_11 : std_logic_vector(15 downto 0);
98                  signal banderas_22 : std_logic_vector(15 downto 0);
99                  signal estado_sig_out : std_logic_vector(4 downto 0);
100                 signal j0_o : std_logic_vector(31 downto 0);
101                 signal j1_o : std_logic_vector(31 downto 0);
102                 signal j2_o : std_logic_vector(31 downto 0);
103                 signal j3_o : std_logic_vector(31 downto 0);
104                 signal j4_o : std_logic_vector(31 downto 0);
105                 signal j5_o : std_logic_vector(31 downto 0);
106                 signal j6_o : std_logic_vector(31 downto 0);
107
108     ---------------------------------------
109     -- Signals for user logic slave model s/w accessible register example
110     ---------------------------------------
111
112     --USER logic implementation added here
```

```
113
114   Inst_principal: principal PORT MAP(
115                  mu => slv_reg0(18 to 21),
116                  inicio => slv_reg0(17),
117                  clk => clk,
118                  reset => slv_reg0(16),
119                  underflow_sr1 => open,
120                  underflow_sr2 => open,
121                  underflow_sr3 => open,
122                  underflow_sr4 => open,
123                  underflow_sr5 => open,
124                  underflow_sr6 => open,
125                  underflow_sr7 => open,
126                  underflow_sr8 => open,
127                  underflow_M1 => open,
128                  underflow_M2 => open,
129                  underflow_M3 => open,
130                  underflow_M4 => open,
131                  underflow_M5 => open,
132                  underflow_M6 => open,
133                  underflow_M7 => open,
134                  underflow_M8 => open,
135                  overflow_sr1 => open,
136                  overflow_sr2 => open,
137                  overflow_sr3 => open,
138                  overflow_sr4 => open,
139                  overflow_sr5 => open,
140                  overflow_sr6 => open,
141                  overflow_sr7 => open,
142                  overflow_sr8 => open,
143                  overflow_M1 => open,
144                  overflow_M2 => open,
145                  overflow_M3 => open,
146                  overflow_M4 => open,
147                  overflow_M5 => open,
148                  overflow_M6 => open,
149                  overflow_M7 => open,
150                  overflow_M8 => open,
151                  invalid_op_sr1 => open,
152                  invalid_op_sr2 => open,
153                  invalid_op_sr3 => open,
154                  invalid_op_sr4 => open,
155                  invalid_op_sr5 => open,
156                  invalid_op_sr6 => open,
157                  invalid_op_sr7 => open,
158                  invalid_op_sr8 => open,
159                  invalid_op_M1 => open,
160                  invalid_op_M2 => open,
161                  invalid_op_M3 => open,
162                  invalid_op_M4 => open,
163                  invalid_op_M5 => open,
164                  invalid_op_M6 => open,
165                  invalid_op_M7 => open,
```

```
166                 invalid_op_M8 => open,
167                 m => slv_reg18(0 to 19),
168                 ocupado => ocupado,
169                 clk_driver => Bus2IP_Clk,
170                 ram_enable_driver => slv_reg0(27),
171                 write_enable_driver => slv_reg0(26),
172                 input_data_driver => slv_reg2,
173                 ram_output_driver => ram_output_driver,
174                 address_driver => slv_reg0(22 to 25),
175                 bloque_ram_selection => slv_reg0(28 to 31),
176                 banderas_11 => banderas_11,
177                 banderas_22 => banderas_22,
178                 estado_sig_out => estado_sig_out,
179                 j0 => slv_reg3,
180                 j1 => slv_reg4,
181                 j2 => slv_reg5,
182                 j3 => slv_reg6,
183                 j4 => slv_reg7,
184                 j5 => slv_reg8,
185                 j6 => slv_reg9,
186                 j0_o => j0_o,
187                 j1_o => j1_o,
188                 j2_o => j2_o,
189                 j3_o => j3_o,
190                 j4_o => j4_o,
191                 j5_o => j5_o,
192                 j6_o => j6_o
193         );
194
195   ------------------------------------------
196   -- Example code to read/write user logic slave model s/w accessible registers
197
198   case slv_reg_read_sel is
199       when "1000000000000000000" => slv_ip2bus_data <= slv_reg0(0 to 9) & estado_sig_out & ocupado
200       & slv_reg0(16 to 31);
201       when "0100000000000000000" => slv_ip2bus_data <= ram_output_driver;
202       when "0010000000000000000" => slv_ip2bus_data <= slv_reg2;
203       when "0001000000000000000" => slv_ip2bus_data <= slv_reg3;
204       when "0000100000000000000" => slv_ip2bus_data <= slv_reg4;
205       when "0000010000000000000" => slv_ip2bus_data <= slv_reg5;
206       when "0000001000000000000" => slv_ip2bus_data <= slv_reg6;
207       when "0000000100000000000" => slv_ip2bus_data <= slv_reg7;
208       when "0000000010000000000" => slv_ip2bus_data <= slv_reg8;
209       when "0000000001000000000" => slv_ip2bus_data <= slv_reg9;
210       when "0000000000100000000" => slv_ip2bus_data <= j0_o;
211       when "0000000000010000000" => slv_ip2bus_data <= j1_o;
212       when "0000000000001000000" => slv_ip2bus_data <= j2_o;
213       when "0000000000000100000" => slv_ip2bus_data <= j3_o;
214       when "0000000000000010000" => slv_ip2bus_data <= j4_o;
215       when "0000000000000001000" => slv_ip2bus_data <= j5_o;
216       when "0000000000000000100" => slv_ip2bus_data <= j6_o;
217       when "0000000000000000010" => slv_ip2bus_data <= banderas_22 & banderas_11;
218       when "0000000000000000001" => slv_ip2bus_data <= slv_reg18;
```

```
219         when others => slv_ip2bus_data <= (others => '0');
220       end case;
```

Script H.4: Versión final del archivo user_logic.vhd

Plantilla del driver

I.1 Plantilla del Driver

```
1  /*****************************************************************************
2   * Driver para una pantalla LCD. Este archivo puede servir como plantilla    *
3   * para otros dispositivos a los que se les quiera implementar un driver.    *
4   *****************************************************************************/
5
6  /***********************
7   * Librerías generales *
8   ***********************/
9
10 /* Soporte para establecer las funciones de inicialización y finalización    *
11  * mediante los macros module_init module_exit                               */
12 #include<linux/init.h>
13
14 /* Obligatorio para cualquier módulo                                         */
15 #include<linux/module.h>
16
17 /* Definiciones generales y la mayoría de los macros que interactuan con el  *
18  * kernel                                                                    */
19 #include<linux/sched.h>
20
21 /* Funciones relacionadas con la escritura drivers, acá estan las funciones  *
22  * para registrar los numeros de dispositivos y los cdev                     */
23 #include<linux/fs.h>
24
25 /* Permite instanciar y manipular los cdev                                   */
26 #include<linux/cdev.h>
27
28 /* ?? */
29 #include<linux/slab.h>
30
31 /* Soporta copy_to_user y copy_from_user                                     */
32 #include<asm/uaccess.h>
33
34 /* Soporte para manejar los puertos                                          */
```

```
35 #include<linux/ioport.h>
36
37 /* Soporte para las funciones inx outx                                    */
38 #include<asm/io.h>
39
40 /* Soporte para las barreras                                              */
41 #include<asm/system.h>
42
43 /* Soporte para msleep */
44 #include<linux/delay.h>
```

Script I.1: plantilla del driver sección de librerías

```
1  /*****************************
2   * Librerías personalizadas *
3   *****************************/
4
5  /* Definiciones para el LCD                                               */
6  #include"LCD.h"
7
8  /* Función importada del EDK                                              */
9  #include"lcd/lcd_lib.h"
10 #include"lcd/lcd_lib.c"
11
12 /*****************************
13  * Librerías personalizadas *
14  *****************************/
15
16 /* Cantidad de caracteres que se copian cada vez que se llama las funciones   *
17  * LCD_write o LCD_read                                                   */
18 #define tamano_del_paquete 128
19
20 /* Habilita la impresión de mensajes de depuración mediante printk        */
21 //#define depura
22 #define PRINTK_LEVEL KERN_INFO
23
24 /* Datos del periférico LCD                                               */
25 #define LCD_BASEADDRESS 0x83C18000
26 #define LCD_NAME "Puerto_LCD"
27
28 /* Puntero devuelto por ioremap                                           */
29 static void * LCD_iomem_pointer;
30
31 /* Licencia bajo la cual se desarrolló el driver                          */
32 MODULE_LICENSE("Dual BSD/GPL");
```

Script I.2: plantilla del driver sección de librerías personalizadas

```
1 /*******************************************
2  * Parámetros al momento de cargar el módulo *
3  *******************************************/
4 static char *whom    = "Mundo";
```

```
5  static int  howmany = 1;
6  module_param(whom,     charp, S_IRUGO);
7  module_param(howmany, int,    S_IRUGO);
8
9  /************************
10  * Parámetros del driver *
11  ************************/
12
13 /* Nombre que aparecerá en el /proc/devices                              */
14 static char *nombre = "LCD";
15 /* El primer minor number es cero                                        */
16 static unsigned int firstminor = 0;
17 /* Indica cuantos devices LCD existen, para reservar igual cantidad de minor  *
18  * numbers.                                                              */
19 static unsigned int count = 1;
20
21 /* Dato de tipo dev_t que se usa en las funciones LCD_init y LCD_exit y que   *
22  * agrupa los major y minor numbers asignados al dispositivo LCD         */
23 static dev_t mydev;
24 /* Indica si el driver ha sido inicializado, en otras palabras, si alguna vez *
25  * se ha ejecutado la función open.                                      */
26 static char bandera;
27
28
29 /* Función change_endianness: Tiene como objetivo cambiar el endianness a un  *
30  * dato de 32 bits tipo u32.                                             */
31 u32 change_endianness(u32 dato)
32 {
33     u32 valret=0;
34     *(((u8 *) &valret)+3) = *(((u8 *)&dato)+0);
35     *(((u8 *) &valret)+2) = *(((u8 *)&dato)+1);
36     *(((u8 *) &valret)+1) = *(((u8 *)&dato)+2);
37     *(((u8 *) &valret)+0) = *(((u8 *)&dato)+3);
38     return valret;
39 }
```

Script I.3: plantilla del driver sección de parámetros del driver

```
1
2  /* Función LCD_llseek:                                                   */
3  loff_t LCD_llseek (struct file *filp, loff_t off, int nose)
4  {
5  #ifdef depura
6          printk(PRINTK_LEVEL "LCD_depura: LCD_llseek: Estoy ejecutando la función LCD_llseek\n");
7  #endif
8  return 0;
9  }
```

Script I.4: plantilla del driver sección función llseek

```
1
2  /* Función LCD_read: El kernel solicita leer "tamano" bytes al driver. El    *
```

```
3   * driver responde con la cantidad de bytes leidos o cero si se llegó al      *
4   * final del archivo.                                                          */
5   ssize_t LCD_read (struct file *puntero_file, char __user *puntero_usuario,
6   size_t tamano, loff_t * puntero_offset)
7   {
8       /* Variable que almacena el valor a devolver (Bytes leidos)              */
9       ssize_t retval;
10      /* Conserva un puntero a la estructura de tipo LCD_dev                   */
11      struct LCD_dev *el_LCD = puntero_file->private_data;
12      /* Tamaño del buffer calcluado según memoria de caracteres               */
13      int tamano_mem_lcd = el_LCD->filas*el_LCD->columnas;
14
15  #ifdef depura
16      printk(PRINTK_LEVEL "LCD_depura: LCD_read: Estoy ejecutando la función LCD_read\n");
17      printk(PRINTK_LEVEL "LCD_depura: LCD_read: Tamano = %d\n",tamano);
18      printk(PRINTK_LEVEL "LCD_depura: LCD_read: El contenido de puntero3 es:
19      %lu\n",(long unsigned int)*puntero_offset);
20  #endif
21      /* Dejamos en tamano el menor entre tamano y tamano_del_paquete porque el *
22       * driver no piensa leer mas que eso (él es perezoso)                    */
23      if(tamano>tamano_del_paquete)
24          tamano = tamano_del_paquete;
25
26      /* Verificamos si se puede sobrepasar el tamaño máximo del buffer y      *
27       * limitamos la cantidad de datos leidos en caso de sobrepasarnos        */
28      if((*puntero_offset + tamano)>=tamano_mem_lcd) { // Si sobrepasa límite...
29          retval = tamano_mem_lcd - *puntero_offset;   // Leeremos lo que podemos
30  #ifdef depura
31          printk(PRINTK_LEVEL "LCD_depura: LCD_read: Lo que nos piden sobrepasa el
32          límite de la memoria del LCD, así que solo leemos lo que podemos hasta llegar
33          al final.  retval = %d\n",retval);
34  #endif
35      } else {                                  // Si no lo sobrepasa...
36          retval = tamano;                      // Leemos lo que nos piden
37  #ifdef depura
38          printk(PRINTK_LEVEL "LCD_depura: LCD_read: La cantidad de datos solicitados
39          no se extienden por fuera de los límites de la memoria del LCD, así que leeremos
40          los datos solicitados. retval = %d\n",retval);
41  #endif
42      }
43
44  #ifdef depura
45      printk(PRINTK_LEVEL "LCD_depura: LCD_read: Voy a leer %d datos\n", retval);
46  #endif
47
48      if(retval > 0) {
49          /* Se realiza el traspaso de la información desde el driver el       *
50           * usuario. En caso de salir mal entonces se envía el código del     *
51           * error.                                                            */
52          if(copy_to_user(puntero_usuario, el_LCD->mem_lcd+(long unsigned int)*puntero_offset,
53          retval))
54              retval = -EFAULT;
55          else {
```

```
56 #ifdef depura
57             printk(PRINTK_LEVEL "LCD_depura: LCD_read: Se leyeron bien los datos\n");
58 #endif
59             /* En caso de que todo salga bien entonces se actualiza el        *
60              * puntero con los datos leidos.                                   */
61             *puntero_offset += retval;
62         }
63 #ifdef depura
64     printk(PRINTK_LEVEL "LCD_depura: LCD_read: Al salir de la función, *puntero_offset =
65     %lu\n\n",(long unsigned int)*puntero_offset);
66 #endif
67     }
68     return retval;
69 }
```

Script I.5: plantilla del driver sección función read

```
1
2  /* Función LCD_write: El kernel solicita escribir tamano bytes al driver. La  *
3   * función efectua la lectura y devuelve la cantidad de bytes que pudo leer.  *
4   * Además actualiza el puntero del usuario puntero3 según los bytes leidos.   */
5  ssize_t LCD_write (struct file *puntero_file, const char __user *puntero_usuario,
6  size_t tamano, loff_t * puntero_offset)
7  {
8      /* Variable para el for */
9      int k;
10     /* Variable temporal usada para formar elcomando                          */
11     char letra;
12     /* Variable que almacena lo que se va a escribir en el registro del LCD   */
13     u32 elcomando;
14     /* Variable que almacena el valor a devolver (Bytes escritos)             */
15     ssize_t retval;
16     /* Conserva un puntero a la estructura de tipo LCD_dev                    */
17     struct LCD_dev *el_LCD = puntero_file->private_data;
18 #ifdef depura
19     printk(PRINTK_LEVEL "LCD_depura: LCD_write: Estoy ejecutando la función LCD_write\n");
20     printk(PRINTK_LEVEL "LCD_depura: LCD_write: Tamano = %d\n",tamano);
21     printk(PRINTK_LEVEL "LCD_depura: LCD_write: El contenido de puntero_offset es:
22     %lu\n",(long unsigned int)*puntero_offset);
23 #endif
24     /* Dejamos en tamano el mayor entre tamano y tamano_del_paquete porque el *
25     * driver no piensa escribir mas que eso (perezoso)                        */
26     if(tamano>tamano_del_paquete)
27         retval = tamano_del_paquete;
28     else
29         retval = tamano;
30
31     /* Se realiza el traspaso de la información hacia el driver el usuario.   *
32          * En caso de salir mal entonces se envía el código del error.         */
33     if(copy_from_user(el_LCD->mem_lcd+(long unsigned int)*puntero_offset, puntero_usuario,
34     retval))
35         retval = -EFAULT;
36
```

```
37 #ifdef depura
38     printk(PRINTK_LEVEL "LCD_depura: LCD_write: Se escribieron bien los datos en el buffer
39     del LCD. \n");
40     printk(PRINTK_LEVEL "LCD_depura: LCD_write: El valor del puntero virtual del I/O Port
41     as I/O Memory base es: %p\n",LCD_iomem_pointer);
42     printk(PRINTK_LEVEL "LCD_depura: LCD_write: Con este puntero se harán todas las
43     escrituras al LCD\n");
44 #endif
45
46    /* Ahora se enviará caracter por caracter la información al LCD          */
47     for(k=0; k<retval; k++) {
48         /* Inicialmente se deben envíar ceros para borrar el registro del LCD */
49         elcomando = 0x00000000;
50         iowrite32(elcomando,LCD_iomem_pointer);
51
52 #ifdef depura
53         printk(PRINTK_LEVEL "LCD_depura: LCD_write: Escribi el valor = 0x%08X\r\n",
54         elcomando);
55 #endif
56         /* Estas barreras son necesarias para evitar optimizaciones del       *
57          * compilador al darse cuenta que se está escribiendo sobre el mismo  *
58          * registro                                                           */
59         barrier();
60         mb();
61
62         /* Entre cada una de las escrituras es necesario esperar un tiempo    */
63         msleep(1);
64
65         /* Ahora si se envía el caracter al LCD y para eso se forma elcomando *
66          * con las máscaras definidas en lcd/lcd_lib.h                        */
67         letra = *(el_LCD->mem_lcd + (*puntero_offset + k));
68         elcomando = (LCD_W_DATO | LCD_WRITE | LCD_START | (letra & LCD_DATO));
69         /* Explicación de porque hay que cambiar el endianness al dato ??     */
70         elcomando = change_endianness(elcomando);
71         iowrite32(elcomando,LCD_iomem_pointer);
72
73 #ifdef depura
74         printk(PRINTK_LEVEL "LCD_depura: LCD_write: Escribi el valor = 0x%08X\r\n",
75         elcomando);
76 #endif
77     }
78
79 #ifdef depura
80     printk(PRINTK_LEVEL "LCD_depura: LCD_write: Se escribieron bien los datos en el
81     LCD.\n");
82 #endif
83
84     /* En caso de que todo salga bien entonces se actualiza el puntero con    *
85      * los datos leidos.                                                      */
86     *puntero_offset += retval;
87
88 #ifdef depura
89     printk(PRINTK_LEVEL "LCD_depura: LCD_write: El contenido de puntero_offset al final de
```

```
90      la función write fue: %lu\n",(long unsigned int)*puntero_offset);
91  #endif
92      return retval;
93  }
```

Script I.6: plantilla del driver sección función write

```
1   /* Función LCD_ioctl:                                                    */
2   int LCD_ioctl (struct inode *inode1, struct file *puntero1, unsigned int comando, unsigned
3   long parametro)
4   {
5   /* Variable temporal usada para formar elcomando                         */
6       char letra;
7       /* Variable que almacena lo que se va a escribir en el registro del LCD   */
8       u32 elcomando;
9
10  #ifdef depura
11          printk(PRINTK_LEVEL "LCD_depura: LCD_ioctl: Estoy ejecutando la función LCD_ioctl\n");
12  #endif
13
14  switch(comando)
15  {
16      case BORRAR_LCD :
17          elcomando = 0x00000000;
18          iowrite32(elcomando,LCD_iomem_pointer);
19
20          /* Estas barreras son necesarias para evitar optimizaciones del       *
21           * compilador al darse cuenta que se está escribiendo sobre el mismo  *
22           * registro                                                           */
23          barrier();
24          mb();
25
26          /* Entre cada una de las escrituras es necesario esperar un tiempo    */
27          msleep(1);
28
29          letra = LCD_COMANDO_CLEAR_DISPLAY;
30          elcomando = (LCD_COMANDO | LCD_WRITE | LCD_START | (letra & LCD_DATO));
31          /* Explicación de porque hay que cambiar el endianness al dato ??     */
32          elcomando = change_endianness(elcomando);
33          iowrite32(elcomando,LCD_iomem_pointer);
34
35          break;
36
37      case SET_ADDRESS :
38          elcomando = 0x00000000;
39          iowrite32(elcomando,LCD_iomem_pointer);
40
41          /* Estas barreras son necesarias para evitar optimizaciones del       *
42           * compilador al darse cuenta que se está escribiendo sobre el mismo  *
43           * registro                                                           */
44          barrier();
45          mb();
46
```

```
47          /* Entre cada una de las escrituras es necesario esperar un tiempo    */
48          msleep(1);
49
50          letra = LCD_COMANDO_SET_ADDRESS | parametro;
51          elcomando = (LCD_COMANDO | LCD_WRITE | LCD_START | (letra & LCD_DATO));
52          /* Explicación de porque hay que cambiar el endianness al dato ??     */
53          elcomando = change_endianness(elcomando);
54          iowrite32(elcomando,LCD_iomem_pointer);
55
56
57          break;
58
59      default:
60          return -ENOTTY;
61  }
62
63  return 0;
64  }
```

Script I.7: plantilla del driver sección función ioctl

```
1   /* Función LCD_open: Esta función se llama cuando algún proceso del sistema   *
2    * operativo abre el nodo /dev/LCD como si fuera un archivo.                  */
3   int LCD_open (struct inode *puntero_inode, struct file *puntero_file)
4   {
5       /* Este puntero almacenará el puntero al LCD_dev que devolverá la función *
6        * container_of y posteriormente se almacena en el puntero_file para que  *
7        * pueda ser usado por las demás funciones del módulo                      */
8       struct LCD_dev *el_LCD;
9       /* Representa el tamaño del LCD que se va a manejar. Esto es calculado    *
10       * según el número de filas y columnas de la pantalla                      */
11      int tamano;
12      /* Variable del for                                                        */
13      int k;
14  #ifdef depura
15      printk(PRINTK_LEVEL "LCD_depura: LCD_open: Estoy ejecutando la función LCD_open\n");
16  #endif
17      /* Esta función busca cual struct LCD_dev contiene puntero_inode->i_cdev, *
18       * el cual es de tipo cdev.                                                */
19      el_LCD = container_of(puntero_inode->i_cdev,struct LCD_dev,cdev);
20      /* El puntero_file llega a todas las funciones, por ello se aprovecha el  *
21       * campo private_data para almacenar el puntero                            */
22      puntero_file->private_data = el_LCD;
23      /* Pregunta si alguna vez ha sido invocado el open para saber si debe     *
24       * asignar los parámetros del periférico                                   */
25      if(bandera==0) {
26  #ifdef depura
27          printk(PRINTK_LEVEL "LCD_depura: LCD_open: Nunca me han inicializado :(\n");
28  #endif
29          el_LCD->baseaddress = 0x83C18000;
30          el_LCD->col_visibles = 16;
31          el_LCD->cursor = 0;
32          el_LCD->blink = 0;
```

```
33          el_LCD->filas = 2;
34          el_LCD->columnas = 64;
35          tamano = el_LCD->filas*el_LCD->columnas;
36          el_LCD->mem_lcd = kmalloc(tamano*sizeof(char),GFP_KERNEL);
37          for(k=0;k<tamano;k++)
38              el_LCD->mem_lcd[k] = 0;
39          bandera=1;
40      } else {
41  #ifdef depura
42          printk(PRINTK_LEVEL "LCD_depura: LCD_open: Ya estaba inicializado :)\n");
43  #endif
44      }
45      /* Reserva del puerto... esto se refleja inmediatamente en /proc/ioports  */
46      if(!request_region(LCD_BASEADDRESS,4,LCD_NAME))
47          printk(PRINTK_LEVEL "LCD_open: No se pudo hacer exitosamente el request del puerto
48          LCD :(\n");
49
50      /* Se mapea la dirección física del puerto en la porción de memoria tipo  *
51       * I/O. El tamaño de la reserva es de 64 bytes                             */
52      LCD_iomem_pointer=ioremap(LCD_BASEADDRESS,0x00000040);
53
54      return 0;
55  }
```

Script I.8: plantilla del driver sección función open

```
1
2  /* Función LCD_release: Esta función libera lo que la función LCD_open ha     *
3   * reservado.                                                                 */
4  int LCD_release (struct inode *inode1, struct file *puntero1)
5  {
6  #ifdef depura
7      printk(PRINTK_LEVEL "LCD_depura: LCD_release: Estoy ejecutando la función
8      LCD_release\n");
9  #endif
10     /* liberacion de la memoria I/O sobre la que se mapeo el puerto           */
11     iounmap(LCD_iomem_pointer);
12
13     /* Liberación del puerto reservado                                        */
14     release_region(LCD_BASEADDRESS,4);
15     return 0;
16 }
```

Script I.9: plantilla del driver sección función release

```
1
2  /* En esta estructura se definen las operaciones que se van implementar en el *
3   * driver                                                                     */
4  struct file_operations LCD_fops = {
5          .owner  = THIS_MODULE,
6          .llseek = LCD_llseek,
7          .read   = LCD_read,
```

```
8           .write  = LCD_write,
9           .ioctl  = LCD_ioctl,
10          .open   = LCD_open,
11          .release = LCD_release
12  };
```

Script I.10: plantilla del driver sección file operation

```
1
2   /* Esta estructura representa los dispositivos LCD que se implementaron. Para *
3    * este caso fue solo uno. La memoria que emplea este dispositivo es          *
4    * reservada en LCD_init                                                      */
5   struct LCD_dev *my_LCD_dev;
```

Script I.11: plantilla del driver sección LCD dev

```
1   /* Función LCD_init: Esta función es llamada cuando se hace insmod y debe     *
2    * reservar los números major y minor y registrar el dispositivo.             */
3   static int LCD_init(void)
4   {
5       /* Variable usada para recibir el código de error de la función cdev_init */
6       int err;
7   #ifdef depura
8       /* Variable del for que repite el mensaje de saludo.                      */
9       int k;
10  #endif
11      /* Variable que recibe el valor de la función alloc_chrdev_region         */
12      int result;
13
14      /* Reserva dinámica de los números del driver.                            */
15      result = alloc_chrdev_region(&mydev,firstminor,count,nombre);
16  #ifdef depura
17      if(result==0)
18          printk(PRINTK_LEVEL "LCD_depura: LCD_init: Los números reservados para el driver
19          fueron:\n Major: %d\n Minor: %d\n",MAJOR(mydev),MINOR(mydev));
20      else
21          printk(PRINTK_LEVEL "LCD_depura: LCD_init: Hubo un error y los números no se
22          reservaron. El error fue: %d\n ",result);
23  #endif
24
25      /* Reserva memoria en el espacio del kernel para LCD_dev, es decir, que   *
26       * inicializa el puntero my_LCD_dev                                       */
27      my_LCD_dev = kmalloc(sizeof(struct LCD_dev), GFP_KERNEL);
28  #ifdef depura
29      if(my_LCD_dev==NULL)
30          printk(PRINTK_LEVEL "LCD_depura: LCD_init: Pailas, tocó reiniciar, no pude reservar
31           memoria\n");
32      else
33      printk(PRINTK_LEVEL "LCD_depura: LCD_init: Congratulations, sos un duro, reservaste
34      memoria para my_LCD_dev, no tocó reiniciar, se reservaron %d Bytes\n ",
35      (int) sizeof(struct LCD_dev));
36  #endif
```

```
37
38      /* No me acuerdo....                                          */
39      cdev_init(&my_LCD_dev->cdev,&LCD_fops);
40      /* Registro del driver en el kernel                           */
41      err = cdev_add(&my_LCD_dev->cdev, mydev, 1);
42      /* Inicializa algunos campos de la estructura my_LCD_dev      */
43      my_LCD_dev->cdev.owner = THIS_MODULE;
44      my_LCD_dev->cdev.ops = &LCD_fops;
45
46  #ifdef depura
47      if(err<0)
48          printk(PRINTK_LEVEL "LCD_depura: LCD_init: Hubo un error al registrar el
49          dispositivo. El error fue: %d\n ",err);
50      /* Mensaje de saludo para demostrar el uso de los parámetros  */
51      for (k=0;k<howmany;k++)
52          printk( "LCD_hello: LCD_init: Hola %s\n",whom);
53      /* Prueba que permite saber cual proceso lo invoca a uno      */
54      printk(PRINTK_LEVEL "LCD_depura: LCD_init: He sido invocado por:[%i] %s\n",current->pid,
55      current->comm);
56  #endif
57
58      /* Se pone la bandera a cero para indicar que el driver nunca ha sido     *
59       * inicializado                                               */
60      bandera=0;
61      return result;
62  }
```

Script I.12: plantilla del driver sección función Init

```
1   /* Función LCD_exit: Esta función es invocada cuando se ejecuta rmmod y debe  *
2    * liberar y deshacer lo que la función LCD_init ha reservado                 */
3   static void LCD_exit(void)
4   {
5       /* Desregistra los números major y minor que le fueron asignados de forma *
6        * dinámica*/
7       unregister_chrdev_region(mydev,count);
8       /* Remueve el dispositivo del kernel                          */
9       cdev_del(&my_LCD_dev->cdev);
10      /* Libera la memoria de buffer del mensaje en el LCD*/
11      if(&my_LCD_dev->mem_lcd != NULL)
12          kfree(my_LCD_dev->mem_lcd);
13      /* Libera la memoria reservada para el struct LCD_dev         */
14      kfree(my_LCD_dev);
15  #ifdef depura
16      /* Mensaje de despedida                                       */
17      printk(PRINTK_LEVEL "LCD_hello: LCD_exit: Bye bye, Mundo\n");
18      /* Prueba que permite saber cual proceso lo invoca a uno      */
19      printk(PRINTK_LEVEL "LCD_depura: LCD_exit: He sido invocado por:[%i] %s\n",current->pid,
20      current->comm);
21  #endif
22  }
```

Script I.13: plantilla del driver sección función exit

```
1 /* Indica cuales funciones son las que se deben ejecutar al ejecutar insmod y *  * rmmod.
  */
2 module_init(LCD_init);
3 module_exit(LCD_exit);
4
5 /* Información adicional sobre el módulo.                                    */
6 MODULE_AUTHOR("William Salamanca y Sergio Abreo.");
7 MODULE_ALIAS("LCD_template");
8 MODULE_DESCRIPTION("Driver que maneja la pantalla LCD y que sirve como plantilla para
9 controladores de otro tipo de periféricos.");
```

Script I.14: plantilla del driver sección de carga y liberación

I.2 Plantilla del driver modificada para el procesador del factor siete.

```
1
2 /****************************
3  * Librerías personalizadas *
4  ****************************/
5
6 /* Definiciones para el LCD                                                  */
7 #include"LCD.h"
8 #include"PERIFERICO.h"
9 /* Función importada del EDK                                                 */
10 #include"lcd/lcd_lib.h"
11 #include"lcd/lcd_lib.c"
12
13 /****************************
14  * Librerías personalizadas *
15  ****************************/
16
17 /* Cantidad de caracteres que se copian cada vez que se llama las funciones   *
18  * LCD_write o LCD_read. Para este caso son 19 registros de 32 bits y con cada
19  escritura traigo un byte                                                    */
20 #define tamano_del_paquete 76 //19 reg * 4 bytes
21
22 /* Habilita la impresión de mensajes de depuración mediante printk           */
23 //#define depura
24 #define PRINTK_LEVEL KERN_INFO
25
26 /* Datos del periférico LCD                                                  */
27 #define PERIFERICO_BASEADDRESS 0x814C0000
28 #define PERIFERICO_NAME "Puerto_PROCESADOR"
29
30 #define PERIFERICO_MAGIC '&'
31 #define BORRAR_LCD _IO(PERIFERICO_MAGIC,1)
32 #define SET_ADDRESS _IOW(PERIFERICO_MAGIC,2,int)
33
34
35 /* Puntero devuelto por ioremap                                              */
```

```
36 static void * LCD_iomem_pointer;
37
38 /* Licencia bajo la cual se desarrolló el driver                    */
39 MODULE_LICENSE("Dual BSD/GPL");
```

Script I.15: Sección librerías personalizadas versión final

```
1
2 /*****************************************
3  * Parámetros al momento de cargar el módulo *
4  *****************************************/
5 static char *whom   = "Mundo";
6 static int  howmany = 1;
7 module_param(whom,     charp, S_IRUGO);
8 module_param(howmany, int,   S_IRUGO);
9
10 /************************
11  * Parámetros del driver *
12  ************************/
13
14 /* Nombre que aparecerá en el /proc/devices                         */
15 static char *nombre = "PROC_FACT_7";
16 /* El primer minor number es cero                                   */
17 static unsigned int firstminor = 0;
18 /* Indica cuantos periféricos existen, para reservar igual cantidad de minor *
19  * numbers.                                                         */
20 static unsigned int count = 1;
21
22 /* Dato de tipo dev_t que se usa en las funciones LCD_init y LCD_exit y que  *
23  * agrupa los major y minor numbers asignados al periférico          */
24 static dev_t mydev;
25 /* Indica si el driver ha sido inicializado, en otras palabras, si alguna vez *
26  * se ha ejecutado la función open.                                 */
27 static char bandera;
28
29
30 /* Función change_endianness: Tiene como objetivo cambiar el endianness a un  *
31  * dato de 32 bits tipo u32.                                        */
32 u32 change_endianness(u32 dato)
33 {
34     u32 valret=0;
35     *((u8 *) &valret)+3) = *((u8 *)&dato)+0);
36     *((u8 *) &valret)+2) = *((u8 *)&dato)+1);
37     *((u8 *) &valret)+1) = *((u8 *)&dato)+2);
38     *((u8 *) &valret)+0) = *((u8 *)&dato)+3);
39     return valret;
40 }
```

Script I.16: Sección parámetros del driver versión final

```
1
2 ssize_t LCD_write (struct file *puntero_file, const char __user *puntero_usuario, size_t tamano,
```

```
3  loff_t * puntero_offset)
4  {
5      /* Variable para el for */
6      int k;
7      int k1 =0;
8      /* Variable temporal usada para formar elcomando                    */
9      unsigned int letra;
10     /* Variable que almacena lo que se va a escribir en los registros del periférico */
11     u32 elcomando;
12     /* Variable que almacena el valor a devolver (Bytes escritos)       */
13     ssize_t retval;
14     /* Conserva un puntero a la estructura de tipo PERIFERICO_dev       */
15     struct PERIFERICO_dev *el_PERIFERICO = puntero_file->private_data;
16 #ifdef depura
17     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Estoy ejecutando la función
18     PERIFERICO_write\n");
19     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Tamano = %d\n",tamano);
20     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: El contenido de puntero_offset
21     es: %lu\n",(long unsigned int)*puntero_offset);
22 #endif
23     /* tamano es la cantidad de bytes que se desean escribir y tamano_de_paquete es la cantidad
24     de registros de 32 bits que se pueden escribir */
25     if(tamano>tamano_del_paquete)
26         retval = tamano_del_paquete;
27     else
28         retval = tamano;/*se ajusta en retval cuantos registros de 32 bits se necesitan para
29         guardar los bytes que se desean escribir*/
30
31     /* Se realiza el traspaso de la información hacia el driver el usuario.   *
32      * En caso de salir mal entonces se envía el código del error.            */
33     if(copy_from_user(el_PERIFERICO->intercambio_de_datos + (long unsigned int)*puntero_offset,
34      puntero_usuario, retval))
35         retval = -EFAULT;
36
37 #ifdef depura
38     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Se escribieron bien los datos en
39     el buffer del PERIFERICO. \n");
40     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: El valor del puntero virtual del
41      I/O Port as I/O Memory base es: %p\n",LCD_iomem_pointer);
42     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Con este puntero se harán todas
43                 las escrituras al PERIFERICO\n");
44 #endif
45
46    /* Ahora se enviará caracter por caracter la información al PERIFERICO     */
47
48
49         for(k=0; k<retval; k=k+4) {
50
51 #ifdef depura
52         printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: El valor de K = %X\r\n",k);
53 #endif
54         /* Estas barreras son necesarias para evitar optimizaciones del       *
55          * compilador al darse cuenta que se está escribiendo sobre el mismo  *
```

```
56          * registro                                                    */
57         barrier();
58         mb();
59
60         letra = *(el_PERIFERICO->intercambio_de_datos + (*puntero_offset + k1));
61
62         k1=k1+1;
63
64         elcomando = change_endianness(letra);
65
66         iowrite32(elcomando,LCD_iomem_pointer + *puntero_offset + k);
67
68 #ifdef depura
69         printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Escribi el valor 0 to 31 =
70          0x%08X\r\n",elcomando);
71         printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Escribi en la direccion =
72         %X\r\n",(unsigned int)LCD_iomem_pointer + k);
73 #endif
74
75 }
76
77
78 #ifdef depura
79     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: Se escribieron bien los datos en
80      el PERIFERICO.\n");
81 #endif
82
83     /* En caso de que todo salga bien entonces se actualiza el puntero con    *
84      * los datos leidos.                                                      */
85     *puntero_offset += retval;
86
87 #ifdef depura
88     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_write: El contenido de puntero_offset al
89      final de la función write fue: %lu\n",(long unsigned int)*puntero_offset);
90 #endif
91     return retval;
92 }
```

Script I.17: Sección de escritura versión final

```
1
2  ssize_t LCD_read (struct file *puntero_file, char __user *puntero_usuario, size_t tamano,
3  loff_t * puntero_offset)
4  {
5      /* Variable que almacena el valor a devolver (Bytes leidos)              */
6      ssize_t retval;
7      /* Conserva un puntero a la estructura de tipo PERIFERICO_dev                    */
8      struct PERIFERICO_dev *el_PERIFERICO = puntero_file->private_data;
9      /* Tamaño del buffer calculado según el número de registros       */
10     int cantidad_de_registros = el_PERIFERICO->numero_de_registros;
11
12     unsigned int leido;
13     /* Variable para el for */
```

```
14      int k;
15      int k1 =0;
16
17
18  #ifdef depura
19      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Estoy ejecutando la función
20      PERIFERICO_read\n");
21      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Tamano = %d\n",tamano);
22      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: El contenido de puntero offset es:
23       %lu\n",(long unsigned int)*puntero_offset);
24  #endif
25      /* Dejamos en tamano el menor entre tamano y tamano_del_paquete porque el *
26       * driver no piensa leer mas que eso (él es perezoso)                     */
27      if(tamano>tamano_del_paquete)
28          tamano = tamano_del_paquete;
29
30      /* Verificamos si se puede sobrepasar el tamaño máximo del buffer y       *
31       * limitamos la cantidad de datos leidos en caso de sobrepasarnos         */
32      if((*puntero_offset + tamano)>=cantidad_de_registros*4) { // Si sobrepasa límite...
33          retval = cantidad_de_registros*4 - *puntero_offset;   // Leeremos lo que podemos
34  #ifdef depura
35          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Lo que nos piden sobrepasa la
36           cantidad de registros del PERIFERICO, así que solo leemos lo que podemos hasta llegar
37            al final.  retval = %d\n",retval);
38  #endif
39      } else {                                  // Si no lo sobrepasa...
40          retval = tamano;                      // Leemos lo que nos piden
41  #ifdef depura
42          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: La cantidad de datos solicitados
43           no excede el numero de registros del PERIFERICO, así que leeremos los datos solicitados.
44            retval = %d\n",retval);
45  #endif
46      }
47
48  #ifdef depura
49      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Voy a leer %d datos\n", retval);
50  #endif
51
52      if(retval > 0) {
53          for(k=0; k<retval; k=k+4) {
54          /* Estas barreras son necesarias para evitar optimizaciones del       *
55           * compilador al darse cuenta que se está escribiendo sobre el mismo  *
56           * registro                                                           */
57          barrier();
58          mb();
59
60
61          leido=ioread32(LCD_iomem_pointer + (*puntero_offset + k));
62          leido = change_endianness(leido);
63  #ifdef depura
64          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: El valor de K = %X\r\n",k);
65          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Lei el valor 0 to 31 = %X\r\n",
66          leido);
```

```
67 #endif
68
69
70          *(el_PERIFERICO->intercambio_de_datos + (*puntero_offset + k1))=leido;
71          k1=k1+1;
72
73 }
74
75          /* Se realiza el traspaso de la información desde el driver el        *
76           * usuario. En caso de salir mal entonces se envía el código del      *
77           * error.                                                             */
78          if(copy_to_user(puntero_usuario, el_PERIFERICO->intercambio_de_datos +
79          (long unsigned int)*puntero_offset, retval))
80              retval = -EFAULT;
81          else {
82 #ifdef depura
83              printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Se leyeron bien los datos\n");
84 #endif
85              /* En caso de que todo salga bien entonces se actualiza el        *
86               * puntero con los datos leidos.                                  */
87              *puntero_offset += retval;
88          }
89 #ifdef depura
90      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_read: Al salir de la función,
91       *puntero_offset = %lu\n\n",(long unsigned int)*puntero_offset);
92 #endif
93      }
94      return retval;
95 }
```

Script I.18: Sección de lectura versión final

```
1
2  loff_t LCD_llseek (struct file *puntero_file, loff_t off, int whence)
3  {
4      /* Variable que almacena el valor a devolver (nuevo puntero)              */
5      loff_t retval;
6      /* Conserva un puntero a la estructura de tipo PERIFERICO_dev             */
7      struct PERIFERICO_dev *el_PERIFERICO = puntero_file->private_data;
8  #ifdef depura
9          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_llseek: Estoy ejecutando la función
10         PERIFERICO_llseek\n");
11 #endif
12     switch(whence){
13         case 0: /* SEEK_SET  */
14                 retval = off;
15         #ifdef depura
16         printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_llseek: Estoy ejecutando SEEK_SET\n");
17         #endif
18                 break;
19         case 1: /* SEEK_CUR  */
20                 retval = puntero_file->f_pos + off;
21         #ifdef depura
```

```
22          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_llseek: Estoy ejecutando SEEK_CUR\n");
23          #endif
24                  break;
25          case 2: /* SEEK_END  */
26                  retval = el_PERIFERICO->numero_de_registros*4 + off;
27          #ifdef depura
28          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_llseek: Estoy ejecutando SEEK_END\n");
29          #endif
30                  break;
31
32          default: /* can´t happen  */
33                  return -EINVAL;
34          }
35          if (retval < 0) return -EINVAL;
36          puntero_file->f_pos = retval;
37  #ifdef depura
38          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_llseek: Salí bien de la función
39          PERIFERICO_llseek\n");
40  #endif
41          return retval;
42  }
```

Script I.19: Sección llseek versión final

```
1
2  int LCD_open (struct inode *puntero_inode, struct file *puntero_file)
3  {
4      /* Este puntero almacenará el puntero al PERIFERICO_dev que devolverá la función *
5       * container_of y posteriormente se almacena en el puntero_file para que  *
6       * pueda ser usado por las demás funciones del módulo                      */
7      struct PERIFERICO_dev *el_PERIFERICO;
8      /* Representa el tamaño del periférico que se va a manejar. Esto es calcu-*
9       * lado según el número de registros de 32 bits que se esten usando.       */
10     int tamano;
11     /* Variable del for                                                        */
12     int k;
13 #ifdef depura
14     printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_open: Estoy ejecutando la función
15     PERIFERICO_open\n");
16 #endif
17     /* Esta función busca cual struct PERIFERICO_dev contiene puntero_inode->i_cdev, *
18      * el cual es de tipo cdev.                                                */
19     el_PERIFERICO = container_of(puntero_inode->i_cdev,struct PERIFERICO_dev,cdev);
20     /* El puntero_file llega a todas las funciones, por ello se aprovecha el  *
21      * campo private_data para almacenar el puntero                            */
22     puntero_file->private_data = el_PERIFERICO;
23     /* Pregunta si alguna vez ha sido invocado el open para saber si debe      *
24      * asignar los parámetros del periférico                                   */
25     if(bandera==0) {
26 #ifdef depura
27         printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_open: Nunca me han inicializado :(\n");
28 #endif
29         el_PERIFERICO->baseaddress = 0x814C0000;
```

```
30          el_PERIFERICO->numero_de_registros = 19;
31          strcpy(el_PERIFERICO->nombre_periferico, "Procesador factor 7");
32          el_PERIFERICO->ocupado = 10;
33          tamano = el_PERIFERICO->numero_de_registros;
34          el_PERIFERICO->intercambio_de_datos = kmalloc(tamano*sizeof(unsigned int),GFP_KERNEL);
35          for(k=0;k<tamano;k++)
36              el_PERIFERICO->intercambio_de_datos[k] = 0;
37          bandera=1;
38  #ifdef depura
39          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_open: El mensaje guardado en
40          el_PERIFERICO->nombre_periferico es: %s\n", el_PERIFERICO->nombre_periferico );
41  #endif
42
43      } else {
44  #ifdef depura
45          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_open: Ya estaba inicializado :)\n");
46  #endif
47      }
48      /* Reserva del puerto... esto se refleja inmediatamente en /proc/ioports  */
49      if(!request_region(PERIFERICO_BASEADDRESS,4,PERIFERICO_NAME))
50          printk(PRINTK_LEVEL "PERIFERICO_open: No se pudo hacer exitosamente el request del
51          puerto PERIFERICO :(\n");
52
53      /* Se mapea la dirección física del puerto en la porción de memoria tipo  *
54       * I/O. El tamaño de la reserva es de 64 bytes                             */
55      LCD_iomem_pointer=ioremap(PERIFERICO_BASEADDRESS,0x00000040);
56
57      return 0;
58  }
```

Script I.20: Sección open versión final

```
1
2  int LCD_release (struct inode *inode1, struct file *puntero1)
3  {
4  #ifdef depura
5      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_release: Estoy ejecutando la función
6      PERIFERICO_release\n");
7  #endif
8      /* liberacion de la memoria I/O sobre la que se mapeo el puerto           */
9      iounmap(LCD_iomem_pointer);
10
11     /* Liberación del puerto reservado                                        */
12     release_region(PERIFERICO_BASEADDRESS,4);//sergio cambio
13     return 0;
14 }
```

Script I.21: Sección release versión final

```
1
2  struct file_operations LCD_fops = {
3          .owner   = THIS_MODULE,
```

```
4           .llseek = LCD_llseek,
5           .read    = LCD_read,
6           .write   = LCD_write,
7           .ioctl   = LCD_ioctl,
8           .open    = LCD_open,
9           .release = LCD_release
10  };
11
12  struct PERIFERICO_dev *my_PERIFERICO_dev;
```

Script I.22: Sección file operation versión final

```
1
2   static int LCD_init(void)
3   {
4       /* Variable usada para recibir el código de error de la función cdev_init */
5       int err;
6   #ifdef depura
7       /* Variable del for que repite el mensaje de saludo.                      */
8       int k;
9   #endif
10      /* Variable que recibe el valor de la función alloc_chrdev_region         */
11      int result;
12
13      /* Reserva dinámica de los números del driver.                            */
14      result = alloc_chrdev_region(&mydev,firstminor,count,nombre);
15  #ifdef depura
16      if(result==0)
17          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: Los números reservados para
18           el driver fueron:\n Major: %d\n Minor: %d\n",MAJOR(mydev),MINOR(mydev));
19      else
20          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: Hubo un error y los números
21           no se reservaron. El error fue: %d\n ",result);
22  #endif
23
24      /* Reserva memoria en el espacio del kernel para PERIFERICO_dev, es decir, que   *
25       * inicializa el puntero my_PERIFERICO_dev                                      */
26      my_PERIFERICO_dev = kmalloc(sizeof(struct PERIFERICO_dev), GFP_KERNEL);
27  #ifdef depura
28      if(my_PERIFERICO_dev==NULL)
29          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: Pailas, tocó reiniciar, no pude
30           reservar memoria\n");
31      else
32      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: Congratulations, sos un duro,
33      reservaste memoria para my_PERIFERICO_dev, no tocó reiniciar, se reservaron %d Bytes\n ",
34      (int) sizeof(struct PERIFERICO_dev));
35  #endif
36
37      /* No me acuerdo....                                          */
38      cdev_init(&my_PERIFERICO_dev->cdev,&LCD_fops);
39      /* Registro del driver en el kernel                                       */
40      err = cdev_add(&my_PERIFERICO_dev->cdev, mydev, 1);
41      /* Inicializa algunos campos de la estructura my_PERIFERICO_dev           */
```

```
42      my_PERIFERICO_dev->cdev.owner = THIS_MODULE;
43      my_PERIFERICO_dev->cdev.ops = &LCD_fops;
44
45  #ifdef depura
46      if(err<0)
47          printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: Hubo un error al registrar
48          el dispositivo. El error fue: %d\n ",err);
49      /* Mensaje de saludo para demostrar el uso de los parámetros              */
50      for (k=0;k<howmany;k++)
51          printk( "PERIFERICO_hello: PERIFERICO_init: Hola %s\n",whom);
52      /* Prueba que permite saber cual proceso lo invoca a uno                  */
53      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_init: He sido invocado por:[%i]
54       %s\n",current->pid,current->comm);
55  #endif
56
57      /* Se pone la bandera a cero para indicar que el driver nunca ha sido     *
58       * inicializado                                                          */
59      bandera=0;
60      return result;
61  }
```

Script I.23: Sección init versión final

```
1
2   static void LCD_exit(void)
3   {
4       /* Desregistra los números major y minor que le fueron asignados de forma *
5        * dinámica*/
6       unregister_chrdev_region(mydev,count);
7       /* Remueve el dispositivo del kernel                                      */
8       cdev_del(&my_PERIFERICO_dev->cdev);
9       /* Libera la memoria de buffer del mensaje en el LCD*/
10      if(&my_PERIFERICO_dev->intercambio_de_datos != NULL)
11          kfree(my_PERIFERICO_dev->intercambio_de_datos);
12      /* Libera la memoria reservada para el struct PERIFERICO_dev                */
13      kfree(my_PERIFERICO_dev);
14  #ifdef depura
15      /* Mensaje de despedida                                                   */
16      printk(PRINTK_LEVEL "PERIFERICO_hello: PERIFERICO_exit: Bye bye, Mundo\n");
17      /* Prueba que permite saber cual proceso lo invoca a uno                  */
18      printk(PRINTK_LEVEL "PERIFERICO_depura: PERIFERICO_exit: He sido invocado por:[%i]
19       %s\n",current->pid,current->comm);
20  #endif
21  }
22
23  /* Indica cuales funciones son las que se deben ejecutar al ejecutar insmod y *  * rmmod.
    */
24  module_init(LCD_init);
25  module_exit(LCD_exit);
26
27  /* Información adicional sobre el módulo.                                     */
28  MODULE_AUTHOR("William Salamanca y Sergio Abreo.");
29  MODULE_ALIAS("PERIFERICO_template");
```

```
30 MODULE_DESCRIPTION("Driver que maneja el periférico que calcula el factor siete y que sirve
31  como plantilla para controlar otros tipos de periféricos.");
```

Script I.24: Sección exit versión final

Apéndice J

Funciones y librerías para interactuar con el Driver

J.1 Funciones y Librerías.

J.1.1 periferico_lib.h

```
1
2  /*************************** PERIFERICO_lib.h ****************************
3   * En este archivo se definen las funciones que permiten usar el periferico *
4   * mas facilmente.                                                        *
5   *************************************************************************/
6  #ifndef PERIFERICO_LIB
7  #define PERIFERICO_LIB
8
9
10 /*************************************************************************
11  ************************** Constantes ***********************************
12  *************************************************************************/
13
14 /* Depuracion
   */
15 #define DEBUG_PERIFERICO
16
17 #define PERIFERICO_ESTADO_SIG          (0x003E0000)
18 #define PERIFERICO_ESTADO_S0           (0x00000000)
19 #define PERIFERICO_ESTADO_S1           (0x00020000)
20 #define PERIFERICO_ESTADO_S2           (0x00040000)
21 #define PERIFERICO_ESTADO_S3           (0x00060000)
22 #define PERIFERICO_ESTADO_S3_W         (0x00080000)
23 #define PERIFERICO_ESTADO_S4           (0x000A0000)
24 #define PERIFERICO_ESTADO_S4_W         (0x000C0000)
25 #define PERIFERICO_ESTADO_S5           (0x000E0000)
26 #define PERIFERICO_ESTADO_S5_W         (0x00100000)
27 #define PERIFERICO_ESTADO_S6           (0x00120000)
28 #define PERIFERICO_ESTADO_S6_W         (0x00140000)
```

```
29 #define PERIFERICO_ESTADO_S7                (0x00160000)
30 #define PERIFERICO_ESTADO_S7_W              (0x00180000)
31 #define PERIFERICO_ESTADO_S8                (0x001A0000)
32 #define PERIFERICO_ESTADO_S8_W              (0x001C0000)
33 #define PERIFERICO_ESTADO_S9                (0x001E0000)
34 #define PERIFERICO_ESTADO_S9_W              (0x00200000)
35 #define PERIFERICO_ESTADO_S10               (0x00220000)
36 #define PERIFERICO_ESTADO_S10_W             (0x00240000)
37 #define PERIFERICO_ESTADO_S11               (0x00260000)
38 #define PERIFERICO_ESTADO_S11_W             (0x00280000)
39 #define PERIFERICO_ESTADO_S12               (0x002A0000)
40 #define PERIFERICO_ESTADO_S12_W             (0x002C0000)
41 #define PERIFERICO_ESTADO_S13               (0x002E0000)
42 #define PERIFERICO_ESTADO_S13_W             (0x00300000)
43 #define PERIFERICO_ESTADO_S14               (0x00320000)
44 #define PERIFERICO_ESTADO_S14_W             (0x00340000)
45 #define PERIFERICO_OCUPADO                  (0x00010000)
46 #define PERIFERICO_RESET_ON                 (0x00008000)
47 #define PERIFERICO_RESET_OFF                (0x00000000)
48 #define PERIFERICO_INICIO_ON                (0x00004000)
49 #define PERIFERICO_INICIO_OFF               (0x00000000)
50 #define PERIFERICO_MU                       (0x00003C00)
51 #define PERIFERICO_MU_1                     (0x00000400)
52 #define PERIFERICO_MU_2                     (0x00000800)
53 #define PERIFERICO_MU_3                     (0x00000C00)
54 #define PERIFERICO_MU_4                     (0x00001000)
55 #define PERIFERICO_MU_5                     (0x00001400)
56 #define PERIFERICO_ADDRESS_DRIVER           (0x000003C0)
57 #define PERIFERICO_ADDRESS_DRIVER_0         (0x00000000)
58 #define PERIFERICO_ADDRESS_DRIVER_1         (0x00000040)
59 #define PERIFERICO_ADDRESS_DRIVER_2         (0x00000080)
60 #define PERIFERICO_ADDRESS_DRIVER_3         (0x000000C0)
61 #define PERIFERICO_ADDRESS_DRIVER_4         (0x00000100)
62 #define PERIFERICO_ADDRESS_DRIVER_5         (0x00000140)
63 #define PERIFERICO_ADDRESS_DRIVER_6         (0x00000180)
64 #define PERIFERICO_ADDRESS_DRIVER_7         (0x000001C0)
65 #define PERIFERICO_ADDRESS_DRIVER_8         (0x00000200)
66 #define PERIFERICO_ADDRESS_DRIVER_9         (0x00000240)
67 #define PERIFERICO_ADDRESS_DRIVER_10        (0x00000280)
68 #define PERIFERICO_ADDRESS_DRIVER_11        (0x000002C0)
69 #define PERIFERICO_ADDRESS_DRIVER_12        (0x00000300)
70 #define PERIFERICO_ADDRESS_DRIVER_13        (0x00000340)
71 #define PERIFERICO_ADDRESS_DRIVER_14        (0x00000380)
72 #define PERIFERICO_ADDRESS_DRIVER_15        (0x000003C0)
73 #define PERIFERICO_WRITE_ENAB_DRIV_ON       (0x00000020)
74 #define PERIFERICO_WRITE_ENAB_DRIV_OFF      (0x00000000)
75 #define PERIFERICO_RAM_ENAB_DRIV_ON         (0x00000010)
76 #define PERIFERICO_RAM_ENAB_DRIV_OFF        (0x00000000)
77 #define PERIFERICO_BLOQUE_RAM_SEL           (0x0000000F)
78 #define PERIFERICO_BLOQUE_RAM_SEL_0         (0x00000000)
79 #define PERIFERICO_BLOQUE_RAM_SEL_1         (0x00000001)
80 #define PERIFERICO_BLOQUE_RAM_SEL_2         (0x00000002)
81 #define PERIFERICO_BLOQUE_RAM_SEL_3         (0x00000003)
```

```
82 #define PERIFERICO_BLOQUE_RAM_SEL_4      (0x00000004)
83 #define PERIFERICO_BLOQUE_RAM_SEL_5      (0x00000005)
84 #define PERIFERICO_BLOQUE_RAM_SEL_6      (0x00000006)
85 #define PERIFERICO_BLOQUE_RAM_SEL_7      (0x00000007)
86 #define PERIFERICO_BLOQUE_RAM_SEL_8      (0x00000008)
87 #define PERIFERICO_BLOQUE_RAM_SEL_9      (0x00000009)
88 #define PERIFERICO_BLOQUE_RAM_SEL_10     (0x0000000A)
89 #define PERIFERICO_BLOQUE_RAM_SEL_11     (0x0000000B)
90 #define PERIFERICO_BLOQUE_RAM_SEL_12     (0x0000000C)
91 #define PERIFERICO_BLOQUE_RAM_SEL_13     (0x0000000D)
92 #define PERIFERICO_BLOQUE_RAM_SEL_14     (0x0000000E)
93 #define PERIFERICO_BLOQUE_RAM_SEL_15     (0x0000000F)
94 #define PERIFERICO_BANDERAS_22           (0xFFFF0000)
95 #define PERIFERICO_BANDERAS_11           (0x0000FFFF)
96 #define PERIFERICO_M                     (0x000FFFFF)
97
98
99 /*************************************************************************
100  ************************** Funciones ************************************
101  *************************************************************************/
102
103
104 /************************PERIFERICO_start***********************************/
105 /* Función que da inicio con el procesamiento de los datos, normalmente se  *
106  * ejecuta después de la función de carga de datos PERIFERICO_load          */
107 int PERIFERICO_start( size_t filedesc, int comando_in );
108
109
110 //Función que ajusta el puntero al registro del periférico que se desea escribir.
111 void PERIFERICO_seek_set(size_t filedesc, int puntero);
112
113 /************************PERIFERICO_load************************************/
114 /* Función que carga los datos que debe procesar el periférico, normalmente*
115  * se ejecuta de primera (se debe tener cuidado porque esta función cambia *
116  * los punteros jx que se definieron como globales) junto con el contenido *
117  * del vector z[x].                                                        */
118
119 int PERIFERICO_load( size_t filedesc, float z[224], int mu, int m, int j0, int j1, int j2, int j3,
120 int j4, int j5, int j6);
121
122 /************************PERIFERICO_extract*********************************/
123 /* Función que retorna los datos que se procesaron en el periférico,       *
124  * normalmente se ejecuta después de que la función PERIFERICO_start       *
125  * termina.                                                                */
126
127 int PERIFERICO_extract( size_t filedesc, float z[224], int m, int j_o[7]);
128
129 /************************PERIFERICO_stop ************************************/
130 /* Función que finaliza bruscamente el procesamiento de los datos,          *
131  * normalmente se ejecuta como una parada de emergencia, perdiéndose los    *
132  * datos calculados hasta ese momento                                      */
133
134 int PERIFERICO_stop( size_t filedesc );
```

```
135
136 /**********************PERIFERICO_debug***********************************/
137 /* Función que permite hacerle seguimiento a la maquina de estados junto con*
138  * las banderas de los módulos de las operaciones de punto flotante para    *
139  * depurar el funcionamiento del mismo.                                      */
140
141 int PERIFERICO_debug_estado(size_t filedesc );
142
143 int PERIFERICO_debug_ocupado(size_t filedesc );
144
145 /**********************PERIFERICO_write***********************************/
146 /* Función que permite hacer la escritura sobre los registros de configuración*
147  * del periférico. Hay dos tipos de escrituras posibles, como flotantes o  *
148  * como entero.                                                              */
149
150 int PERIFERICO_write( size_t filedesc, int comando, int size );
151 int PERIFERICO_write_f( size_t filedesc, float comando, int size);
152
153 /**********************PERIFERICO_read************************************/
154 /* Función que permite hacer la lectura de los registros de configuración *
155  * del periférico. Hay dos tipos de lecturas posibles, como flotantes o   *
156  * como entero.                                                            */
157
158 int PERIFERICO_read( size_t filedesc, int buf, int size );
159 float PERIFERICO_read_f( size_t filedesc, float buf, int size );
160 #endif
```

Script J.1: Librería periferico_lib.h

J.1.2 Cabecero de periferico_lib.c

```
1
2 /**********************periferico_lib.c ***********************************
3  * En este archivo se definen las funciones que permiten usar el periférico *
4  * mas fácilmente.                                                          *
5  ***************************************************************************/
6 #include "periferico_lib.h"
7 /*#define depura*/
8 /*#define depura1*/
9 /*#define depura2*/
10 /*#define depura3*/
11
12 extern struct PERIFERICO_dev *el_PERIFERICO;
```

Script J.2: Cabecero de periferico_lib.c

J.1.3 función PERIFÉRICO_principal

```
1
2  int PERIFERICO_principal(float z[224], float z_o[224], int m, int mu, int j0,
3  int j1, int j2, int j3, int j4, int j5, int j6, int j[7]){
4
5          int m_inv=0;
6  #ifdef depura2
7          int i;
8  #endif
9
10         size_t filedesc = open("/dev/PROC_FACT_7",O_RDWR);
11         if(filedesc < 0)
12         return 1;
13
14 #ifdef depura1
15         printf("Inicio PERIFERICO_stop \n");
16 #endif
17         m_inv=PERIFERICO_stop(filedesc);
18
19 #ifdef depura1
20         printf("Inicio PERIFERICO_load \n");
21         printf("El valor de m es %d\n",m);
22 #endif
23         m_inv=PERIFERICO_load(filedesc,z,mu,m,j0,j1,j2,j3,j4,j5,j6);
24 #ifdef depura1
25         printf("El valor de MU es %X\n",m_inv);
26         printf("Inicio PERIFERICO_extract \n");
27 #endif
28
29 #ifdef depura2
30         PERIFERICO_extract(filedesc,z_o,m,j);
31
32         for (i=0;i<7;i++){
33         printf("El valor de j%d_o es %d\n",i,j[i]);
34         }
35         for (i=0;i<m*14;i++){
36         printf("El valor de z_o[%d] es %f\n",i,z_o[i]);
37         }
38 #endif
39
40 #ifdef depura1
41         printf("Inicio PERIFERICO_start \n");
42 #endif
43         m_inv=PERIFERICO_start(filedesc,m_inv);
44
45 #ifdef depura1
46         printf("Inicio PERIFERICO_extract \n");
47
48 #endif
49         while(m_inv!=0){
50         m_inv =PERIFERICO_debug_ocupado(filedesc);
51 #ifdef depura3
52         printf("El valor de PERIFERICO_ocupado es %X\n",m_inv);
```

```
53 #endif
54         }
55         PERIFERICO_extract(filedesc,z_o,m,j);
56
57 #ifdef depura3
58         for (i=0;i<7;i++){
59         printf("El valor de j%d_o es %d\n",i,j[i]);
60         }
61         for (i=0;i<m*14;i++){
62         printf("El valor de z_o[%d] es %f\n",i,z_o[i]);
63         }
64 #endif
65
66         if(close(filedesc) < 0)
67         return 0;
68 }
```

Script J.3: Función PERIFÉRICO_principal

J.1.4 función PERIFÉRICO_start

```
1  int PERIFERICO_start( size_t filedesc, int comando_in ){
2          int comando,comando_in1;
3          int size=4;
4
5          comando_in1=comando_in;
6
7          PERIFERICO_seek_set(filedesc, 0);
8          comando = (PERIFERICO_INICIO_ON | (PERIFERICO_MU & comando_in1));
9          if(PERIFERICO_write( filedesc, comando, size )!=1)
10         {
11                 return 0;//no se pudo escribir
12         }
13
14         PERIFERICO_seek_set(filedesc, 0);
15         comando = (PERIFERICO_INICIO_OFF | (PERIFERICO_MU & comando_in1));
16         if(PERIFERICO_write( filedesc, comando, size )!=1)
17         {
18                 return 0;//no se pudo escribir
19         }
20 #ifdef depura
21         write(2,"Ya se inicio el periferico\n",27);
22 #endif
23         return 1;
24 }
```

Script J.4: Función PERIFÉRICO_start

J.1.5 función PERIFÉRICO_seek_set

```
1  void PERIFERICO_seek_set(size_t filedesc, int puntero){
2
3          if(lseek(filedesc,puntero,SEEK_SET) < 0)
4          {
5          write(2,"El ajuste del puntero fallo\n",26);
6          }
7
8  #ifdef depura
9          write(2,"El puntero se ajusto correctamente\n",35);
10 #endif
11 }
```

Script J.5: Función PERIFÉRICO_seek_set

J.1.6 función PERIFÉRICO_debug_estado

```
1
2  int PERIFERICO_debug_estado(size_t filedesc ){
3          int buf=0;
4          int estado;
5          int size=4;
6          int i=0;
7          PERIFERICO_seek_set(filedesc, i);
8          buf=PERIFERICO_read(filedesc,buf,size);
9          estado = (PERIFERICO_ESTADO_SIG & buf);
10         return estado;
11 }
```

Script J.6: Función PERIFÉRICO_debug_estado

J.1.7 función PERIFÉRICO_debug_ocupado

```
1  int PERIFERICO_debug_ocupado(size_t filedesc ){
2          int buf=0;
3          int ocupado;
4          int size=4;
5          int i=0;
6
7          PERIFERICO_seek_set(filedesc, i);
8          buf=PERIFERICO_read(filedesc,buf,size);
9          ocupado = (buf & PERIFERICO_OCUPADO);
10
11         return ocupado;
12 }
```

Script J.7: Función PERIFÉRICO_debug_ocupado

J.1.8 función PERIFÉRICO_stop

```
1
2  int PERIFERICO_stop( size_t filedesc ){
3          int comando;
4          int size=4;
5
6          PERIFERICO_seek_set(filedesc , 0);
7          comando = PERIFERICO_RESET_ON;
8
9          if(PERIFERICO_write( filedesc , comando, size )!=1)
10         {
11                 return 0;//no se pudo escribir
12         }
13
14         PERIFERICO_seek_set(filedesc , 0);
15         comando = PERIFERICO_RESET_OFF;
16
17         if(PERIFERICO_write( filedesc , comando, size )!=1)
18         {
19                 return 0;//no se pudo escribir
20         }
21
22 #ifdef depura
23         write(2,"Ya se reseteo el periferico\n",28);
24 #endif
25         return 1;
26
27 }
```

Script J.8: Función PERIFÉRICO_stop

J.1.9 función PERIFÉRICO_load

```
1  int PERIFERICO_load( size_t filedesc , float z[224], int mu, int m, int j0 , int j1 , int j2 , int j3 ,
2  int j4 , int j5 , int j6 ){
3          int dato_inv;
4          int size=4;
5          float dato_f;
6          int i,j;
7          int comando1, comando2, comando3;
8
9          //configuracion del registro j0
10         dato_f=j0;
11         PERIFERICO_seek_set(filedesc ,12);
12         PERIFERICO_write_f( filedesc , dato_f, size );
13
14         //configuracion del registro j1
15         dato_f=j1;
16         PERIFERICO_seek_set(filedesc ,16);
17         PERIFERICO_write_f( filedesc , dato_f, size );
```

```
18
19
20          //configuracion del registro j2
21          dato_f=j2;
22          PERIFERICO_seek_set(filedesc,20);
23          PERIFERICO_write_f( filedesc, dato_f, size );
24
25          //configuracion del registro j3
26          dato_f=j3;
27          PERIFERICO_seek_set(filedesc,24);
28          PERIFERICO_write_f( filedesc, dato_f, size );
29
30
31          //configuracion del registro j4
32          dato_f=j4;
33          PERIFERICO_seek_set(filedesc,28);
34          PERIFERICO_write_f( filedesc, dato_f, size );
35
36          //configuracion del registro j5
37          dato_f=j5;
38          PERIFERICO_seek_set(filedesc,32);
39          PERIFERICO_write_f( filedesc, dato_f, size );
40
41          //configuracion del registro j6
42          dato_f=j6;
43          PERIFERICO_seek_set(filedesc,36);
44          PERIFERICO_write_f( filedesc, dato_f, size );
45
46
47          //ajuste de m
48          dato_inv= m << 12;
49
50          //configuracion del registro M
51          PERIFERICO_seek_set(filedesc,72);
52          PERIFERICO_write( filedesc, dato_inv, size );
53
54          //Carga de la informacion en las memoria z00 a z06+1
55
56          comando1 = (PERIFERICO_WRITE_ENAB_DRIV_ON | PERIFERICO_RAM_ENAB_DRIV_ON );
57  #ifdef depura
58          printf("El valor de load comando1 es %X\n",comando1);
59  #endif
60                  for (i=0;i<m;i++){
61
62                          switch(i){
63                          case 0:
64                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_0 );
65                          break;
66                          case 1:
67                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_1 );
68                          break;
69                          case 2:
70                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_2 );
```

```
71                          break;
72                          case 3:
73                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_3 );
74                          break;
75                          case 4:
76                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_4 );
77                          break;
78                          case 5:
79                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_5 );
80                          break;
81                          case 6:
82                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_6 );
83                          break;
84                          case 7:
85                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_7 );
86                          break;
87                          case 8:
88                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_8 );
89                          break;
90                          case 9:
91                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_9 );
92                          break;
93                          case 10:
94                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_10 );
95                          break;
96                          case 11:
97                          comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_11 );
98                          break;
99                          case 12:
100                         comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_12 );
101                         break;
102                         case 13:
103                         comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_13 );
104                         break;
105                         case 14:
106                         comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_14 );
107                         break;
108                         case 15:
109                         comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_15 );
110                         break;
111                         default:
112                         comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_0 );
113                         }
114
115 #ifdef depura
116                         printf("El valor de load comando2 es %X\n",comando2);
117 #endif
118
119                         for (j=0;j<14;j++){
120                                 switch(j){
121                                 case 0:
122                                 comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_0;
123                                 break;
```

```
124                             case 1:
125                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_1;
126                             break;
127                             case 2:
128                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_2;
129                             break;
130                             case 3:
131                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_3;
132                             break;
133                             case 4:
134                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_4;
135                             break;
136                             case 5:
137                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_5;
138                             break;
139                             case 6:
140                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_6;
141                             break;
142                             case 7:
143                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_7;
144                             break;
145                             case 8:
146                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_8;
147                             break;
148                             case 9:
149                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_9;
150                             break;
151                             case 10:
152                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_10;
153                             break;
154                             case 11:
155                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_11;
156                             break;
157                             case 12:
158                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_12;
159                             break;
160                             case 13:
161                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_13;
162                             break;
163                             default:
164                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_0;
165                             break;
166                             }
167
168 #ifdef depura
169                             printf("El valor de load comando3 es %X\n",comando3);
170 #endif
171
172                             PERIFERICO_seek_set(filedesc,0);
173                             if(PERIFERICO_write( filedesc, comando3, size )!=1)
174                             {
175                                     return 0;//no se pudo escribir
176                             }
```

```
177
178                                     PERIFERICO_seek_set(filedesc,8);
179                                     dato_f= z[i*14+j];
180                                     PERIFERICO_write_f( filedesc, dato_f, size );
181
182                                     }
183                     }
184
185             //-------------------------------------------------------
186
187             //ajuste de mu
188             dato_inv= mu << 10;
189
190             //configuracion de mu
191             PERIFERICO_seek_set(filedesc,0);
192             PERIFERICO_write( filedesc, dato_inv, size );
193
194             PERIFERICO_seek_set(filedesc,0);//ubicacion del reg M
195             dato_inv=PERIFERICO_read(filedesc,dato_inv,size);
196             dato_inv= (dato_inv & PERIFERICO_MU );
197
198             return dato_inv;//devuelve el valor de mu
199     }
```

Script J.9: Función PERIFÉRICO_load

J.1.10 función PERIFÉRICO_extract

```
1   int PERIFERICO_extract(size_t filedesc, float z[224], int m, int j_o[7]){
2           int size=4;
3           float dato_f=0;
4           int i,j;
5           int comando1, comando2, comando3;
6
7           //lectura del valor de j0_o
8           PERIFERICO_seek_set(filedesc,40);
9           dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
10  #ifdef depura
11          printf("El valor de j0_o es %f\n",dato_f);
12  #endif
13          j_o[0]=dato_f;
14
15          //lectura del valor de j1_o
16          PERIFERICO_seek_set(filedesc,44);
17          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
18  #ifdef depura
19          printf("El valor de j1_o es %f\n",dato_f);
20  #endif
21          j_o[1]=dato_f;
22
23          //lectura del valor de j2_o
```

```
24          PERIFERICO_seek_set(filedesc,48);
25          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
26  #ifdef depura
27          printf("El valor de j2_o es %f\n",dato_f);
28  #endif
29          j_o[2]=dato_f;
30
31          //lectura del valor de j3_o
32          PERIFERICO_seek_set(filedesc,52);
33          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
34  #ifdef depura
35          printf("El valor de j3_o es %f\n",dato_f);
36  #endif
37          j_o[3]=dato_f;
38
39          //lectura del valor de j4_o
40          PERIFERICO_seek_set(filedesc,56);
41          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
42  #ifdef depura
43          printf("El valor de j4_o es %f\n",dato_f);
44  #endif
45          j_o[4]=dato_f;
46
47          //lectura del valor de j5_o
48          PERIFERICO_seek_set(filedesc,60);
49          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
50  #ifdef depura
51          printf("El valor de j5_o es %f\n",dato_f);
52  #endif
53          j_o[5]=dato_f;
54
55          //lectura del valor de j6_o
56          PERIFERICO_seek_set(filedesc,64);
57          dato_f=PERIFERICO_read_f(filedesc,dato_f,size);
58  #ifdef depura
59          printf("El valor de j6_o es %f\n",dato_f);
60  #endif
61          j_o[6]=dato_f;
62
63
64          comando1 = (PERIFERICO_WRITE_ENAB_DRIV_OFF | PERIFERICO_RAM_ENAB_DRIV_ON );
65  #ifdef depura
66          printf("El valor de extract comando1 es %X\n",comando1);
67  #endif
68          for (i=0;i<m;i++){
69                  switch(i){
70                  case 0:
71                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_0 );
72                  break;
73                  case 1:
74                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_1 );
75                  break;
76                  case 2:
```

```
77                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_2 );
78                  break;
79                  case 3:
80                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_3 );
81                  break;
82                  case 4:
83                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_4 );
84                  break;
85                  case 5:
86                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_5 );
87                  break;
88                  case 6:
89                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_6 );
90                  break;
91                  case 7:
92                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_7 );
93                  break;
94                  case 8:
95                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_8 );
96                  break;
97                  case 9:
98                  comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_9 );
99                  break;
100                 case 10:
101                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_10 );
102                 break;
103                 case 11:
104                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_11 );
105                 break;
106                 case 12:
107                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_12 );
108                 break;
109                 case 13:
110                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_13 );
111                 break;
112                 case 14:
113                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_14 );
114                 break;
115                 case 15:
116                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_15 );
117                 break;
118                 default:
119                 comando2 = (comando1 | PERIFERICO_ADDRESS_DRIVER_0 );
120                 }
121 #ifdef depura
122                 printf("El valor de extract comando2 es %X\n",comando2);
123 #endif
124                 for (j=0;j<14;j++){
125                                 switch(j){
126                                 case 0:
127                                 comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_0;
128                                 break;
129                                 case 1:
```

```
130                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_1;
131                             break;
132                             case 2:
133                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_2;
134                             break;
135                             case 3:
136                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_3;
137                             break;
138                             case 4:
139                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_4;
140                             break;
141                             case 5:
142                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_5;
143                             break;
144                             case 6:
145                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_6;
146                             break;
147                             case 7:
148                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_7;
149                             break;
150                             case 8:
151                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_8;
152                             break;
153                             case 9:
154                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_9;
155                             break;
156                             case 10:
157                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_10;
158                             break;
159                             case 11:
160                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_11;
161                             break;
162                             case 12:
163                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_12;
164                             break;
165                             case 13:
166                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_13;
167                             break;
168                             default:
169                             comando3 = comando2 | PERIFERICO_BLOQUE_RAM_SEL_0;
170                             break;
171                             }
172 #ifdef depura
173                             printf("El valor de extract comando3 es %X\n",comando3);
174 #endif
175                             PERIFERICO_seek_set(filedesc,0);
176                             if(PERIFERICO_write( filedesc, comando3, size )!=1)
177                             {
178                                     return 0;//no se pudo escribir
179                             }
180                             PERIFERICO_seek_set(filedesc,4);
181                             dato_f=PERIFERICO_read_f( filedesc, dato_f, size );
182                             z[i*14+j]=dato_f;
```

```
183                     }
184             }
185             return 1;
186 }
```

Script J.10: Función PERIFÉRICO_extract

J.1.11 función PERIFÉRICO_write

```
1  int PERIFERICO_write( size_t filedesc, int comando, int size ){
2          if(write(filedesc,&comando,size)<0)
3          {
4                  write(2,"There was an error writing the peripheral\n",41);
5                  return 0;
6          }
7          return 1; //si la escritura fue correcta
8  }
```

Script J.11: Función PERIFÉRICO_write

J.1.12 función PERIFÉRICO_write_f

```
1  int PERIFERICO_write_f( size_t filedesc, float comando, int size ){
2          if(write(filedesc,&comando,size)<0)
3          {
4                  write(2,"There was an error writing the peripheral\n",41);
5                  return 0;
6          }
7          return 1; //si la escritura fue correcta
8  }
```

Script J.12: Función PERIFÉRICO_write_f

J.1.13 función PERIFÉRICO_read

```
1  int PERIFERICO_read( size_t filedesc, int buf, int size ){
2          if(read(filedesc,&buf,size)<0)
3          {
4                  write(2,"There was an error reading the peripheral\n",41);
5                  return 0;
6          }
7          return buf; //si la escritura fue correcta
8  }
```

Script J.13: Función PERIFÉRICO_read

J.1.14 función PERIFÉRICO_read_f

```
float PERIFERICO_read_f( size_t filedesc, float buf, int size ){
        if(read(filedesc,&buf,size)<0)
        {
                write(2,"There was an error reading the peripheral\n",41);
                return -1;
        }
        return buf; //si la escritura fue correcta
}
```

Script J.14: Función PERIFÉRICO_read_f

J.1.15 Programa de ejemplo que usa las funciones .c

```

#include <unistd.h>
#include <fcntl.h>
#include <asm/ioctl.h>
#include <stdio.h>
/***********Librerías que facilitan la interacción con el periférico***********/

#include"/compartido/comun/home/sergio/driverprocesador/PERIFERICO/periferico_lib.h"
#include"/compartido/comun/home/sergio/driverprocesador/PERIFERICO/periferico_lib.c"

int main()
{
        float z[224];
        float z_o[224];
        int m=16;//valores menores a 16.
        int mu=3;
        int j0=10;
        int j1=5;
        int j2=8;
        int j3=16;
        int j4=4;
        int j5=32;
        int j6=20;
        int j[7];
        int retorno,i;

                for(i=0;i<m*14;i++){
        z[i]=(i+1)*100;
        }

        system("echo Inicio del PERIFERICO_principal >> tiempo_periferico_principal");
        system("date +%F_%H:%M:%S:%N >> tiempo_periferico_principal");
        retorno=PERIFERICO_principal(z,z_o,m,mu,j0,j1,j2,j3,j4,j5,j6,j);
        system("echo Inicio del PERIFERICO_principal >> tiempo_periferico_principal");
        system("date +%F_%H:%M:%S:%N >> tiempo_periferico_principal");

```

```
37          return 0;
38  }
```

Script J.15: programa_usa_ioctl.c

J.1.16 Makefile

```
1
2  programa_usa_ioctl: programa_usa_ioctl.o
3          gcc programa_usa_ioctl.o -o programa_usa_ioctl
4  programa_usa_ioctl.o: programa_usa_ioctl.c
5          gcc -c programa_usa_ioctl.c
6  clean:
7          rm *.o
8          rm programa_usa_ioctl
```

Script J.16: Makefile que compila el programa principal

J.1.17 función Pfafft.c modificada

```
1
2  static int kfax[] = { 16,13,11,9,8,7,5,4,3,2 };
3          register float *z=(float*)cz;
4          register int j00,j01,j2,j3,j4,j5,j6,j7,j8,j9,j10,j11,j12,j13,j14,j15,jt;
5          int nleft,jfax,ifac,jfac,jinc,jmax,ndiv,m,mm=0,mu=0,l;
6          float t1r,t1i,t2r,t2i,t3r,t3i,t4r,t4i,t5r,t5i,
7                  t6r,t6i,t7r,t7i,t8r,t8i,t9r,t9i,t10r,t10i,
8                  t11r,t11i,t12r,t12i,t13r,t13i,t14r,t14i,t15r,t15i,
9                  t16r,t16i,t17r,t17i,t18r,t18i,t19r,t19i,t20r,t20i,
10                 t21r,t21i,t22r,t22i,t23r,t23i,t24r,t24i,t25r,t25i,
11                 t26r,t26i,t27r,t27i,t28r,t28i,t29r,t29i,t30r,t30i,
12                 t31r,t31i,t32r,t32i,t33r,t33i,t34r,t34i,t35r,t35i,
13                 t36r,t36i,t37r,t37i,t38r,t38i,t39r,t39i,t40r,t40i,
14                 t41r,t41i,t42r,t42i,
15                 y1r,y1i,y2r,y2i,y3r,y3i,y4r,y4i,y5r,y5i,
16                 y6r,y6i,y7r,y7i,y8r,y8i,y9r,y9i,y10r,y10i,
17                 y11r,y11i,y12r,y12i,y13r,y13i,y14r,y14i,y15r,y15i,
18                 c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
19
20         #ifdef sergio_proc
21         float zz[224];/*nuevo*/
22         float z_o[224];/*nuevo*/
23         int j[7];/*nuevo*/
24         int retorno;/*nuevo*/
25         int m_16=0;/*nuevo*/
26         int v00[17],v00_1[17],v01[17],v01_1[17],v02[17],v02_1[17],v03[17],v03_1[17],v04[17],
27         v04_1[17],v05[17],v05_1[17],v06[17],v06_1[17];/*nuevo*/
28         #endif
29
```

```
30                  /* if factor is 7 */
31                  if (ifac==7) {
32          #ifdef sergio_original
33                          if (mu==1) {
34                                  c1 = P623;
35                                  c2 = -P222;
36                                  c3 = -P900;
37                                  c4 = P781;
38                                  c5 = P974;
39                                  c6 = P433;
40                          } else if (mu==2) {
41                                  c1 = -P222;
42                                  c2 = -P900;
43                                  c3 = P623;
44                                  c4 = P974;
45                                  c5 = -P433;
46                                  c6 = -P781;
47                          } else if (mu==3) {
48                                  c1 = -P900;
49                                  c2 = P623;
50                                  c3 = -P222;
51                                  c4 = P433;
52                                  c5 = -P781;
53                                  c6 = P974;
54                          } else if (mu==4) {
55                                  c1 = -P900;
56                                  c2 = P623;
57                                  c3 = -P222;
58                                  c4 = -P433;
59                                  c5 = P781;
60                                  c6 = -P974;
61                          } else if (mu==5) {
62                                  c1 = -P222;
63                                  c2 = -P900;
64                                  c3 = P623;
65                                  c4 = -P974;
66                                  c5 = P433;
67                                  c6 = P781;
68                          } else {
69                                  c1 = P623;
70                                  c2 = -P222;
71                                  c3 = -P900;
72                                  c4 = -P781;
73                                  c5 = -P974;
74                                  c6 = -P433;
75                          }
76                          for (l=0; l<m; l++) {
77                                  t1r = z[j01]+z[j6];
78                                  t1i = z[j01+1]+z[j6+1];
79                                  t2r = z[j2]+z[j5];
80                                  t2i = z[j2+1]+z[j5+1];
81                                  t3r = z[j3]+z[j4];
82                                  t3i = z[j3+1]+z[j4+1];
```

```
83                                  t4r = z[j01]-z[j6];
84                                  t4i = z[j01+1]-z[j6+1];
85                                  t5r = z[j2]-z[j5];
86                                  t5i = z[j2+1]-z[j5+1];
87                                  t6r = z[j3]-z[j4];
88                                  t6i = z[j3+1]-z[j4+1];
89                                  t7r = z[j00]-0.5*t3r;
90                                  t7i = z[j00+1]-0.5*t3i;
91                                  t8r = t1r-t3r;
92                                  t8i = t1i-t3i;
93                                  t9r = t2r-t3r;
94                                  t9i = t2i-t3i;
95                                  y1r = t7r+c1*t8r+c2*t9r;
96                                  y1i = t7i+c1*t8i+c2*t9i;
97                                  y2r = t7r+c2*t8r+c3*t9r;
98                                  y2i = t7i+c2*t8i+c3*t9i;
99                                  y3r = t7r+c3*t8r+c1*t9r;
100                                 y3i = t7i+c3*t8i+c1*t9i;
101                                 y4r = c6*t4r-c4*t5r+c5*t6r;
102                                 y4i = c6*t4i-c4*t5i+c5*t6i;
103                                 y5r = c5*t4r-c6*t5r-c4*t6r;
104                                 y5i = c5*t4i-c6*t5i-c4*t6i;
105                                 y6r = c4*t4r+c5*t5r+c6*t6r;
106                                 y6i = c4*t4i+c5*t5i+c6*t6i;
107                                 z[j00] = z[j00]+t1r+t2r+t3r;
108                                 z[j00+1] = z[j00+1]+t1i+t2i+t3i;
109                                 z[j01] = y1r-y6i;
110                                 z[j01+1] = y1i+y6r;
111                                 z[j2] = y2r-y5i;
112                                 z[j2+1] = y2i+y5r;
113                                 z[j3] = y3r-y4i;
114                                 z[j3+1] = y3i+y4r;
115                                 z[j4] = y3r+y4i;
116                                 z[j4+1] = y3i-y4r;
117                                 z[j5] = y2r+y5i;
118                                 z[j5+1] = y2i-y5r;
119                                 z[j6] = y1r+y6i;
120                                 z[j6+1] = y1i-y6r;
121                                 jt = j6+2;
122                                 j6 = j5+2;
123                                 j5 = j4+2;
124                                 j4 = j3+2;
125                                 j3 = j2+2;
126                                 j2 = j01+2;
127                                 j01 = j00+2;
128                                 j00 = jt;
129                         }
130         #endif
131         #ifdef sergio_proc
132         while(m>0){
133
134                 v00[0]=j00;
135                 v00_1[0]=j00+1;
```

```
136                 v01[0]=j01;
137                 v01_1[0]=j01+1;
138                 v02[0]=j2;
139                 v02_1[0]=j2+1;
140                 v03[0]=j3;
141                 v03_1[0]=j3+1;
142                 v04[0]=j4;
143                 v04_1[0]=j4+1;
144                 v05[0]=j5;
145                 v05_1[0]=j5+1;
146                 v06[0]=j6;
147                 v06_1[0]=j6+1;
148
149                 while((m>0) && (m_16<16)){
150                         l=m_16;
151                         zz[l*14]=z[j00];
152                         zz[l*14+1]=z[j00+1];
153                         zz[l*14+2]=z[j01]   ;
154                         zz[l*14+3]=z[j01+1];
155                         zz[l*14+4]=z[j2]    ;
156                         zz[l*14+5]=z[j2+1];
157                         zz[l*14+6]=z[j3]  ;
158                         zz[l*14+7]=z[j3+1];
159                         zz[l*14+8]=z[j4]    ;
160                         zz[l*14+9]=z[j4+1];
161                         zz[l*14+10]=z[j5]   ;
162                         zz[l*14+11]=z[j5+1];
163                         zz[l*14+12]=z[j6]   ;
164                         zz[l*14+13]=z[j6+1];
165
166                         jt=j6+2;
167                         j6=j5+2;
168                         j5=j4+2;
169                         j4=j3+2;
170                         j3=j2+2;
171                         j2=j01+2;
172                         j01=j00+2;
173                         j00=jt;
174
175                         v00[l+1]=j00;
176                         v00_1[l+1]=j00+1;
177                         v01[l+1]=j01;
178                         v01_1[l+1]=j01+1;
179                         v02[l+1]=j2;
180                         v02_1[l+1]=j2+1;
181                         v03[l+1]=j3;
182                         v03_1[l+1]=j3+1;
183                         v04[l+1]=j4;
184                         v04_1[l+1]=j4+1;
185                         v05[l+1]=j5;
186                         v05_1[l+1]=j5+1;
187                         v06[l+1]=j6;
188                         v06_1[l+1]=j6+1;
```

```
189                         m=m-1;
190                         m_16=m_16+1;
191                 }
192
193                 retorno=PERIFERICO_principal(zz,z_o,m_16,mu,v00[0],v01[0],v02[0],v03[0],v04[0],
194                 v05[0],v06[0],j);
195                 for (l=0;l<m_16;l++){
196                         z[v00[l]]   = z_o[l*14];
197                         z[v00_1[l]]= z_o[l*14+1];
198                         z[v01[l]]   = z_o[l*14+2];
199                         z[v01_1[l]]= z_o[l*14+3];
200                         z[v02[l]]   = z_o[l*14+4];
201                         z[v02_1[l]]= z_o[l*14+5];
202                         z[v03[l]]   = z_o[l*14+6];
203                         z[v03_1[l]]= z_o[l*14+7];
204                         z[v04[l]]   = z_o[l*14+8];
205                         z[v04_1[l]]= z_o[l*14+9];
206                         z[v05[l]]   = z_o[l*14+10];
207                         z[v05_1[l]]= z_o[l*14+11];
208                         z[v06[l]]   = z_o[l*14+12];
209                         z[v06_1[l]]= z_o[l*14+13];
210                 }
211                 m_16=0;
212         }
213         #endif
214         #ifdef sergio_debug
215         system("echo Fin del calculo del factor 7 >> calculo_ifac_tarjeta");
216         system("date +%F_%H:%M:%S:%N >> calculo_ifac_tarjeta");
217         #endif
218                         continue;
219                 }
```

Script J.17: Solo muestra las secciones modificadas de Pfafft.c