DESARROLLO DE MATERIAL DIDÁCTICO PARA EL ÁREA DE SISTEMAS DIGITALES BASADO EN LA PLATAFORMA SIE. ASIGNATURA: PROCESADORES.

JUAN MIGUEL FORERO MARTÍNEZ
YAMID JAVIER RODRIGUEZ MOLINA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA
Y DE TELECOMUNICACIONES
BUCARAMANGA
2012

DESARROLLO DE MATERIAL DIDÁCTICO PARA EL ÁREA DE SISTEMAS DIGITALES BASADO EN LA PLATAFORMA SIE. ASIGNATURA: PROCESADORES.

JUAN MIGUEL FORERO MARTÍNEZ
YAMID JAVIER RODRIGUEZ MOLINA

Trabajo de grado para optar al título de Ingeniero Electrónico

Director:
MSc. Jorge H. Ramón Suarez

Codirectores:
MSc. William A. Salamanca B.
ING. Carlos A. Angulo J.
MSc. Carlos A. Fajardo A.
MSc. Sergio A. Abreo C.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2012

AGRADECIMIENTOS

A todas las personas que participaron e hicieron posible este proyecto. A nuestro director MSc. Jorge H. Ramón Suarez por la oportunidad de hacer parte de esta gran experiencia pedagogica y didactica , a todos nuestros codirectores, especialmente al profesor Msc. William A. Salamanca B por compartir su atención, enseñansas, tiempo y conocimientos .A nuestros compañeros de los proyectos asociados por todos los valiosos y enriquecedores aportes y sugerencias. Y a todos nuestros amigos por el apoyo incondicional en todos estos años que compartimos.

RESUMEN

TÍTULO: DESARROLLO DE MATERIAL DIDÁCTICO PARA EL ÁREA DE SISTEMAS DIGITALES BASADO EN LA PLATAFORMA SIE. ASIGNATURA: PROCESADORES[*]

AUTORES: JUAN MIGUEL FORERO MARTINÉZ
YAMID JAVIER RODRIGUEZ MOLINA[**]

PALABRAS CLAVES: Procesadores, Prácticas de Laboratorio, Plataforma SIE, Material Didáctico, Diseño digital avanzado.

CONTENIDO:
La Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones (E3T) de la Universidad Industrial de Santander a través de la asignaturas Procesadores, tomó la decisión de implementar la plataforma SIE como plataforma de trabajo, con el fin de afianzar de manera práctica conocimientos en el área de los sistemas digitales. La realización de este proyecto de grado define un conjunto de experiencias de laboratorio que sirven de apoyo a la asignatura de procesadores, permitiendo el aprovechamiento de un conjunto de herramientas en el área de los sistemas digitales avanzados, mejorando así la calidad de los profesionales.
Este informe recopila los elementos empleados, las actividades desarrolladas y los principales resultados logrados teniendo en cuenta las necesidades pedagógicas que plantea la asignatura, es decir: las prácticas que se realicen y los proyectos que se propongan deben servir para medir y evaluar el desempeño de los estudiantes.
En definitiva se realizó el diseño, montaje y desarrollo de 6 prácticas de laboratorio, un proyecto final y un manual de usuario intermedio de la plataforma SIE basado en la asignatura, las cuales tienen como objetivo que el estudiante pueda afianzar los conocimientos acerca de los temas tratados, resaltando aspectos relacionados con los recursos brindados por la plataforma SIE. Esto incluye la utilización de Leds, pulsadores y demás periféricos empleados en las experiencias de laboratorio. La etapa final del proyecto de grado constituye la validación y verificación de cada una de las prácticas, lo cual se realizará mediante un grupo de estudiantes con el fin de comprobar que cumplan con los objetivos que se plantearon.

[*] Trabajo de grado

[**] **Facultad:** Ingenierías Físico Mecánicas, **Escuela:** Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. **Grupo de Investigación:** CPS. **Director:** MSc. Jorge H. Ramón Suarez. **Codirectores:** MSc. William A. Salamanca B., ING. Carlos A. Angulo J., MSc. Carlos A. Fajardo A., MSc. Sergio A. Abreo C.

ABSTRACT

TITLE: DEVELOPMENT OF EDUCATIONAL MATERIAL FOR THE AREA OF DIGITAL SYSTEMS BASED ON THE PLATFORM SIE. SUBJECT: PROCESSORS*

AUTHORS: JUAN MIGUEL FORERO MARTINÉZ
YAMID JAVIER RODRIGUEZ MOLINA**

KEY WORDS: Processors, Laboratories, Platform SIE, Didactic Material, Advanced Digital Design .

CONTENTS: The School of Electrical , Electronics and Telecommunications Engineering (E3T) of the Industrial University of Santander, through the subjects Processors, took the decision to implement the SIE as a work platform in order to strengthen the knowledge in a practical way for the area of digital systems. With this project we define a set of laboratory experiments that support the processors course. Giving tools to the student in the area of advanced digital systems, thereby improving the skills.

This report lists the elements used, the activities and the main results achieved, taking into account the educational needs arising from the subject : Exercises held and proposed projects must serve to measure and evaluate the student performance.

Also this project gives the design, installation and development of 6 laboratory sesions, a final project and a user manual performed for the SIE intermediate platform based on the subject, which are intended to enable the student to consolidate the knowledge of the topics covered in the Processors course, highlighting aspects of the resources provided by the platform SIE. This includes the use of the LEDs, switches and other peripherals . The final stage of the project is the validation and verification of each of the practices, performed by a group of students to ensure the proposed objectives.

*Degree work

Faculty: Physical Mechanical Engineering , **school: Electrical, Electronic and telecommunications. **Investigación group:** CPS. **Director:** MSc. Jorge H. Ramón Suarez. **Codirectors:** MSc. William A. Salamanca B., ING. Carlos A. Angulo J., MSc. Carlos A. Fajardo A., MSc. Sergio A. Abreo C.

CONTENIDO

pág

LISTA DE FIGURAS

pág

LISTA DE TABLAS

pág

LISTA DE ANEXOS

pág

INTRODUCCIÓN

En las sociedades modernas las nuevas tecnologías en el área de los sistemas digitales han tenido un crecimiento preponderante. La abundancia de herramientas digitales ha influido notablemente en las destrezas tecnológicas de las personas. Actualmente el desarrollo de estas tecnologías no solo está transformado los sistemas digitales ya conocidos, sino que está cambiando la manera de cómo trabajar sobre ellos. De esta forma es imperativo que los centros educativos como las universidades revisen sus paradigmas, estructuras y funcionamiento a la luz de las posibilidades que ofrecen estos grandes avances tecnológicos.

Hasta hace poco se consideraba que los conocimientos adquiridos por un joven en su etapa de estudiante le funcionarían para toda su vida, sin embargo la realidad ha cambiado. El avance de la ciencia y la tecnología es continuo, las personas desempeñan trabajos que no existían cuando nacieron. Caracteristicas, requerimientos y necesidades del mundo actual deben ser tenidas en cuenta por las instituciones educativas para el mejoramiento de la formación de los individuos, para luego transformarse en una pieza fundamental en el desarrollo profesional de los estudiantes y preparandolos para enfrentar los retos que le propone el acelerado progreso en cuanto a tecnologías se refiere.

El desarrollo de este proyecto cobra gran importancia desde el punto de vista de las aplicaciones y herramientas que ofrece la plaforma SIE a los estudiantes de ingeniería electrónica de la Universidad Industrial de Santander para incursionar de manera práctica en el área de los sistemas digitales en especial en la asignatura “ procesadores”.

La escuela de ingenierías eléctrica, electrónica y de telecomunicaciones, ofrece en su programa académico de ingeniería electrónica la asignatura “procesadores”, la cual otorga un conjunto de herramientas al estudiante en el área de los sistemas digitales avanzados, para complementar y afianzar conocimientos. De igual forma permiten que la universidad se mantenga a la vanguardia en nuevas tecnologías, mejorando así la calidad de sus profesionales.

Este informe recopila los elementos empleados en las práctica, las actividades desa-

rrolladas y los principales resultados logrados en el proyecto de grado. Inicialmente tenemos las especificaciones del trabajo de grado, incluidos el objetivo general y los objetivos específicos planteados desde el inicio. El segundo capítulo desarrolla los principales conceptos teóricos en que se fundamenta este trabajo de grado, se plantea el contexto académico de la asignatura procesadores, una breve introducción a Linux y generalidades sobre la plataforma SIE. La información pertinente sobre el desarrollo de las prácticas propuestas, especificando el contenido estructural y los objetivos planteados de cada una de ellas se explican en el tercer capítulo. Todos los aspectos relacionados con el diseño y construcción de la propuesta de proyecto final de la asignatura "procesadores", donde se aplicarán todos los conocimientos adquiridos durante el trayecto del curso se expondrán en el cuarto capítulo. El quinto capítulo consta de la metodología utilizada, incluyendo la validación y verificación de las actividades propuestas. Con el firme propósito de afianzar conocimientos y destrezas sobre el manejo de la tarjeta SIE y herramientas de la asignatura "procesadores", se explica en el sexto capítulo el manual de usuario intermedio de la plataforma SIE. El séptimo y octavo capítulo resumen las principales conclusiones y recomendaciones.

1. ESPECIFICACIONES DEL PROYECTO

Este informe de resultados se desarrolla con el fin de recopilar el proceso seguido y los principales logros alcanzados en desarrollo del trabajo de grado: "Desarrollo de material didáctico para el área de sistemas digitales basados en la plataforma SIE. Asignatura: Procesadores".

1.1. OBJETIVO GENERAL

Desarrollar material didáctico basado en la plataforma SIE para el área de sistemas digitales, en la asignatura procesadores.

1.2. OBJETIVOS ESPECÍFICOS

- Elaborar un manual de usuario intermedio de la plataforma SIE orientando su aplicación a la asignatura Procesadores.
- Diseñar e implementar seis prácticas de laboratorio para la asignatura Procesadores abarcando: Glitch, metaestabilidad e implementación de FSM (1); Datapath y FSM (1); Metodología para el diseño de procesadores específicos (2) y Pipeline (2).
- Diseñar e implementar dos proyectos finales para la asignatura Procesadores.
- Verificación y validación experimental de las prácticas y proyectos propuestos.

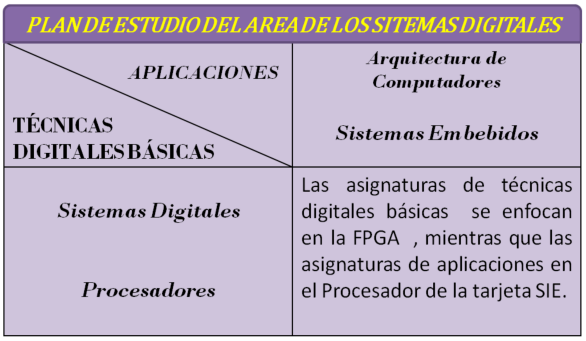
2. FUNDAMENTOS TEÓRICOS

En esta sección se presentarán todos los apartados relacionados con el desarrollo del trabajo de grado y las herramientas importantes en la correcta conducción del tema, como son: El contexto de la asignatura procesadores, introducción al sistema operativo Linux y la plataforma SIE.

2.1. CONTEXTO DE LA ASIGNATURA PROCESADORES

Hoy en día los programas académicos de las universidades usan herramientas de alto nivel como las tarjetas de programación para el desarrollo de los cursos relacionados con el area de los sistemas digitales, lo cual hace que los profesionales solo adquieran habilidades en el diseño y no en la construcción de hardware. En ausencia de estas habilidades la consecuencia inmediata es la poca generación de empleo local a personas con un alto nivel de formación, pero con falencias en la construcción de diseños Hardware[6]. En consecuencia la Universidad Industrial de Santander dentro del plan de estudios de Ingeniería Electrónica orientado a los sistemas digitales, distingue una primera parte correspondiente a las técnicas digitales básicas y una segunda parte que tiene que ver con aplicaciones específicas(véase la figura 1)

Figura 1. Plan de estudios del área de sistemas digitales. Los autores



El grupo de docentes que tiene a cargo estas asignaturas ha decidido acompañar un grupo de proyectos de grado para desarrollar material de apoyo para las mismas. Con este conjunto de proyectos se busca integrar y coordinar estos cursos, haciendo énfasis en una metodología de diseño de sistemas embebidos centrada en el concepto de Codiseño, y acoger una plataforma común de laboratorio (SIE) para los cuatro cursos.

Debido a esto se definen un conjunto de experiencias de laboratorio para la asignatura procesadores del área de sistemas digitales, basadas en la plataforma SIE. En términos amplios, el objetivo de la asignatura de procesadores es:

- PROCESADORES: Concebir, definir las especificaciones, modelar y diseñar un Sistema Digital utilizando la metodología de diseño de Sistemas Embebidos y realizar su implementación óptima utilizando herramientas de *Hardware* .

2.2. INTRODUCCIÓN A LINUX

Para empezar se realiza una visión general de cómo Linux se convirtió en el sistema operativo actual, teniendo en cuenta sus ventajas y desventajas [7].

Linux es el acontecimiento más importante en lo que a software gratuito se refiere. Se ha convertido en el sistema operativo para los negocios, la educación y el aprovechamiento personal. Universidades de todo el mundo usan Linux para dar cursos de programación y diseño de sistemas operativos [5].

Linux es un clónico del sistema operativo UNIX, distribuida gratuitamente en los términos de la Licencia GNU. El sistema operativo Unix nace a finales de los años 60 en los laboratorios Bell AT&T. Ken Thompson, insatisfecho con el sistema operativo que manejaba en ese entonces toma la decisión de construir su propio Sistema Operativo en lenguaje ensamblador, pero luego lo reescribe en un lenguaje de programación llamado B. Después de un tiempo Dennis Ritchie junto a Ken Thompson, traducen Unix a lenguaje C [5].

Dada la imposibilidad de comercializarlo, AT&T decide distribuirlo con fines generosos

a universidades a cambio de un pago simbólico, lo cual tiene como consecuencia la rápida extensión y uso en el mundo científico, pero también la creación de un gran número de versiones, debido a que no había nadie quien dirigiera su desarrollo y evolución. De estas versiones surge Linux, inicialmente desarrollado por Linus Torvalds en la universidad de Helsinki, en Finlandia. Inspirándose en un sistema operativo conocido como Minix, un pequeño UNIX desarrollado por Andy Tanenbaum. El 5 de octubre de 1991, Linus anunció la primera versión de Linux, la 0.02 el cual ejecutaba bash (el Shell de GNU) y gcc (el compilador de C de GNU), pero no hacía mucho más. Esta primera versión carecía de soporte de usuarios, distribuciones y documentación [5].

Hoy en día Linux contiene una gran variedad de características, a continuación se mencionan algunas de ellas [5].

- **Multitarea:** Consiste en que el sistema operativo puede atender múltiples tareas al mismo tiempo. Ya sea desde un único usuario o varios usuarios.
- **Multiusuario:** el sistema operativo atiende a un único usuario o varios usuarios desde un mismo computador.
- **De planificación mixta:** Consiste en la distribución de tiempo de la CPU entre los diversos procesos, previendo que varios procesos quieran utilizar el microprocesador.
- **Con implementaciones de memoria virtual:** Hace uso de espacio de almacenamiento en disco como si fuera memoria adicional, es decir la memoria efectiva puede ser mayor que la real.
- **Librerías compartidas de carga dinámica y librerías estáticas:** Las librerías DLL[1] permiten al programador de aplicaciones sustituir funciones ya definidas con su propio código.
- **Todos los códigos fuente están disponibles:** incluyendo el núcleo completo y todos los drivers, la herramienta de desarrollo y todos los programas de usuario. También todo ello se puede distribuir libremente.
- **Libre desarrollo:** Cualquier persona puede acceder a Linux y desarrollar nuevos módulos o cambiarlos a su antojo.

[1]DLL: Dinamically Linked Library (acronimo de windows)

Una de las características de este sistema operativo, es que por defecto el trabajo de Linux no se realiza de una forma gráfica, sino introduciendo comandos de forma manual.
Por ello dispone de varios programas que se encargan de interpretar los comandos que introduce el usuario y realiza las acciones correspondientes. Estos programas son denominados Shell [5].

En la Tabla 1 se explican las funciones de los comandos más utilizados en el Sistema Operativo Linux.

Tabla 1. Listado de comandos de Linux.Tomada de [15]

FUNCIÓN	**DESCRIPCIÓN**
ls	Lista de archivos y directorios.
mkdir	Crear un directorio.
cd directory	Cambiar al directorio llamado.
cd	Cambio a la casa de la guía.
cd ..	Cambiar al directorio padre.
pwd	Mostrar la ruta del directorio actual.
cp file1 file2	Copia archivo1 archivo2 y lo llaman.
mv file1 file2	Mover o cambiar el nombre de archivo1 a archivo2.
rm file	Eliminar un archivo.
rmdir directory	Eliminar un directorio.
cat file	Mostrar un archivo.
chmod [options] file	Los derechos de cambiar el acceso de archivo con el nombre.
make	Ayuda en el desarrollo de programas grandes, manteniendo un registro de qué partes de la totalidad del programa han cambiado.la compilación sólo de aquellas partes del programa que han cambiado desde la última compilación.

2.3. PLATAFORMA SIE

La plataforma SIE fue creada como herramienta versátil y de bajo costo para diseñadores y desarrolladores de hardware, constituyéndose en herramienta de implementación de los cursos del área de sistemas digitales de universidades como la Nacional de Colombia y la Universidad Industrial de Santander[6].

El proyecto SIE se realizó a partir de otro llamado SAKC (Swiss Army Kinfe Card) el cual a su vez fue creado de otro proyecto Ben NanoNote,una ventaja ya que todo el desarrollo realizado para Ben NanoNote es aplicable a SIE. Esto garantiza el mantenimiento, actualización y disponibilidad del software necesario para el desarrollo de aplicaciones. La plataforma SIE (antes conocida como SACK), posee una combinación de procesador MIPS[2] y FPGA, constituyendose en una sencilla y versatil herramienta de desarrollo[6].

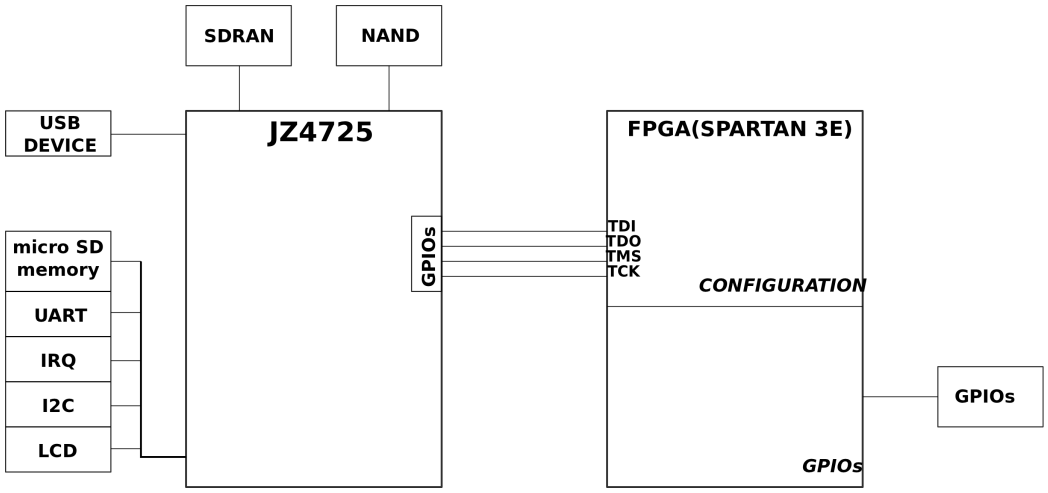
El diagrama de bloques de la plataforma SIE se observa en la Figura 2, detallando los principales componentes. Aquí vemos un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, 2 entradas y salidas de audio estéreo, 2 entradas análogas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs por sus siglas en ingles General Purpose Input/Ouput), dos canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, permitiendo la configuración de la FPGA desde el procesador (lo que elimina la necesidad de cables de programación) y otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA. El procesador es un Ingenic JZ4725 (MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 64 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux[6].

También posee un puerto USB-device que es configurado como una interface de red (usb0), proporcionando un canal de comunicación. Permitiendo la transferencia de archivos y ejecución por medio de una consola remota mediante el protocolo *ssh* y

[2]MIPS: Microprocessor without Interlocked Pipeline Stages

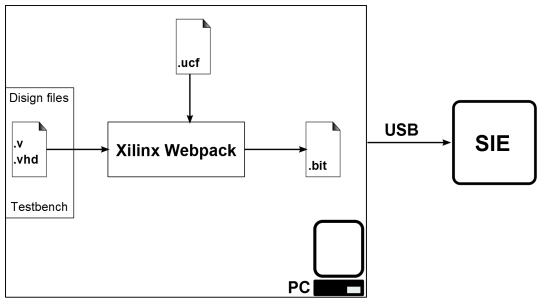
programando la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB [6].

Figura 2. Diagrama de bloques de SIE. Tomada de: [1]



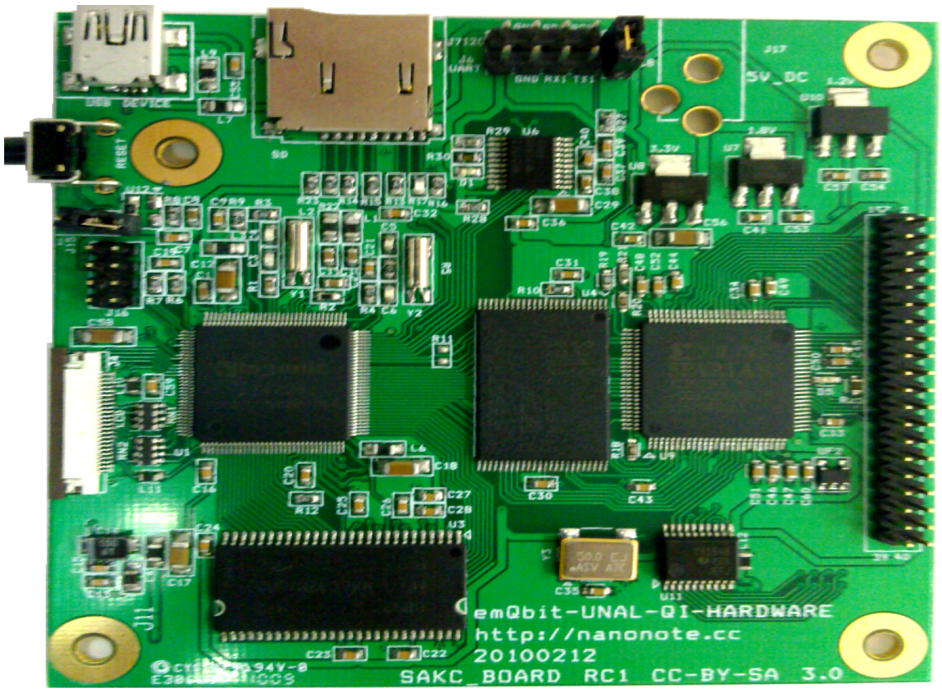
SIE posee aplicaciones y librerías que pueden ser implementadas en un computador normal, utilizando para el componente hardware las herramientas gratuitas suministradas por *Xilinx*. Por consiguiente es posible la aplicación práctica de un conjunto de conocimientos y habilidades enfocadas al desarrollo de estrategias de codiseño haciendo posible la implementación de sistemas que emplearán hardware y software para realizar una función específica. La Figura 3 muestra el diagrama de flujo y las herramientas utilizadas para las tareas de hardware.

Figura 3. Flujo de diseño hardware. Tomada de: [6] y modificada.



Continuando con la descripción de la plataforma SIE, tenemos las siguientes especificaciones físicas: 8cm de largo, 8 cm de ancho, 1.6mm de altura; su placa de circuito impreso es de dos capas, utiliza líneas de 0.2mm, vías de 0.3mm de diámetro, los componentes se encuentran en una sola capa y son TQFP[3] o SMD[4], no posee componentes BGA[5] o QFN[6] lo que facilita el montaje manual. El costo de fabricación es aproximadamente 75 usd para 50 unidades (aproximadamente $ 135000 pesos colombianos). La Figura 4 muestra la plataforma de desarrollo SIE.

Figura 4. Plataforma de desarrollo SIE. Tomada de: [1].



[3] TQFP: Thin Quad Flat Package; Encapsulado cuadrado plano.

[4]SMD: Surface Mounted Device; Dispositivo de montaje superficial.

[5]BGA:Ball Grid Array ; Posee bolitas en la parte inferior del encapsulado.

[6]QFN: Quad flat pack ; Versión más pequeña y sin pines de QFP.

3. PRÁCTICAS DE LABORATORIO

Mediante el desarrollo de este capítulo se evidencian los tópicos que se tuvieron en cuenta para las prácticas del laboratorio en la asignatura procesadores. Para el montaje y desarrollo de las prácticas se abordó la importancia conceptual de temáticas que son relevantes en el proceso de aprendizaje del estudiante que implemente las prácticas propuestas[3].

Con el objetivo de cumplir satisfactoriamente las metas trazadas en la asignatura procesadores, se hace necesario el desarrollo de prácticas propositivas, donde al estudiante se le brinde una herramienta teórica sobre la temática tratada y tenga la competencia necesaria para realizar su propio diseño digital[3]. La elaboración de las prácticas está orientada hacia la compresión del estudio de 6 temas : *Glitch*, máquinas de estado (FSM)[7], metaestabilidad, sincronizadores, segmentación y Diseño digital avanzado (Unidad de control y *datapath*).

La estructura de las prácticas de laboratorio contiene: Titulo de la temática a implementar, introducción, objetivos que se pretenden alcanzar durante el desarrollo de la práctica, marco teórico que contenga conceptos previos requeridos para el desarrollo satisfactorio de la actividad, descripción del problema que aborda el ejercicio descrito de manera que el estudiante comprenda lo que se requiere en la actividad, informe de laboratorio que evidencie los resultados obtenidos en la realización de la actividad planteada y por último preguntas de prueba que evalúen lo aprendido por parte del estudiante.

Finalmente todas las prácticas diseñadas durante el proyecto se recopilarán dentro de un texto en forma de libro, añadiéndole un manual de usuario intermedio basado en la plataforma SIE con énfasis en aspectos relacionados con la asignatura procesadores.

Adicionalmente, se desarrollarón las prácticas propuestas por parte de los autores del proyecto, para comprobar el correcto funcionamiento y garantizando que no contengan

[7]FSM: Finite State Machine

errores de desarrollo dentro de la realización de la actividad. Este material será incluido al final del informe, en el anexo del proyecto.

A continuación se describen los principales objetivos de las prácticas propuestas.

3.1. PRÁCTICA 1: IMPLEMENTACIÓN DE CIRCUITOS COMBINACIONALES Y *GLITCHES*

En la implementación de circuitos combinacionales pueden darse valores transitorios y pulsos no deseados. Estos valores transitorios se conocen como *glitches*. Dependiendo del uso que se le dé a las salidas, estos pulsos pueden ser o no causa de graves problemas en la implementación de circuitos [9].

Lo que se quiere lograr con esta práctica es la identificación de problemas comunes en los circuitos combinacionales y secuenciales, adquiriendo destrezas en la aplicación de metodologías de diseño que nos permitan corregir problemas comunes en el mal uso de señales asíncronas. Por medio de la aplicación de estrategias que permitan solucionar el problema que generan los *glitches* en un sistema digital avanzado, asi como el uso de herramientas y opciones avanzadas de síntesis que nos brinda el entorno de desarrollo de *Xilinx*.

3.2. PRÁCTICA 2: IMPLEMENTACIÓN DE MÁQUINAS DE ESTADO EN VHDL

Las máquinas de estado son uno de los medios más utilizados en la implementación de aplicaciones digitales, permitiendo definir una serie de estados y para cada uno de ellos un comportamiento distinto a las entidades que los contienen, donde podemos encontrarnos en un instante y dadas unas entradas puede hacernos transitar de un estado a otro o permitiéndonos activar acciones o salidas relacionadas con dicho estado. Por su simplicidad y rapidez en el diseño e implementación, se han convertido en una he-

rramienta muy poderosa que utiliza una manera sistemática para el diseño secuencial síncrono de circuitos dada una especificación funcional[10].

En esta práctica se busca aplicar todo el conocimiento sobre las máquinas de estados, implementando un ejemplo donde se coloque en práctica la metodología para realizar máquinas de estado de manera correcta, adquiriendo destrezas necesarias para implementarlas usando el lenguaje VHDL y corrigiendo los problemas de *glitches* presentados en la práctica anterior.

3.3. PRÁCTICA 3: VIOLACIONES DE *SETUP* Y *HOLD* "METAESTABILIDAD"

La Metaestabilidad es el conjunto de oscilaciones o indefinición del valor de la salida de un biestable durante un tiempo indeterminado, el cual se debe al incumplimiento de los tiempos de *setup* y *hold*. Debido a la aparición de este fenómeno, la salida puede comportarse de forma imprevista, tardando muchas veces más de lo normal en estabilizarse al estado correcto. En muchos casos, la metaestabilidad en los biestables se puede evitar asegurándose que los datos y las entradas de control se mantengan constantes durante un periodo de tiempo especificado antes y después del flanco de reloj. Desafortunadamente, no siempre es posible cumplir estos requisitos[11].

En esta práctica se emplearán conceptos como metaestabilidad, tiempo de *Setup* y *Hold* para comprender las causas, cuáles son sus consecuencias y como se puede evitar este fenómeno. Por medio de la implementación de un ejercicio práctico se observará y analizará cuando aparece el fenómeno de la metaestabilidad. También se utilizará la información contenida dentro del datasheet de la tarjeta SIE, para conocer los tiempos de *setup* y *hold* en beneficio de una buena implementación.

3.4. PRÁCTICA 4: DOMINIOS DE RELOJ Y SINCRONIZADORES

Como se ha analizado, los sistemas digitales síncronos constan de una serie de componentes: elementos de memoria (registros), lógica combinacional que interconecta los registros y un circuito de distribución de la señal de reloj. El funcionamiento de un sistema digital ha quedado condicionado principalmente por la característica de sincronizar las señales de datos. Dicha función es esencial para la correcta operación de este tipo de sistemas, siendo fundamental prestar mucha atención a las características de las señales de reloj y por supuesto a su red de distribución, ya que el funcionamiento y fiabilidad del sistema dependen de ambos[11].

El objetivo de esta práctica es comprender y analizar los conceptos de dominios de reloj y sincronizadores, empleando los recursos de la FPGA para administrar dominios de reloj e implementando circuitos sincronizadores, los cuales presentan varios casos para satisfacer distintos requisitos, comprendiendo los beneficios de mejora en un sistema digital y teniendo en cuenta que aunque siempre se busca un diseño completamente síncrono, no siempre es posible.

3.5. PRÁCTICA 5: PARALELISMO Y PROCESADORES SEGMENTADOS "*PIPELINE*"

En la actualidad el diseño de procesadores segmentados es una constante, habiendo sido incorporado el término "segmentación" a la "jerga" informática. El diseño de un programa paralelo tiene que considerar entre otras cosas el tipo de arquitectura sobre la cual se va a ejecutar el programa y las necesidades de tiempo y espacio que requiere la aplicación, surgiendo una pregunta muy importante a tener en cuenta ¿Cómo aumentar la velocidad del procesador? dicha velocidad de ejecución de los programa depende de muchos factores, haciendo más rápidos los circuitos con que se construyen los procesadores, claro está considerando el costo que supone una mejora y que el limite a esta velocidad lo impone el estado del arte actual de la tecnología[12].

Como objetivo general en esta práctica se buscan conocer las principales características

del paralelismo y los procesadores segmentados, comprendiendo y utilizando de manera correcta los conceptos de paralelismo espacial y paralelismo temporal. Implementando mejoras en un sistema utilizando cualquiera de los tipos de paralelismo, precisando como afectan las mejoras en el *throuhghput*o rendimiento y la latencia a un sistema propuesto.

3.6. PRÁCTICA 6: DISEÑO DIGITAL AVANZADO "UNIDAD DE CONTROL Y *DATAPATH* "

El datapath es una colección de unidades funcionales, por ejemplo ALUs o multiplicadores, donde se realiza el procesamiento de datos y las operaciones. La mayoría de los procesadores consisten en un *datapath* y una unidad de control, con una gran parte de la unidad de control dedicada a regular la interacción entre el *datapath* y la memoria. El *datapath* tiene dos tipos de puertos de entrada y salida, uno de ellos es un puerto que maneja datos y el otro es un puerto que maneja señales de control. Hace algunos años era muy difícil realizar diseños digitales complejos. Sin embargo, el vertiginoso avance de la tecnología permite realizar fácilmente diseños complejos y de bajo costo, es decir, es posible diseñar sistemas a la medida de una aplicación dada utilizando herramientas como FPGAs[13].

En esta práctica se coloca en contexto todo el conocimiento teórico para diseñar de manera apropiada un procesador, implementando un *datapath* de propósito específico por medio del lenguaje de descripción de *hardware*, aplicando algunas de las metodologías de diseño existentes en la asignatura procesadores.

4. PROYECTO FINAL DE LA ASIGNATURA PROCESADORES

Mediante este capítulo se expondrán los aspectos más importantes de la propuesta de proyecto final de la asignatura procesadores. El objetivo principal es la aplicación de las temáticas de diseño digital avanzado contemplado durante el curso, evidenciando destrezas en el manejo de conceptos primordiales en la elaboración de diseños digitales. Por tal motivo se analizaron características importantes que debería tener la actividad propuesta para cumplir con el objetivo, como por ejemplo: una idea de proyecto innovadora, aplicable en la vida real y de mediana complejidad que cumpla con los requisitos estructurales de diseño en la construcción de sistemas digitales avanzados. También que evidencie las destrezas en el manejo de temas esenciales en la asignatura de procesadores (*glitches*, *datapath*, máquinas de estado, etc.) cumpliendo con todas las competencias propositivas y argumentativas que se desarrollan en la implementación de circuitos digitales a través de la utilización de la plataforma SIE y el entorno de trabajo Xilinx: ISE WebPack.[8]

Por tal motivo se decidió implementar un sistema de tratamiento de efectos de audio, el cual consiste en el desarrollo de una interfaz para el control de un códec a través de un algoritmo en código VHDL. Este sistema utiliza un teclado ps/2 para que el usuario pueda interactuar con el circuito eligiendo el efecto que se desea escuchar por medio de la pulsación de una tecla y luego reproducirlo en un altavoz[4].

El sistema de tratamiento de efectos de audio cuenta con 11 módulos inicialmente, descrito de manera clara y concisa para que el estudiante comprenda correctamente el funcionamiento de cada uno de ellos y pueda implementar todos los diseños satisfactoriamente en la FPGA de la tarjeta SIE. Cada módulo es desarrollado en el entorno de Xilinx: ISE WebPack y estructurado mediante el lenguaje de programación VHDL, empleando herramientas de hardware como: altavoces, dispositivo códec, teclado ps/2, periféricos (swicth, pulsadores, etc.) para la correcta implementación del sistema de tratamiento de efecto de audio.

[8]Herramienta gratuita que se puede descargar en: http://www.xilinx.com/webpack/classics/wpclassic/index.htm

Teniendo en cuenta que la propuesta de proyecto final ofrece un tiempo estimado de trabajo para un semestre, se planteó con ayuda del encargado de la asignatura[9] realizar un proyecto prolongado, donde en futuros semestres se le agreguen diferentes módulos afines a un sistema de audio que funcionen como proyecto final, utilizando los módulos ya existentes como base para los diseños posteriores. Está metodología ayuda a darle continuidad a una idea de proyecto bien planteada, contribuyendo a construir a partir de un diseño base, diferentes proyectos finales para semestres futuros sin la necesidad de cambiar constantemente, al contrario aportando cada semestre una idea innovadora que mejore el proyecto inicial. Se hace necesario en algún momento cambiar la idea de proyecto pero no la metodología utilizada.

A continuación se describe la función principal de cada uno de los 11 módulos del proyecto final.

- **Interfaz Códec:** Es el módulo principal del proyecto, dónde se conectan todos los módulos implicados de manera conveniente para poder realizar la elección de los efectos por medio del teclado.
- **Interfaz PS/2 :** Es el encargado de servir de conexión entre el módulo que realiza los efectos de audio y el teclado, para que el usuario interactue eligiendo el efecto que desea escuchar pulsando una tecla determinada.
- **Unidad de control PS/2:** Este módulo tiene como finalidad configurar el teclado PS2 para ser conectado en la FPGA de la tarjeta SIE.
- **Conversor serie/paralelo:** Este módulo se encarga de recibir los bits que componen una muestra de la señal que el audio códec acaba de discretizar.
- **Conversor paralelo/serie:** Este módulo toma la muestra tratada y la descompone en bits para realizar la transmisión serie bit a bit al audio códec y que éste la transforme en una señal analógica que pueda ser reproducida por unos altavoces.
- **Generador de frecuencias:** Genera las señales necesarias, con el fin de controlar el audio códec y se realicen las conversiones análogo/digital y digital/análogo correctamente.
- **Interfaz con SRAM:** Este módulo es el encargado de realizar la comunicación entre la FPGA y la memoria SRAM.
- **Módulo distribución de efectos:** Es el módulo encargado de aplicar los distintos

[9] MSc.William A. Salamanca Becerra

efectos sobre la señal de audio de entrada.

- ▷ **Efect_Delay:** Esta entidad se encarga de almacenar las muestras de audio en la memoria RAM y tratar ésta como una cola FIFO genérica.
- ▷ **Lfo:** Este módulo implementa un oscilador triangular de baja frecuencia. Toma como entrada un tiempo mínimo y máximo de persistencia y lo modula como señal triangular. Su salida la utiliza el módulo Delay como tamaño de la cola.
- ▷ **Efect_Wah:** Este módulo es el encargado de simular el efecto Wah por medio de incrementos y decrementos cíclicos del volumen de la muestra original.

Para dar continuidad a este sistema de audio a semestres posteriores, se proponen los siguientes módulos que servirán como proyecto futuros:

- ▷ **Interfaz gráfica:** Este módulo es el encargado de construir una interfaz amable, donde se muestre un número de efectos de audio, utilizando la pantalla del computador para que el usuario interactue eligiendo alguno de ellos por medio de un click con el mouse PS/2.
- ▷ **Interfaz Mouse:** Esta entidad es la encargada de realizar la conexión entre el mouse, el computador y el módulo encargado de realizar los efectos de audio.
- ▷ **Interfaz Micrófono:** Este módulo se encarga de enviar una señal de voz al audio códec para ser tratada y poder reproducirla en unos altavoces.
- ▷ **Unidad de control Mouse:** Este módulo tiene como finalidad configurar el Mouse PS/2 para ser conectado en la FPGA de la tarjeta SIE.

En el proyecto final de la asignatura de procesadores se realizaron cuatro efectos de audio sobre la temporización de la señal (*delay, chorus, flanger, wah*), es decir sobre el tiempo de la señal de entrada. Ahora bien para el sistema de audio en semestres posteriores se plantea trabajar sobre otros tipos de efectos, los cuales se aplican sobre la frecuencia de la señal de sonido. Un ejemplo de ello es el ecualizador, el cual consiste en modificar el contenido en frecuencias de la señal que procesa. Para ello, cambia las amplitudes de sus coeficientes de Fourier, lo que se traduce en diferentes volúmenes para cada frecuencia. Con esto se puede variar de forma independiente la intensidad de los tonos básicos[8].

5. METODOLOGÍA

Mediante el desarrollo de este proyecto de grado se planteó una metodología, teniendo en cuenta tres etapas especificas las cuales se fueron realizando en busca de resusltados y conclusiones que nos permitierón cumplir los objetivos planteados desde el inicio. De manera general se describen a continuación cada una de las etapas empleadas.

5.1. DOCUMENTACIÓN Y RECOPILACIÓN DE LA INFORMACIÓN

la etapa de documentación y recopilación de la información consiste en reunir documentos referente a las temáticas ya definidas para las prácticas de laboratorio de Procesadores, dentro de estos tenemos: Articulos, hojas de datos y manuales que se utilizaron en el marco teórico de los laboratorios. También se recopiló información acerca de prácticas de laboratorio en otras universidades como por ejemplo la Universidad Nacional de Colombia y la Universidad de Vigo en España.
Luego de la documentación se realizó una breve introducción a Linux, con el fin de conocer los comandos de base para la implementación de los laboratorios en la tarjeta SIE.

5.2. DESARROLLO DE LAS PRÁCTICAS DE LABORATORIO.

Una vez realizada la etapa inicial de recopilación de información, se realizó el montaje de los laboratorios, el manual de usuario y el proyecto final de la asignatura, guiados un poco con la estructura aplicada en la Universidad Nacional de Colombia, la cual está implementando la plataforma SIE dentro de sus cursos en el área de los sistemas digitales. Igualmente se desarrolló la revisión de las prácticas para comprobar que cumplieran con los objetivos de la asignatura. En este proceso se contó con la ayuda de los compañeros encargados de los proyectos sobre la plataforma SIE y el docente a

cargo de la asignatura Procesadores.

5.3. VERIFICACIÓN Y VALIDACIÓN DE LAS PRÁCTICAS DE LABORATORIO

Una de las tareas más importantes dentro del desarrollo del proyecto es la verificación y la validación de las prácticas de la asignatura procesadores, dónde el objetivo fundamental del proceso de validación y verificación es la de producir material didáctico confiable qué al momento de aplicarlo en un aula de clase cumpla con las expectativas planteadas y alcance los objetivos mencionados satisfactoriamente para cada actividad. En general la verificación enfoca el tema de la consistencia interna de un modelo, mientras que la validación se relaciona con la correspondencia de ese modelo y con la realidad. De manera más sencilla la verificación trata sobre la cuestión ¿Se está construyendo correctamente el sistema? mientras que la validación responde a ¿Se está construyendo el sistema correcto?[2].

Partiendo de esas dos premisas se comprobó por medio de un grupo de estudiantes de Ingeniería Electrónica de la Universidad Industrial de Santander qué las prácticas desarrolladas cumplen con las características que debe tener el material didáctico. Dicho grupo de estudiantes fue elegido por los autores del proyectó basados en la experiencia obtenida por ellos en el manejo de la plataforma SIE y en algunas temáticas planteadas en cada una de las actividades. También se tuvo en cuenta la participación y los comentarios realizados por parte del profesor encargado de la asignatura Procesadores [9] para corroborar que los objetivos se cumplierón. Adicionalmente las correcciones realizadas por parte del grupo de estudiantes y el docente encargado de la asignatura fueron tomadas en cuenta, garantizando que las prácticas no contengan errores y permitiendo la debida realimentación hacia las actividades propuestas.

A continuación se especifica el estudiante encargado de cada práctica. Cabe mencionar que algunos de los estudiantes escogidos se encuentran realizando el proyecto de

[9] MSc.William A. Salamanca Becerra

grado con el grupo de investigación CPS basados en la plataforma SIE con énfasis en las asignaturas Sistemas Digitales, Sistemas Embebidos y Arquitectura de Computadores:

Tabla 2. Listado de estudiantes encargados de las prácticas

PRÁCTICA	**ESTUDIANTE**	**CÓDIGO**
1	Héctor Gomez	2031745
2	Robinson Hernandez	2052152
3	Luis Carlos Ubayan	2032882
4	Miguel Muñoz	2032635
5	Pedro Vargas	2063234
6	Cesar Quiñonez	2005452

Finalmente como parte principal en el desarrollo de la verificación y validación de las actividades y el proyecto final de la asignatura, el encargado de la asignatura de procesadores realizó una serie de correcciones, sugerencias y comentarios a cada una de las actividades propuestas y el proyecto final, ofreciendo un nivel de seguridad en la metodología utilizada, tanto en la escogencia de las temáticas como también en la representación de las prácticas. Cada una de las correcciones y sugerencias fueron atendidas y aplicadas a las actividades, completando de ésta manera el proceso de verificación y validación.

6. MANUAL DE USUARIO

Este capítulo del informe se desarrolló con el fin de cumplir el siguiente objetivo específico:

- Elaborar un manual de usuario intermedio de la plataforma SIE orientando su aplicación a la asignatura Procesadores.

Con el objetivo de cumplir satisfactoriamente la apropiación de las temáticas expuestas dentro de las prácticas propuestas, se decidió realizar la construcción de un manual intermedio de la plataforma SIE orientado a aspectos relacionados con la asignatura procesadores. Este manual contiene una estructura clara y sencilla, facilitando la comprensión de procedimientos técnicos que no son expuestos en las prácticas y que son necesarios en el desarrollo de la actividad propuesta.

La estructura del manual es intermedia, debido a que los usuarios poseen conocimientos de algunos aspectos relacionados con la plataforma SIE, teniendo en cuenta que ya han cursado y aprobado la asignatura anterior de Sistemas Digitales. Con este enfoque se busca que el estudiante haga un recuento de aspectos necesarios para el desarrollo satisfactorio de las prácticas .Teniendo en cuenta recomendaciones por parte del encargado de la asignatura de procesadores y criterios propios de los autores, basados en la experiencia que se logró con el desarrollo e implementación de las actividades propuestas, se decide que el manual contenga los siguientes temas:

- **Introducción**: Permite entregar una idea de la estructura, haciendo énfasis en el objetivo principal del manual, consiguiendo que el estudiante comprenda y utilice de manera correcta la información contenida dentro del manual de usuario de la plataforma SIE.
- **Tarjeta SIE**: Aunque los aspectos relacionados con la tarjeta SIE están contenidos dentro del manual de usuario de la asignatura Sistemas Digitales, se decidió incluirlo en este manual con el propósito que el estudiante realice nuevamente una revisión a las especificaciones ofrecidas por la tarjeta sin necesidad de dirigirse al manual ofrecido en la asignatura anterior. A través de esta sección del manual el estudiante se encontrara con un breve recuento de las especificaciones de la tarjeta

SIE, brindándole una idea específica de los dispositivos más comúnmente utilizados dentro de los diseños a realizar.

- **Diagramas *RTL* y *Technology Schematic***: Teniendo en cuenta aspectos muy importantes dentro del buen análisis de los diseños realizados en las actividades propuestas, se hace necesario la utilización de herramientas muy poderosas ofrecidas por el entorno de diseño de Xilinx: ISE WebPack. Por ese motivo se plantea un procedimiento sencillo y conciso de los pasos que se deben realizar para obtener y comprender los diagramas *RTL* y *Technology Schematic*. Cabe mencionar que dentro del informe que el estudiante debe entregar como evidencia de la realización de la actividad se requiere la inclusión de los diagramas *RTL* y *Technology Schematic* como soporte de un buen diseño de circuitos.
- **Simulación *Post-Route***: Para llevar a cabo el desarrollo satisfactorio de la actividad propuesta dentro de la práctica 1 (Implementación de circuitos combinacionales y *glitches*) se hace necesario la utilización de la herramienta de simulación Post-Route que brinda ISE WebPack. Debido a que los *glitches* son fenómenos imperceptibles con la simulación realizada normalmente, el estudiante debe tener la capacidad de observarlos por medio de herramientas más precisa. Dentro del manual de usuario se incluye un procedimiento corto y sencillo para obtener la simulación *Post-Route* y detectar la aparición de los *glitches*.
- **Accediendo al procesador de la plataforma SIE**: Se incluye esta sección dentro del manual de usuario como parte de un recuento general de comandos importantes de Linux para la utilización de la tarjeta SIE. Se conoce de antemano que esta información está contenida dentro del manual de usuario de la asignatura Sistemas Digitales con mayor detalle. Por ese motivo solo se plantean algunos comandos que son de mucha utilidad en la implementación de las actividades propuestas con la plataforma SIE. Es de suma importancia que el estudiante maneje de manera correcta el procedimiento para acceder al procesador de la plataforma SIE ya que depende de ello que la implementación funcione correctamente.

7. CONCLUSIONES

- La plataforma SIE proporciona herramientas académicas necesarias para desarrollar un gran número de actividades prácticas contenidas dentro de un curso de técnicas de diseño digital avanzado.

- Las prácticas de la asignatura procesadores tienen como objetivo desarrollar en el estudiante destrezas en el diseño de sistemas digitales avanzados y procesadores de propósitos específicos, los cuales en algun momento de su vida profesional prodrían ser aplicadas para crear dispositivos comercializables que satisfagan una necesidad determinada en la comunidad.

- Las prácticas de laboratorio de la asignatura procesadores están diseñadas de manera que le otorguen al docente una herramienta que permita ayudar al estudiante en el desarrollo de sus competencias propositivas en cuanto al diseño digital avanzado se refiere.

- El proyecto final de la asignatura procesadores permite la utilización de herramientas abiertas para que el estudiante conozca y aplique diferentes metodologías de diseños, siendo capaz de ajustarlas a diferentes situaciones.

- Adicionalmente se logró dar continuidad a los contenidos del proyecto final de la asignatura como base para proyectos futuros, utilizando la metodología del proyecto prolongado, logrando que el estudiante aporte ideas innovadoras a un tema especifico.

- El manual de usuario brinda al estudiante información básica que le permite solucionar problemas relacionados con la implementación de la plataforma SIE en la asignatura procesadores y algunos aspectos importantes de la herramienta Xilinx: ISE Web Pack.

- La verificación y validación garantizó que el montaje y desarrollo de las actividades propuestas dentro de las prácticas y el proyecto final de la asignatura cumplierán con las especificaciones planteadas, satisfaciendo las necesidades del estudiante y

asegurando que el modelo propuesto sea correcto con relación al comportamiento del diseño implementado que se analiza.

8. RECOMENDACIONES

Mediante el desarrollo de los laboratorios de la asignatura procesadores se evidenciarón una serie de sugerencias las cuales son importantes para que el desarrollo del proyecto de grado se realice de manera correcta.

Las prácticas de laboratorio fueron elaboradas como apoyo experimental a cada una de las temáticas vistas en el transcurso de la asignatura. Asi que se recomienda implementar una exposicion teórica de conceptos acerca del desarrollo de cada una de las experiencias de laboratorio. De esta manera el estudiante cuenta con una herramienta de apoyo que colabore para alcanzar los objetivos planteados.

Se sugiere que cuando se implementen las prácticas de laboratorio en la asignatura procesadores, se exponga herramientas bibliográficas acerca de la utilización del entorno Xilinx:ISE WebPack, para que el estudiante se documente y afronte situaciones que se presentarán al momento de interactuar con esta herramienta, ya que muchos estudiantes en el trancurso de su proceso de formación lo utilizán de manera ineficiente dejando de lado muchos elementos importantes.

Finalmente se recomienda innovar en nuevas temáticas en el transcurso de la asignatura procesadores y también en el desarrollo del proyecto final, ya que esto ayudará a estar a la vanguardia de nuevas tegnologias en el área de los sistemas digitales.

BIBLIOGRAFÍA

[1] http://en.qi-hardware.com/wiki/SIE, Abril 2012.

[2] P. de la Fuente, "Verificación y Validación,"Universidad de Valladolid, http://jair.lab.fi.uva.es/~pablfue/leng_simulacion/materiales/v_v_0405.pdf

[3] M. L. Sepúlveda Gómez and J. M. Forero Gamboa, "Apropiación tecnológica de los dispositivos disponibles en el laboratorio de redes de datos de la escuela de ingenierías eléctrica, electrónica y telecomunicaciones," Universidad Industrial de Santander, 2007.

[4] J. D. Hernández, C. Vásquez Blanco, and E. Duque López, "PROCESADO DE EFECTOS DE AUDIO BAJO FPGAs," 2002.

[5] M.I.Kubski,"Introduccion a LINUX,"Universidad Nacional del Nordeste http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/LINUXMariana.pdf

[6] C. I. Camargo, " SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales" pp. 3-5, http://linuxencaja.net/images/a/af/Open_HW_education.pdf

[7] M. Garrels,"Introduction to Linux, A Hands on Guide," vol. 6, no. December 2002, p. 15, 2006.

[8] E.G.Gutiérrez,"Efectos Digitales Básicos,"Escuela superior de musica http://www.dtic.upf.edu/~egomez/teaching/sintesi/SPS1/Tema10-EfectosDigitales.pdf

[9] J. Wakerly, Diseño Digital. Pearson educación.

[10] C. Pong P, FPGA PROTOTYPING BY VHDL EXAMPLES, Tercera. Jhon Wiley & Sons, 2008.

[11] L.J.Álvarez Ruiz de Ojeda,"Diseño Digital con Lógica Programable (Metaestabilidad)," Universidad de Vigo, 2007, http://www.dte.uvigo.es/logica_programable/documentos/curso_disenho_digital_con_CDCs/Metaestabilidad.pdf

[12] J.Bastida Ibáñez, "Procesadores Segmentados," Universidad de Valladolid, http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Segment.pdf

[13] C. A. Fajardo and J. H. Ramón, Introducción al Diseño Digital Avanzado, 1.1 ed. Universidad Industrial de Santander, 2010.

[14] XILINX, "XST User Guide," vol. 10.1, pp. 1-600, http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf

[15] http://www.esdebian.org/wiki/lista-comandos-gnulinux-i

ANEXO A: PRÁCTICAS DE LABORATORIO PROCESADORES

LABORATORIO 1

IMPLEMENTACIÓN DE CIRCUITOS COMBINACIONALES Y GLITCHES

El experimentador que no sabe lo que está buscando no comprenderá lo que encuentra.

CLAUDE BERNARD

1.1 INTRODUCCIÓN

Hacer un buen diseño de un circuito combinacional no consiste unicamente en simplificar la función lógica del circuito, puesto que se han de tener en cuenta ciertos parámetros para la implementación en la FPGA, de tal forma que se emplearan de manera correcta elementos tales como las celdas lógicas compuestas básicamente por:*LUT*, *Flip Flop*, *Mux*, *And* y *Latch*.

Por otra parte para lograr un diseño que se desempeñe de la mejor manera, debemos eliminar los problemas más comunes que se presentan en los circuitos combinacionales y secuenciales, ya que particularmente en los circuitos combinacionales pueden presentarse transitorios anómalos en las salidas por culpa de la existencia de retardos en los dispositivos lógicos. Dependiendo del uso que se vaya a hacer de las salidas, la aparición de pulsos indeseables puede ser irrelevante o catastrófica, estos valores indeseables son denominados *glitch*. Por consiguiente se debe tener en cuenta los problemas que generan los *glitchs* determinando cuando son malos y como se evitan o detectan. Además entender que no se quieren en las señales de reloj, señales asíncronas de set/reset y las señales de habilitación en triestados. En pocas palabras para detectar dichos *glitchs* es necesario conocer las causas, por lo tanto a través de esta práctica se identificarán las causas de la aparición del *glitch* en un circuito medianamente complejo, analizando que estrategias se pueden aplicar para evitar estos riesgos y de esta forma mejorar el diseño del sistema digital.

1.2 OBJETIVOS

- Identificar problemas comunes en los circuitos combinacionales y secuenciales
- Adquirir destrezas en la aplicación de metodologías de diseño que permitan corregir los problemas comunes en el mal uso de las señales asíncronas.
- Aplicar estrategias que permitan generar los *glitchs* en el diseño de sistemas digitales.

1.3 MARCO TEÓRICO

En la implementación de circuitos combinacionales pueden presentarse pulsos no deseados debido a la aparición de retardos en los circuitos lógicos. Dependiendo del uso que se le dé a las salidas estos pulsos pueden ser o no causa de graves problemas en la implementación. Además si el circuito presenta transitorios pueden dar lugar a oscilaciones en la salida[2]. Estos valores transitorios se conocen como glitches y se clasifican en dos tipos, por su forma: estáticos, dinámicos y por sus causas en: funcionales y lógicos.

1.3.1 TIPOS DE *GLITCHS*

- **Según su forma**

  Estáticos

  Son pulsos que aparecen debido a un cambio en las entradas que no genera ningun cambio en el estado de la salida, clasificándose en static-zero o static-one (ver figura 1.1 y 1.2)[2].

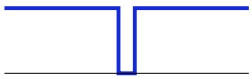


Figura 1.1: Static-one.Tomada de [2]

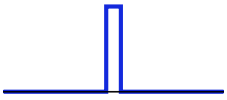


Figura 1.2: Static-cero.Tomada de [2]

  Dinámicos

  Son pulsos falsos que aparecen por culpa de un cambio en las entradas que produce un cambio no deseado en el estado de la salida (ver figura 1.3)[2].

Figura 1.3: *Glitch dynamic.*Tomado de [2]

- **Según sus causas**

Funcionales

Son glitchs que sólo aparecen cuando se producen cambios en más de una variable de entrada (ver figura 1.4)[2].

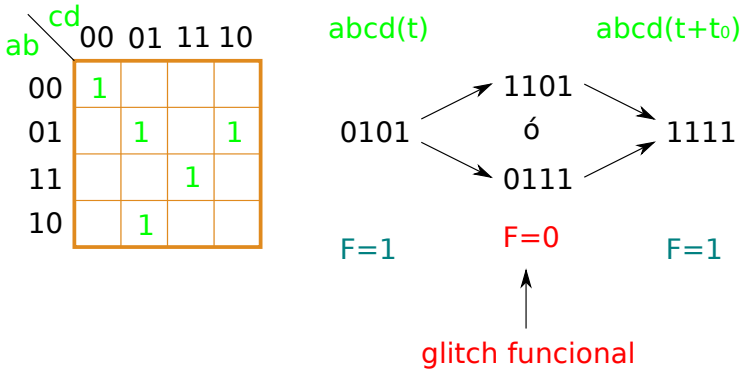


Figura 1.4: *Glitches* funcionales.Tomada de [2]

Lógicos

Son debidos a la realización hardware del circuito y pueden producirse aunque sólo cambie de estado una entrada (ver figura 1.5)[2].

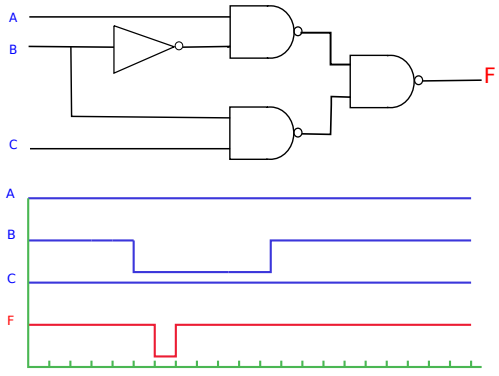


Figura 1.5: *Glitches* logicos.Tomada de [2]

1.3.2 ¿COMO DETECTAR Y CORREGIR LOS *GLITCHS*?

Para evitar los *Glitchs* hay diferentes alternativas, las cuales se mencionan a continuación :

- **Inserción de retardos hardware**: Mediante la inserción de retardos en lugares específicos y ecualizando dichos retardos la probabilidad de que aparezcan disminuye, pero su valor es difícilmente controlable y varía con las condiciones de funcionamiento, pudiendo dar lugar a nuevos riesgos. Este método es conocido como **Ecualización**[2].
- **Inserción de lógica redundante**: *Permite eliminar únicamente los riesgos lógicos*[2].
- **Inserción de lógica registrada**: Esta solución es la más ampliamente utilizada y es tambien conocida como **Metodología de diseño Síncrono** y consiste en no eliminar los *glitchs* sino su efecto, ya que muestrea la lógica cuando esta ha concluido el régimen transitorio para despues almacenar los estados de salida en *Flip Flops* y de esta manera eliminar los *glitchs*[2].

Para poder evitar la aparición de un *glitch* se responden varias preguntas las cuales se muestran a continuación:

¿Cómo se detectan los *glitchs*?

Primero que todo hay que entender que los *glitchs* no se pueden detectar por medio de la simulación y la medición. La mejor forma de detectarlos es identificando las causas que los generan y a través del análisis del circuito es la única forma. Los mapas de *Karnaugh* pueden utilizarse para detectar los riesgos, pero la existencia o inexistencia depende del diseño del circuito, en la figura 1.6 se muestra el mapa de *karnaugh* y el circuito de un posible *glitch*.

La transición que genera el posible glitch, atraviesa de un grupo en el mapa a otro.

¿Cómo se corrigen los *glitchs*?

Una manera de corregir el problema de los *glitchs* es la adicción de un grupo extra en el mapa de *karnaugh* que mantenga la salida siempre activa durante la transición critica (véase la figura 1.7).

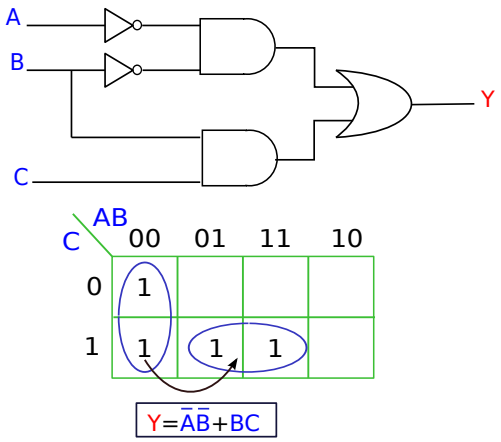


Figura 1.6: Circuito con riesgos y su mapa de *karnaugh*.Tomada de [1]

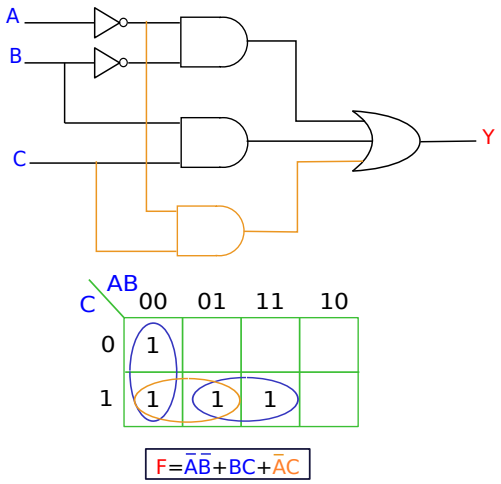


Figura 1.7: Circuito sin riesgos y su mapa de *karnaugh*.Tomada de [1]

La transición que generaba el *glitch* es eliminada, agregando una compuerta *And* en el diseño del circuito.

Ahora bien, se sabe que para corregir los riesgo hay que detectarlos y para detectarlos hay que encontrar sus causas, luego se entiende que hay muchas transiciones que generan *glitchs* y se llega a un punto donde es muy complicado hacer un diseño de un circuito combinacional completamente libres de estos riesgos. La solución a este problema es hacer diseños de forma que los circuitos que reciban los *glitchs* no se afecten por ellos, un claro ejemplo de estos circuitos son los *Flip Flop*.

Cualquier circuito combinacional puede ser analizado en busca de riesgos. Sin embargo, un sistema digital síncrono, muy bien diseñado está estructurado de modo que el análisis de riesgo no sea necesario [1].

1.4 DESCRIPCIÓN DEL PROBLEMA

La práctica requiere que el estudiante pueda identificar los posibles *glitchs* en un circuito combinacional y desarrolle estrategias de diseños libres de este problema, por esta razón se propone el siguiente ejercicio: Implementar y diseñar un circuito generador de *glitchs*. Para esto tenga en cuenta qué tipo de circuitos generan *glitchs*. Luego de haber definido su circuito generador, realice un circuito que mida la cantidad de *glitchs* en el cual se pueda mostrar y/o detectar si es posible que efectivamente se están generando los *glitchs*.
Con la ayuda del Pmod siete segmentos mostrar el número de *glitchs* generados en la implementación.Tenga en cuenta comprobar por medio de la simulación el buen funcionamiento del diseño. Recuerde que para desarrollar la actividad debe entender el funcionamiento del sintetizador de Xilinx(xst) con mayor profundidad. Con el objetivo de que usted pueda comprender mejor el funcionamiento del sintetizador, se propone el siguiente ejercicio: Implementar una compuerta *XOR* y AND con 2 entradas, con 4, con 6 y por ultimo con 8 entradas y observe los circuitos RTL y Technology schematic, identificando que componentes lógicos son utilizados y cuantos son aprovechados dentro de la implementación de cada una de las entradas. Adicionalmente consulte las propiedades del sintetizador y como influyen en la implementación del respectivo diseño. Se recomienda apoyarse en el documento XST User Guide[4].

Realice la descripción del hardware, eligiendo los elementos necesarios para el correcto funcionamiento de la práctica, implementándolo completamente en la FPGA de la tarjeta SIE.

Nota: Adicionalmente para la realización del circuito generador de *glitchs* , se expone a manera de investigación un método diferente al mostrado en el marco teórico. La nueva opción consiste en construir el circuito generador de *glitchs* por medio de los bloques básicos de Xilinx (primitivas), conectando diferentes elementos como *LUT, And y MUX*, para formar un circuito medianamente complejo que ocasione oscilaciones en la salida y por ende *glitch*.Toda la información para construir estos circuitos se puede obtener por medio de la siguiente bibliografia [4].

1.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- *RTL y Technology Schematic*.
- Códigos en VHDL
- Simulación de los módulos descritos: Generador de *glitch* , detector de *glitch* y circuito general donde se acoplen todos los diseños del sistema.

- Circuito esquemático completo que incluya todos los dispositivos a utilizar (Tarjeta SIE (como una caja negra), dipswitch, resistencias, sensores, Pmod, etc)).

1.6 PREGUNTAS DE PRUEBA

1. ¿Qué tan frecuentes son los glitchs en un circuito digital?.
2. ¿Cómo podemos medir la frecuencia con la que se generan estos glitchs?.
3. ¿Cuáles son las causas más comúnes por las cuales aparecén los glitchs?.

1.7 REFERENCIAS BIBLOGRAFÍCAS

[1]J. Wakerly, Diseño Digital. Person educación.

[2]Escuela Universitaria de ingeniería Técnica de la telecomunicación, "Diseño Síncrono de circuitos digitales," Politécnica de Madrid, 2011,
`url:http://www.euitt.upm.es/uploaded/464/teoria/tema3/Tema3.pdf.`

[3]R. S. S. Hernández, "Procesador digital genérico(Subsistemas Combinacionales)," Universidad Politécnica de Madrid, 2010,
`url:http://dcse.die.upm.es/Grupo4445/2_IntroCombinacionales.pdf.`

[4]XILINX, "XST User Guide," vol. 10.1, pp. 1-600,
`url:http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf.`

LABORATORIO 2

IMPLEMENTACIÓN DE MÁQUINAS DE ESTADO EN VHDL

La formulación de un problema, es más importante que su solución..

ALBERT EINSTEIN

2.1 INTRODUCCIÓN

Las maquinas de estados (FSM por sus siglas en ingles) son uno de los medios más utilizados en la implementación de aplicaciones digitales.

Las FSMs permiten definir una serie de estados y cada uno con un comportamiento distinto a las entidades que los contienen, en consecuencia nos podemos encontrar en un instante en particular y gracias a unas entradas hacernos transitar de un estado a otro permitiéndonos activar acciones o salidas relacionadas con dicho estado. Por su simplicidad y rapidez en el diseño e implementación las máquinas de estados se han convertido en una herramienta muy poderosa para el ingeniero electrónico de hoy y por lo tanto en esta práctica se busca aplicar todo el conocimiento sobre las FSMs a través de la implementación de un ejemplo, donde se aplique la metodología para realizar maquinas de estado de manera correcta usando el lenguaje VHDL y corrigiendo los problemas de *glitchs*.

Cualquier sistema reactivo que se preste al diseño con Máquinas de Estado Finito (FSM), debería hacer uso de este patrón de programación, ya que éste provee una abstracción del sistema que resulta natural, segura, fácil de entender y eficiente.

2.2 OBJETIVOS

- Describir las principales característica de las máquinas de estado y los convenios utilizados para implementarlas de manera correcta.
- Adquirir destrezas en la implementación de maquinas de estados usando VHDL a través de la aplicación de un ejemplo práctico.
- Implementar la metodología utilizada para corregir los problemas de *glitchs* que se generan en una maquina de estado

2.3 MARCO TEÓRICO

En la figura 2.1 se observa un ejemplo de circuitos secuenciales sincrónicos. Estas formas se conocen como máquinas de estados finitas(FSM). Ellas deben su nombre a un circuito con k registros, donde puede estar en uno de un número finito de estados únicos (2^k). La FSM tiene M entradas,N salidas,y k bits de estado,también recibe un reloj y opcionalmente, una señal de reset[3].

La FSM consiste en dos bloques de lógica combinacional, la lógica del estado siguiente y de la lógica de salida, también un registro que almacena el estado en cada flanco de reloj, el FSM avanza al siguiente estado, que se calcula en base al estado actual[3].

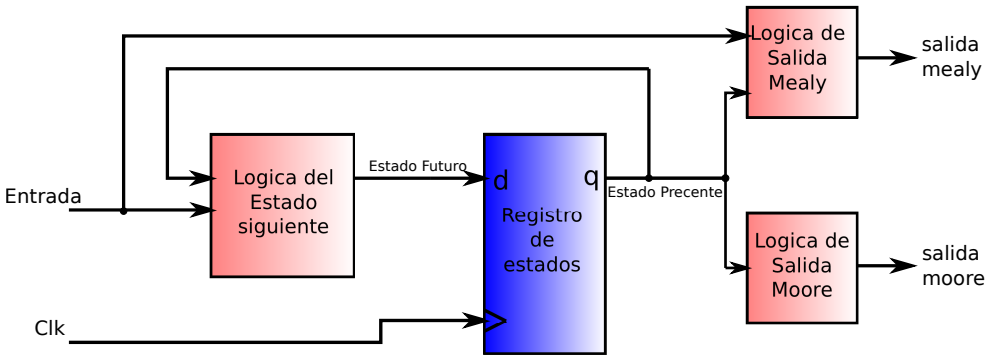


Figura 2.1: FSM (libro de Chu).Tomada de [3]

Las FSMs se pueden clasificar en dos clases generales, que se identifican porsus características funcionales (Moore y Mealy).

- **Moore**: Las salidas dependen sólo del estadoactual de la máquina.
- **Mealy**: Las salidas dependen tanto del estado actual y las entradas del circuito.

Las Maquinas de Estados Finitas proporcionan una manera sistemática para el diseño secuencial síncrono de circuitos debido a una especificación funcional en particulatr [3].

Por otra parte para hacer un diseño óptimo de la FSM, se debe recordar el problema que se presenta debido a los *glitchs*. Al realizar la inspección de la FSM presentada hasta el momento vemos que en las salidas los elementos combinacionales pueden generar *glitchs*. Una solución posible para evitar este fenomeno es insertar *Flip Flops* directamente en las salidas para eliminar los *glitchs*, en consecuencia esto las retarda un ciclo entonces haciendo depender la salida directamente del estado siguiente es posible eliminar el ciclo de retardo que se genera al insertar los *Flip Flops* a la salida para eliminar los *glitchs*.

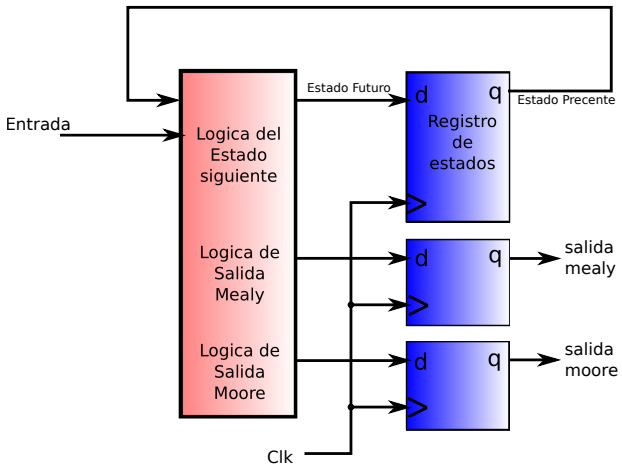


Figura 2.2: FSM libre de *glitchs*. Tomada de [3]

Despues de haber realizado el analisis de como evitar la apricion de *glitchs* en las FSMs , se procede a implementar la máquina de estados corregida en VHDL, por lo tanto hay que tener en cuenta insertar las siguientes sentencias para evitar que el sintetizador codifique los estados y de esta manera se realice de forma manual.

En el script 2.1 se observa el código en VHDL de la entidad.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity FSM is
7      Port ( clk      : in  STD_LOGIC;
8             reset    : in  STD_LOGIC;
9             entrada  : in  STD_LOGIC;
10            salida   : out STD_LOGIC);
11
12          attribute FSM_EXTRACT: string;
13          attribute FSM_EXTRACT of FSM: entity is "NO";
14 end FSM;
```

Script 2.1: Código VHDL de la entidad FSM libre de Glitch.

La descripcion de los estados se realiza de la siguiente manera:

Primero se define el numero bits del registro de estados como el numero de estados mas el numero de salidas. Luego se declaran los estados , las señales de estado presente y estado siguiente como se muestra en la figura 2.3.

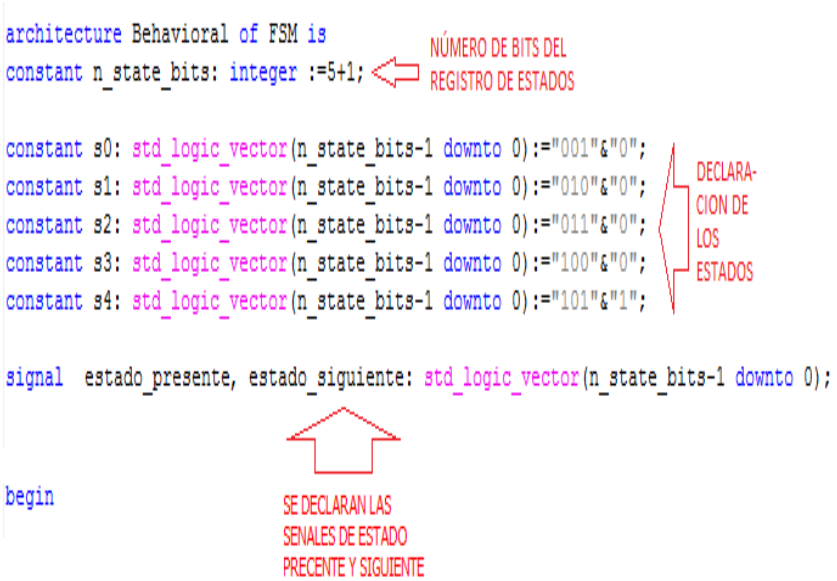


Figura 2.3: Codigo VHDL de la declaracion de los estados

La asignación de la salida de la FSM sin *glitchs* al estado presente del sistema se hace de la siguiente manera (véase figura 2.4).

```
salida <= estado_presente(0);
```

Figura 2.4: Asignación de la salida de la FSM sin *glitchs*.

2.4 DESCRIPCIÓN DEL PROBLEMA

En la figura 2.5 se muestra el diagrama de bloques del problema a resolver por medio de máquina de estados (FSM). Se resolverá un ejercicio práctico, el cual implementa una maquina de estados que cuente el número de automóviles en un parqueadero. De la siguiente manera:
Con la ayuda de dos sensores (D1, D2) se implementá el contador de los automóviles en el parqueadero de esta manera: primero al activarse D1 y luego D2, indica que un auto está ingresando al parqueadero, por el contrario, si se activa primero D2 y luego D1 nos indica que un auto está saliendo del parqueadero. Realice el control del ingreso y salida de los automóviles, teniendo en cuenta que se debe conocer el numero de autos que se encuentran dentro del parqueadero, como la diferencia entre el número de ingresos, menos el número de salidas de los automóviles. Todo este diseño se implementará en la tarjeta SIE y la construcción del hardware de la práctica se deja a criterio del estudiante. Tenga en cuenta realizar todos los pasos que se encuentran en el contenido del marco teórico para el diseño de este circuito secuencial (FSM sin *glitchs*).
Realice la descripción del hardware, eligiendo los elementos y herramientas a utilizar para el correcto funcionamiento de la práctica. Se debe diseñar el control general de la práctica, implementándolo completamente en la FPGA de la tarjeta SIE.

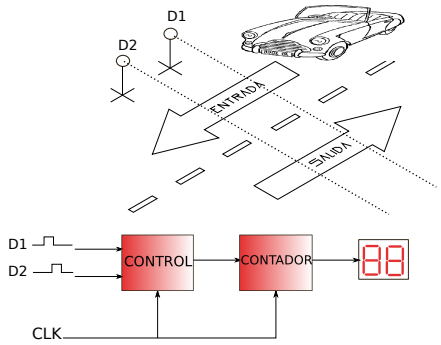


Figura 2.5: Diagrama de bloques del problema.Tomada de [1] y modificada

2.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- Diagrama de estado.
- *RTL y Technology Schematic.*
- Códigos en VHDL
- Simulación del diseño completo
- Circuito esquemático completo que incluya todos los dispositivos a utilizar (Tarjeta SIE (como una caja negra), dipswitch, resistencias, sensores, Pmod, etc)).

2.6 PREGUNTAS DE PRUEBA

1. ¿Por qué es importante tener una maquina de estados libre de glitches?.
2. ¿Cuál es la diferencia entre sintetizar los estados de forma manual y no hacerlo?

2.7 REFERENCIAS BIBLOGRAFÍCAS

[1]http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/laboratorios/070101.htm

[2]J. Wakerly, Diseño Digital. PEARSON EDUCACIÓN.

[3]C. Pong P, FPGA PROTOTYPING BY VHDL EXAMPLES, Canada. Jhon Wiley & Sons, 2008.

LABORATORIO 3

VIOLACIONES DE SETUP Y HOLD "METAESTABILIDAD"

Un científico debe tomarse la libertad de plantear cualquier cuestión, de dudar de cualquier afirmación, de corregir errores.

Julius Robert Oppenheimer

3.1 INTRODUCCIÓN

En el transcurso de la asignatura de procesadores, se han venido mencionando conceptos teóricos interesantes y que permiten tener una visión más clara de cómo diseñar de manera correcta un sistema. En esta práctica se dará a conocer un problema denominado metaestabilidad que tienden a sufrir los biestables síncronos. Conceptos importantes como tiempo de *Setup* y tiempo de *Hold*, se deben tener en cuenta para poder entender las causas de este fenómeno. También se dará a conocer la definición de metaestabilidad, qué implica, causas de las violaciones de *Setup* y *Hold*, qué sucede cuando aparece y por supuesto cómo se puede evitar. Por medio de un ejercicio se colocará en práctica todos los conceptos anteriormente mencionados, utilizando como herramienta principal la implementación del mismo y observar cuando aparece este fenómeno.

3.2 OBJETIVOS

- Emplear conceptos como tiempo de *Setup* y *Hold* para comprender qué causa una violación de estos tiempos y cuáles son sus consecuencias.
- Conocer la definición de metaestabilidad, cuáles son sus causas y como se puede evitar.
- Implementar un ejercicio práctico para observar y entender cuando aparece el fenómeno de la metaestabilidad.
- Utilizar la información contenida dentro del datasheet de la tarjeta SIE, para conocer los tiempos de *Setup* y *Hold* en beneficio de una buena implementación de la práctica.

3.3 MARCO TEÓRICO

Con el fin de comprender y analizar las causas de la metaestabilidad, se presentan las definiciones de tiempo de *Setup*, tiempo de *Hold* y tiempo de propagación, como parte del material de apoyo para la realización de la práctica.

La salida de un *Flip Flop* cambia luego de un flanco activo, ya sea de subida o de bajada (ver figura 3.1). Cada fabricante al caracterizar el funcionamiento de estos biestables, establece en la hoja de datos ciertos tiempos, los cuales deben ser respetados por la entrada del mismo para que funcione correctamente. También indica el tiempo que demora la salida en cambiar luego de un flanco de reloj[1]. A continuación se definen estos tiempos, denominados tiempo de *Setup*, tiempo de *Hold* y tiempo de propagación.

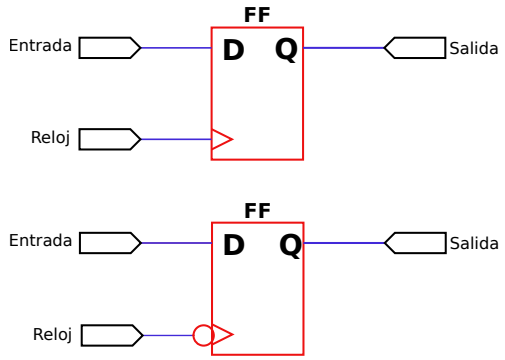


Figura 3.1: a)*Flip Flop* sensible al flanco de subida b)*Flip Flop* sensible al falnco de bajada.Tomada de [1]

- **Tiempo de *Setup***

  Es el mínimo tiempo en el que debe estar constante la entrada de datos de un *Flip Flop* antes del flanco activo del reloj para funcionar correctamente (ver figura 3.2)[1].

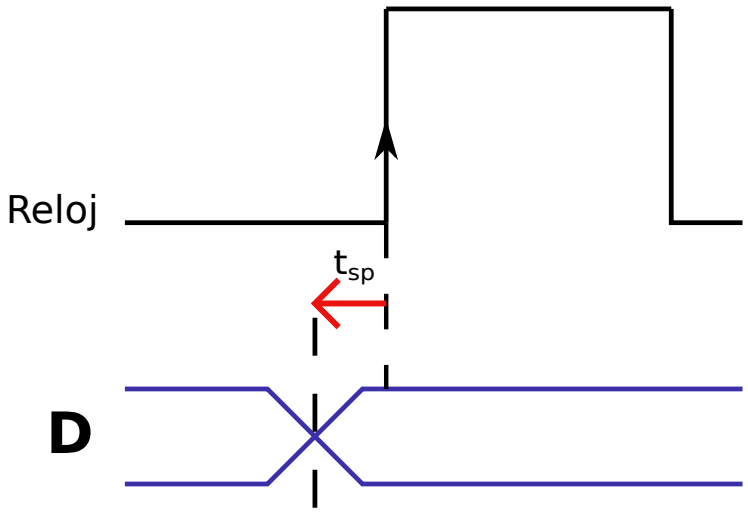


Figura 3.2: Tiempo de *Setup* (*Flip Flop* sensible al flanco de subida).Tomada de [1]

- **Tiempo de *Hold***

  Es el mínimo tiempo que debe permanecer constante la entrada de datos de un *Flip Flop* después de un flanco de reloj (ver figura 3.3)[1].

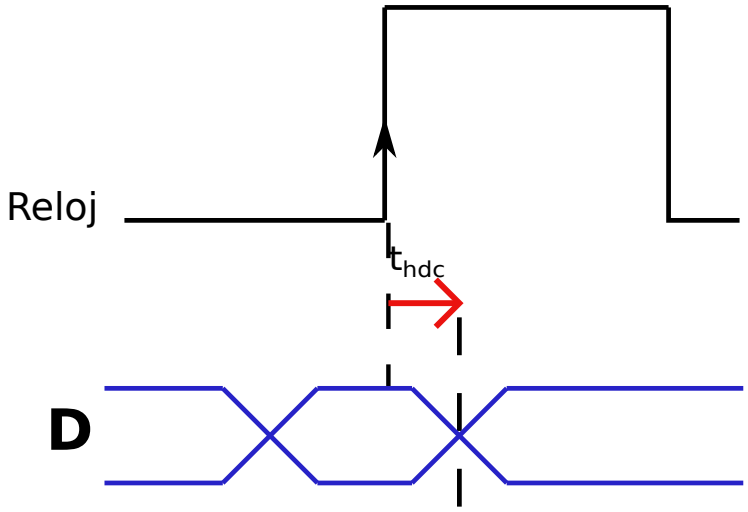


Figura 3.3: Tiempo de *Hold* (*Flip Flop* sensible al flanco de subida).Tomada de [1]

- **Tiempo de propagación**

  Es el tiempo que transcurre luego de un flanco activo de reloj para que la salida del *Flip Flop* quede estable (ver figura 3.4)[1].

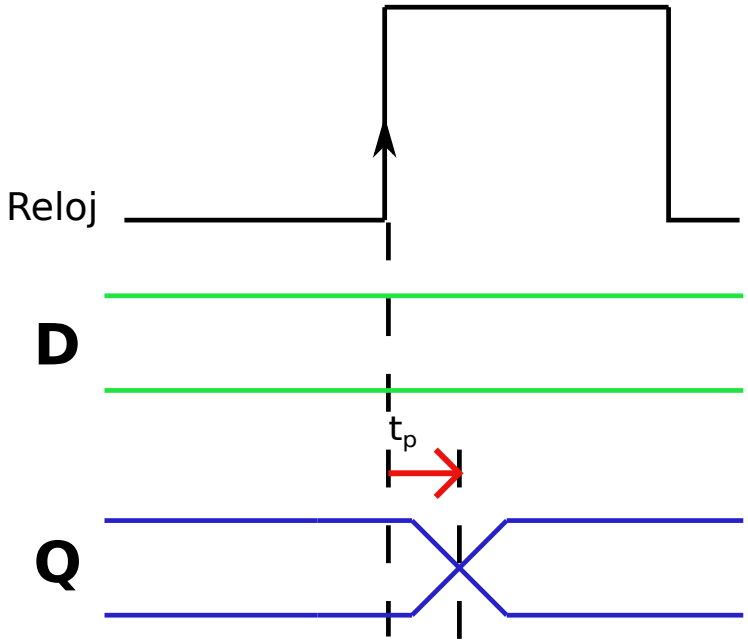


Figura 3.4: Tiempo de propagación (*Flip Flop* sensible al flanco de subida).Tomada de [1]

Se Analiza las definiciones anteriormente mencionadas, planteándose esta pregunta: ¿Qué sucede si una transición de reloj sucede al tiempo con un cambio en la señal de entrada de un *Flip Flop*?. Esto genera que la salida toma el estado de la entrada justo antes de la transición del reloj. Si esto llega a suceder la salida queda automáticamente en estado de metaestabilidad. Los biestables tiende a sufrir este problema, debido a esto definiremos que es metaestabilidad, que implica, como se puede evitar y además se define el concepto de *Clock skew*.

- **Metaestabilidad**

  Es el conjunto de oscilaciones o indefinición del valor de la salida de un biestable durante un tiempo indeterminado, el cual se debe al incumplimiento de los tiempos de *setup* y *hold*[4].

- **Implicaciones de la metaestabilidad**

  Debido a la aparición de este fenómeno, la salida puede comportarse de forma imprevista, tardando muchas veces más de lo normal en estabilizarse al estado correcto, o incluso podría oscilar repetidas veces hasta terminar en su estado estable.

- **Causas de las violaciones de *setup* y *hold***

  En muchos casos, la metaestabilidad en los biestables se puede evitar asegurándose de que los datos y las entradas de control se mantengan constantes durante un periodo de tiempo especificado antes y después del flanco de reloj. Desafortunadamente, no siempre es posible cumplir estos requisitos. Existen dos causas para que esto suceda las cuales son:

  - **Causas externas**

    Se presenta cuando existen señales externas que no se encuentran sincronizadas con el reloj. Para poder corregir este inconveniente se puede introducir un Flip Flop en la entrada, para poder tener las señales sincronizadas, de esta forma la probabilidad de un suceso metaestable puede reducirse considerablemente, pero nunca podrá eliminarse por completo. Desafortunadamente el nuevo biestable, también puede entrar en metaestabilidad, haciendo que el retardo introducido por la metaestabilidad afecte el primer Flip Flop, entrando también en estado metaestable. Para reducir este suceso podemos añadir más etapas de sincronización, esto genera que la probabilidad de metaestabilidad en el primer Flip Flop disminuya considerablemente, pero con la consideración de que el tiempo de retardo de llegada de la señal asíncrona aumente.

  - **Causas internas**

    Dentro de las causas internas existen dos casos críticos:

    * **Violación del tiempo de *Setup***

      Se presenta cuando el retardo de propagación de la señal desde un ciclo de reloj hasta que se estabiliza la señal de salida se vuelve muy grande.

    * **Violación del tiempo de *Hold***

      Este caso se presenta cuando el retardo de propagación desde la etapa anterior puede ser muy corto.

- ***Clock skew***

  El Clock skew es uno de los atributos más importantes de las señales de reloj y consiste en la diferencia del retraso de propagación de la señal de reloj desde la fuente hasta dos registro secuencialmente adyacente[3]. Si este *skew* es muy elevado, puede llegar afectar seriamente el funcionamiento del sistema y provocar que el dato enviado sea capturado incorrectamente por el registro[3].

  actualmente hay una regla de diseño que dice que el *Clock skew* no debe superar el 10% del periodo de la señal de reloj[3], aunque cabe anotar que según los requerimientos de cada sistema, esta

relación puede ser válida o no. Por otra parte el *Clock skew* puede ser negativo o positivo, dependiendo cuando llega la señal a cada registro[3]. En la figura 3.5 se observa los dos ejemplos antes mencionados.

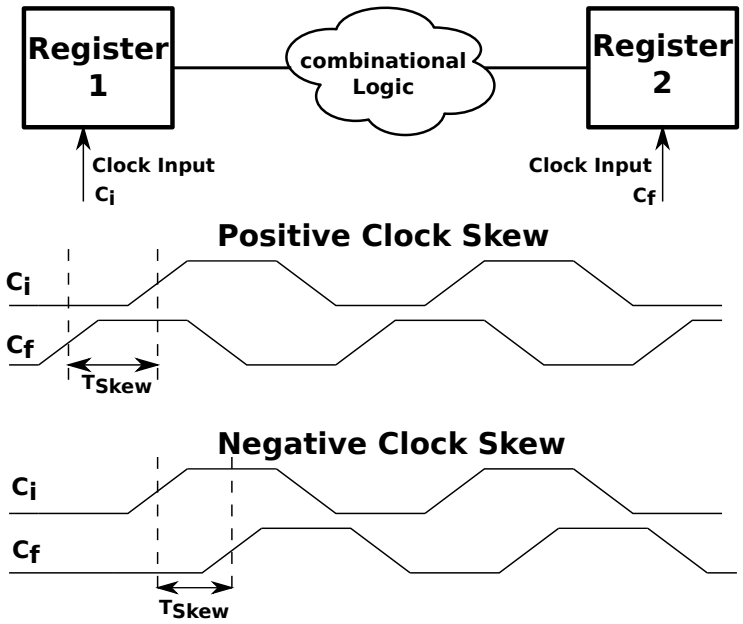


Figura 3.5: *Clock Skew.*Tomada de [3]

Para registros secuencialmente adyacentes, se tiene que C_i y C_f son las señales de reloj y proviene de la misma fuente de señal. El retraso de propagación de las señales de reloj desde el origen hasta R1 y R2 es T_{C_i} y T_{C_f} respectivamente.
La diferencia en la llegada de las señales de reloj de los registros secuencialmente adyacentes se define como el *Clock skew* y se expresa de la siguiente forma[3]:

$$T_{skew} = T_{C_i} - T_{C_f}. \tag{3.1}$$

Cabe anotar que esta expresión es (T_{skew}) válida para registros secuencialmente adyacentes, no teniendo sentido para los que no son adyacentes

3.4 DESCRIPCIÓN DEL PROBLEMA

En esta práctica se requiere que el estudiante analice el comportamiento del fenómeno de metaestabilidad en elementos biestables como los *Flip Flops*, implementando un circuito sencillo en VHDL y determinando

la frecuencia máxima de funcionamiento. También especificando los tiempos de setup, hold y propagación contenidos en la hoja de datos de la FPGA de la tarjeta SIE. Para observar el fenómeno de la metaestabilidad, se planeta introducir un Flip Flop en estado metaestable de la siguiente manera: Primero deberá diseñar un circuito emisor que envié una serie de datos a un circuito receptor, dicho circuito receptor es un *Flip Flop* que trabaja a una frecuencia f. El diagrama de bloques del circuito se muestra a en la figura 3.6.

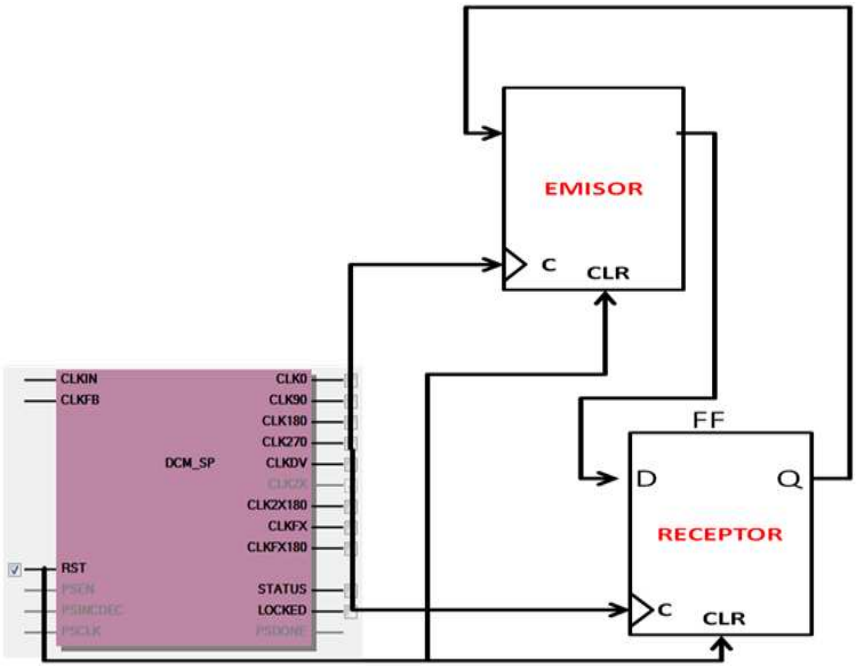


Figura 3.6: Diagrama de blouqes del problema planteado

Se compara la salida del circuito receptor (*Flip Flop*) con la salida del circuito generador de los datos de entrada, de esta manera comprobaremos si los datos enviados son los mismo que los recibidos. En el caso que el dato enviado no coincida con el dato recibido, se determina que el circuito receptor se encuentra en estado metaestable.

Para causar el estado metaestable en el circuito receptor debe usar el DCM que tiene la FPGA de la tarjeta SIE de la siguiente manera: Por medio del DCM genere un *skew*, que desplace la señal de reloj del *Flip Flop* hasta los límites del tiempo de *setup*, hasta conseguir el tiempo necesario para que el biestable entre en metaestabilidad.

Realice este proceso para diferentes valores de *skew* y determine el porcentaje de pérdida de datos por causas de la metaestabilidad e inclúyalos en una tabla donde la primera columna corresponde al valor del skew, la segunda columna al porcentaje de perdida (%pm,) y la tercera el numero de datos enviados.

3.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- Valores de los tiempos de *Setup*, *Hold* y propagación correspondientes a la tarjeta SIE que se encuentra en el datasheet .También el valor de la máxima frecuencia de funcionamiento del circuito.
- Tabla que contenga el número de datos que se envían, los datos errados en la salida del *Flip Flop* para diferentes valores de *skew*..
- Códigos en VHDL
- Comentario del comportamiento para los diferentes valores de *skew*.

3.6 PREGUNTAS DE PRUEBA

1. ¿Qué sucede al aumentar el valor del Skew al circuito receptor?.
2. ¿El fenomeno de la metaestabilidad se puede observar por medio de la simulacion .Justifique su respuesta?

3.7 REFERENCIAS BIBLOGRAFÍCAS

[1]F. Haim, "Estudio de tiempos de flip flops," vol. 1.0, pp. 1-3, 2007, `url:http://www.fing.edu.uy/iie/ense/asign/dislog/material/tiempos.pdf.`

[2]XILINX, "XST User Guide," vol. 10.1, pp. 1-600, `url:http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf.`

[3]L. M. Santana Gallego, "Investigación y Simulación del Clock Skew en Circuitos Integrados Modernos," Universidad de sevilla, `url:http://bibing.us.es/proyectos/abreproy/11286/fichero/Memoria.pdf.`

[4]L. J. Álvarez Ruiz de Ojeda, "Diseño Digital con Lógica Programable (Metaestabilidad)," Universidad de Vigo, 2007, `url:http://www.dte.uvigo.es/logica_programable/documentos/curso_disenho_digital_con_CDCs/Metaestabilidad.pdf.`

[5]XILINX, "Spartan-3 Generation FPGA User Guide", vol. 1.8, 2011, `url:http://www.xilinx.com/support/documentation/user_guides/ug331.pdf.`

LABORATORIO **4**

DOMINIOS DE RELOJ Y SINCRONIZADORES

Tan a destiempo llega el que va demasiado deprisa como el que se retrasa demasiado.

WILLIAM SHAKESPEARE

4.1 INTRODUCCIÓN

Gracias al fenomeno de la metaestabilidad, el funcionanmiento de un sistema digital ha quedado condicionado a tratar de reduciir la probabilidad de aparción de errores dentro de un ciruito digital, por lo tanto la utilización de tecnicas de alto nivel como el uso de circuitos sincronizadores minimiza la probabilidad de que suceda hasta el punto de considerarlo despreciable. Por consiguiente una de las principales materias de estuido es la distribución de la señal de reloj en los sistemas digitales de gran velocidad, teniendo en cuenta que las senales de reloj son muy utilizadas para definir un referencia temporal , puesto que la red de distribución de los relojes influye en la fiabilidad del sistema y en su funcionamiento.
Dentro de esta práctica se analizara e implementará circuitos sincronizadores, los cuales presenta varios casos para satisfacer distintos requisitos, comprendiendo los beneficios de mejora en un sistema digital.Por otra parte hay que tener en cuenta que aunque siempre se busca un diseño completamente síncrono, no siempre es posible, en consecuencia mediante un ejercicio se aplicarán conceptos importantes y a través de la implementación en la FPGA se dispondrán de recursos de enrutamiento especiales para las señales de reloj.

4.2 OBJETIVOS

- Comprender y analizar los conceptos de dominios de reloj y sincronizadores para su correcta utilización.
- Emplear los recursos del FPGA para administrar dominios de reloj.
- Aplicar por medio de la implementación práctica, toda la teoría vista sobre dominios de reloj y sincronizadores, teniendo en cuenta todos los parámetros y técnicas de diseños presentadas a lo largo del curso de Procesadores.

4.3 MARCO TEÓRICO

Los sistemas digitales síncronos constan de una serie de componentes (registros, lógica combinacional que interconecta los registros y una red distribución de la señal de reloj) que conforman lo que se conoce como *datapath* (ver figura 4.1)[5]. En el *datapath* los datos son almacenanados por medio de registros a la espera de la señal de reloj, para luego cuando la señal de reloj este activa el dato salga del registro y se propaga a través de la lógica combinacional, por consiguiente si el circuito funciona correctamente, el dato es capturado por el siguiente registro en el próximo flanco de reloj[5].

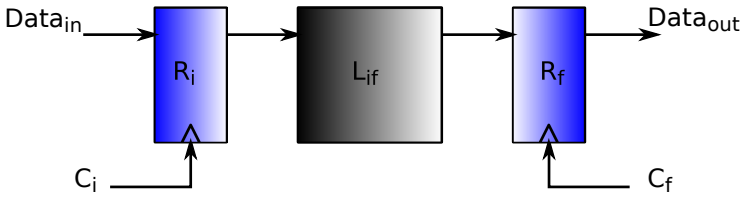


Figura 4.1: Elementos de un sistema digital síncrono.Tomado de [5]

En un sistema perfecto este comportamiento sería lo ideal, pero hay que tener en cuenta que hay diseños que no son completamente síncronos. Cuando existen diferentes señales de reloj en el sistema, esto segmenta el diseño creando los dominios de reloj (véase figura 4.2).

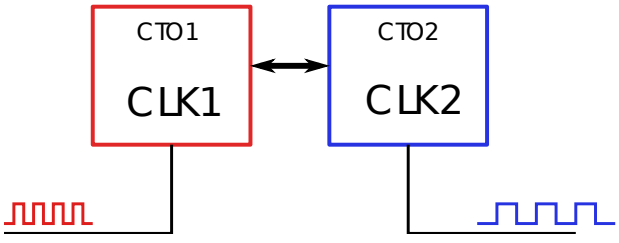


Figura 4.2: Sistema digital síncrono

La FPGA de la tarjeta SIE dispone de recursos de enrutamiento especiales conocidos como DCM (por sus siglas en ingles) para las señales de reloj, que subdivide el área para distintos dominios [1].

Si se tiene un sistema compuesto por varios circuitos que trabajan independientemente y no intercambian datos entre sí, no se presenta ningún inconveniente, pero si no es así, el principal problema es el fenómeno de metaestabilidad. Para ello se hace importante introducir circuitos sincronizadores que puedan evitar el estado metaestable (ver figura 4.3).

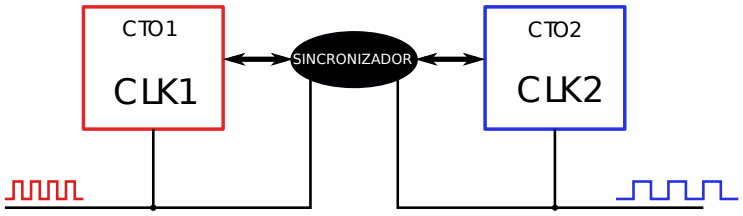


Figura 4.3: Sistema digital sincronizado

Existen varios tipos de sincronizadores: sincronizadores de señales asíncronas de un bit y los sincronizadores de varios bits, también conocidos como sincronizadores de buses.

- **Sincronizadores de señales asíncronas de un bit**

  Para sincronizar datos de un bit se utiliza varios *Flip Flops*, obteniendo un sistema más fiable, evitando el problema de metaestabilidad. En la figura 4.4 se observa el circuito sincronizador que se emplea. Cabe anotar que este sincronizador se utiliza cuando los pulsos de entrada pueden ser menor que un período de reloj de ancho.

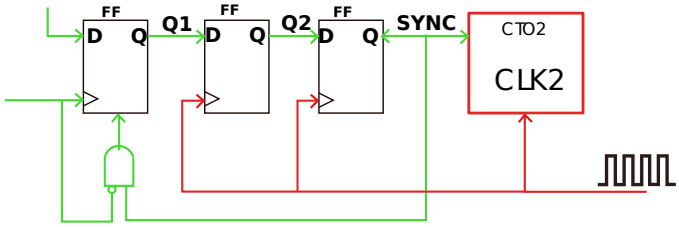


Figura 4.4: Sincronizador de señal asíncrona de un bit.

- **Sincronizadores de buses**

  En este tipo de sincronizadores se estudián varios casos según algunas características de las señales de reloj, analizando si bajo esas condiciones es seguro transferir los datos de un dominio de reloj hacia otro.

- **Caso 1 (Clk2 =Clk1)**

  En este caso el diseño es síncrono y es completamente seguro transferir los datos. No requiere de ningún circuito sincronizador.

- **Caso 2 (Clk2 =Not(Clk1))**

  Se tiene que el reloj del circuito 2 es el negado del reloj correspondiente al circuito 1 y los datos se pueden transferir siempre y cuando el periodo del circuito 1 ($\mathbf{T_{C_1}}$) sea suficientemente grande. No es necesario ningún circuito sincronizador.

- **Caso 3 (Clk2=división de Clk1)**

  Particularmente esta condición no se debe hacer, debido a que el skew de Clk1 y Clk2 puede ser muy peligroso. En este caso no es seguro transferir los datos y no se requiere ningún circuito.

- **Caso 4 (Clk2= división de Not(Clk1))**

  Generalmente los datos no son seguros para trasmitir. Debido a que el Circuito 2 puede entran en metaestabilidad. Solamente es seguro trasmitir en bajas frecuencias.

- **Caso 5 (Clk2 =Clk1 ;pero externos)**

  No se utiliza ningún circuito debido a que el software no es capaz de determinar el skew entre el Clk1 yClk2.No es seguro trasmitir los datos.

- **Caso 6 ($\mathbf{f_{Clk_2}} << \mathbf{f_{Clk_1}}$)**

  Para este caso la frecuencia de Clk2 no alcanza a solucionar el problema de metaestabilidad en el circuito 2, lo que indica que no es seguro transferir los datos y no se utiliza ningún circuito.

- **Caso 7 ($\mathbf{f_{Clk_2}} << \mathbf{f_{Clk_1}}$; Inteligente)**

  Este caso es muy particular ya que parte de la condición del caso anterior, pero con una mejora en el circuito sincronizador para eliminar la metaestabilidad con la ayuda de *Flip Flops*. Es totalmente seguro transmitir los datos y el circuito del sincronizador se muestra en la figura 4.5.

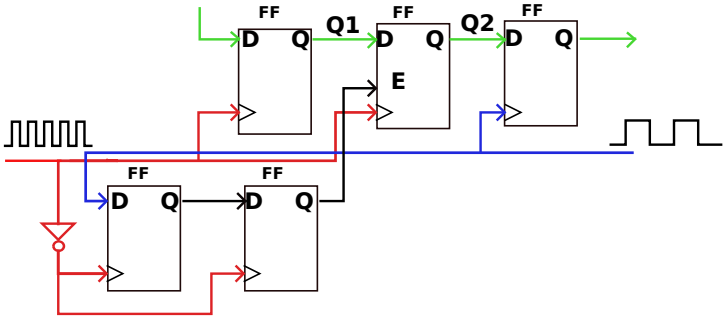


Figura 4.5: Sincronizador de buses (caso 7)

– **Caso 8 ($f_{Clk_2} >> f_{Clk_1}$;Inteligente)**

Este caso es muy similar al caso 7, con la diferencia que los datos funcionan del dominio lento al dominio rápido. Es totalmente seguro trasmitir los datos y el circuito sincronizador se muestra en la figura 4.6 .

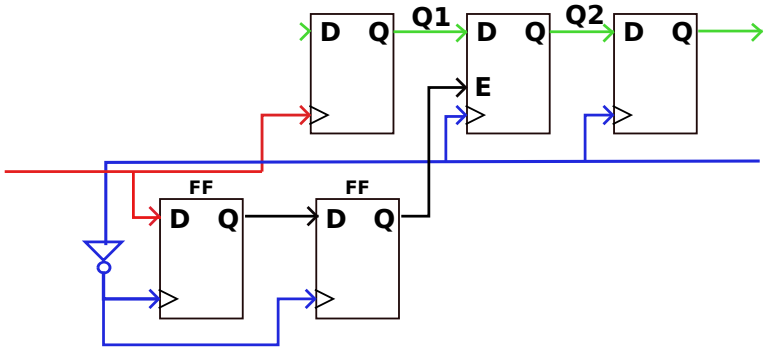


Figura 4.6: Sincronizador de buses (caso 8)

Como última opción, se puede utilizar una memoria FIFO de doble puerto, la cual se muestra en la figura 4.7 .

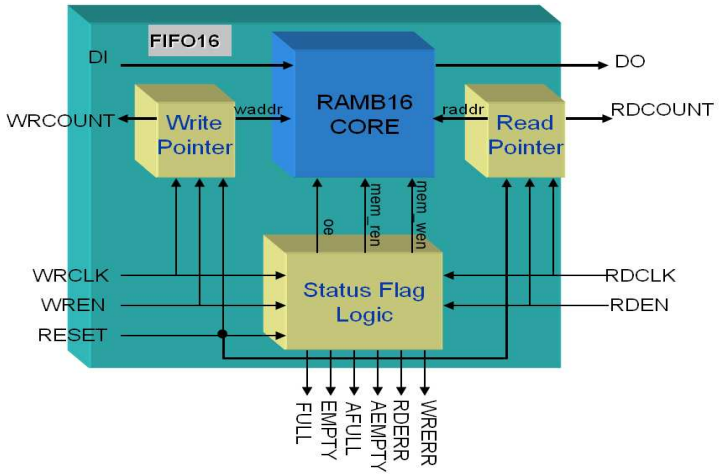


Figura 4.7: Memoria FIFO de doble puerto

4.4 DESCRIPCIÓN DEL PROBLEMA

Para esta práctica se requiere que el estudiante diseñe un sincronizador de buses para un sistema emisor y receptor de datos de esta forma. Se enviara la siguiente secuencia de datos (00FF00F) en hexadecimal a través de un circuito emisor, el cual tendrá una frecuencia de reloj **f1 MHz**. Los datos serán recibidos por un circuito receptor que trabaja a una frecuencia de reloj **f2 MHz**, el cual tiene como función enviar el dato a un Pmod siete segmentos para ser visualizado (ver figura 4.8).

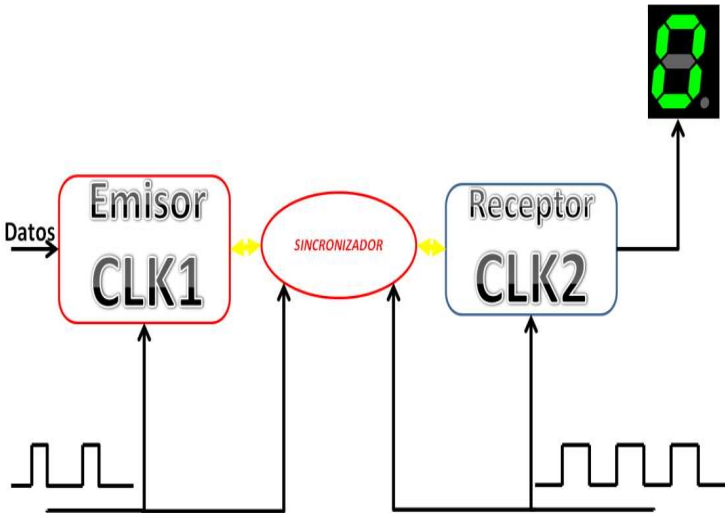


Figura 4.8: Diagrama general de la práctica

Debido a la existencia de dominios de reloj, se debe realizar un circuito sincronizador que evite la aparición del fenómeno de metaestabilidad y el proceso de recibir-enviar datos se complete de manera correcta .Con la ayuda del DCM, genere las dos señales de reloj (clk1,clk2) para los circuitos emisor y receptor de datos. Tenga en cuenta comprobar por medio de la simulación el buen funcionamiento del diseño.

Realice la descripción del hardware, eligiendo los elementos y herramientas a utilizar para el correcto funcionamiento de la práctica. Se debe diseñar el control general de la práctica, implementándolo completamente en la FPGA de la tarjeta SIE.

4.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- *RTL y Technology Schematic.*
- Códigos en VHDL

- Circuito esquemático completo que incluya todos los dispositivos a utilizar [SIE (como una caja negra), dipswitch, resistencias, sensores, pmod, etc.].

4.6 PREGUNTAS DE PRUEBA

1. ¿Cuando son necesarios los circuitos sincronizadores?
2. ¿Cómo se puede transferir datos de un dominio reloj a otro sin afectarse los datos por la metaestabilidad?

4.7 REFERENCIAS BIBLOGRAFÍCAS

[1]XILINX, "XST User Guide," vol. 10.1, pp. 1-600, `url:http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf.`

[2]J. Wakerly, Diseño Digital. PEARSON EDUCACIÓN.

[3]L. J. Álvarez Ruiz de Ojeda, "Diseño Digital con Lógica Programable (Metaestabilidad)," Universidad de Vigo, 2007, `url:http://www.dte.uvigo.es/logica_programable/documentos/curso_disenho_digital_con_CDCs/Metaestabilidad.pdf.`

[4]XILINX, "Spartan-3 Generation FPGA User Guide," vol. 1.8, 2011, `url:http://www.xilinx.com/support/documentation/user_guides/ug331.pdf.`

[5]L. M. Santana Gallego, "Investigación y Simulación del Clock Skew en Circuitos Integrados Modernos," Universidad de sevilla, `url:http://bibing.us.es/proyectos/abreproy/11286/fichero/Memoria.pdf.`

LABORATORIO **5**

PARALELISMO Y PROCESADORES SEGMENTADOS "PIPELINE"

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

ALBERT EINSTEIN

5.1 INTRODUCCIÓN

Desde hace muchos años hemos venido observando un enorme crecimiento en las capacidades y el rendimiento de los sistemas electrónicos digitales. Estos avances se deben al constante cambio tecnológico y arquitectural. Actualmente el diseño de procesadores segmentados ha sido incorporado por medio el término "*Segmentación*" a la informática de hoy en dia , por lo cual conviene de manera inmediata que el estudiante de ingeniería electrónica se familiarizarse con el mismo. los cambios tecnológicos permiten un aumento en las capacidades de los circuitos, como por ejemplo incrementar las velocidades en los relojes e incrementar la velocidad con la que realizan las funciones de un circuito. El diseño de un programa paralelo tiene que considerar entre otras cosas, el tipo de arquitectura sobre la cual se va a ejecutar el programa, las necesidades de tiempo y espacio que requiere la aplicación. Surgiendo una pregunta muy importante a tener en cuenta, ¿Cómo aumentar la velocidad del procesador?, apareciendo dos respuestas:

- Construir circuitos mas rápidos pero ¿a qué precio?
- Concurrencia: nivel del procesador (Paralelismo) y nivel de instrucciones (*Pipelining*).

Dicha velocidad de ejecución de los programa depende de muchos factores, haciendo mas rápidos los circuitos con la que se construye los procesadores, claro está considerando el costo que supone una mejora

y que el limite a esta velocidad lo impone el estado del arte actual de la tecnología. Otra posibilidad y es la que se verá en esta práctica la cual consiste en organizar el hardware para poder ejecutar simultáneamente más de una instrucción a fin de aplicar conceptos de optimización de los procesos como latencia, *throughput*, máxima frecuencia de trabajo, paralelismo temporal y espacial.

5.2 OBJETIVOS

- Conocer las principales características del paralelismo y los procesadores segmentados comprendiendo y utilizando de manera correcta los conceptos: paralelismo espacial y paralelismo temporal.
- Encontrar las relaciones entre los diferentes tipos de paralelismo espacial y temporal, aplicando los conceptos de *troughput* y la latencia.Precisando como afectan las mejoras en el trouhgput y la latencia a un sistema propuesto.
- Implementar mejoras en un sistema utilizando cualquiera de los tipos de paralelismo propuesto teóricamente.

5.3 MARCO TEÓRICO

En nuestros dias la gran mayoría de procesadores de propósito general tienen relojes cuya frecuencia alcanza valores más elevados que los que pueden alcanzar los de un FPGA. Sin embargo las FPGA pueden realizar tareas como si tuviera configurado varios procesadores trabajando en paralelo, al contrario de los procesadores de propósito general que solo pueden realizar una instrucción por cada ciclo de reloj[3].
Por otra parte se han realizado muchas definiciones del significado de paralelismo, pero generalmente se define como:
la acción de realizar o ejecutar varias instrucciones o sentencias de programa en un mismo período de tiempo, de manera que su verdadero significado la acción de realizar más instrucciones en menos tiempo [3].

Tenemos dos grandes tipos de paralelismo que son:

- **Paralelismo espacial:** consiste en la ejecución simultánea de las tareas mediante varias unidades de procesamiento. En un instante dado, esas unidades pueden estar ejecutando una misma tarea (o instrucción) o tareas diferentes. Este tipo de paralelismo puede definirse también como MIMD (*Multiple Instruction stream, Multiple Data stream*)[3].
- **Paralelismo temporal:** también conocido como tunelizado hace referencia a la ejecución de una tarea como una cascada de sub-tareas. Existe aquí una unidad de procesamiento para llevar a cabo

cada sub-tarea. Todas las unidades pueden trabajar al mismo tiempo de manera que se solapan las tareas. Puede definirse también como SIMD (*Single Instruction stream, Multiple Data stream*)[3].

5.3.1 Procesadores segmentados

La segmentación es una técnica de implementación de procesadores que desarrolla el paralelismo a manera que se ejecuten varias acciones al mismo tiempo. Además por medio de los procesadores segmentados se puede realizar tareas simultanemente ejecutando múltiples instrucciones[1]. El procesamiento segmentado utiliza la fabricación en cadena la cual consiste en que cada etapa de la segmentación (o segmento) completa una sub-tarea de la tarea total. Teniendo en cuenta que la salida de cada segmento es la entrada del otro, formando una especie de cause donde se procesa la tarea[1].
Una gran ventaja de la segmentación es la posibilidad de iniciar una nueva tarea sin la necesidad de que la anterior haya finalizado. *La medida de la eficacia de un procesador segmentado no es el tiempo total transcurrido desde que se comienza una tarea hasta que se finalice, denominado tiempo de latencia de un procesador, sino el tiempo máximo que puede pasar entre la finalización de dos tareas consecutivas*[1].

Ahora bien considere un ejemplo de un procesador segmentado y otro que no está segmentado, en este caso tenemos una tarea compuesta por sub-tareas en donde las sub-tareas se procesan de forma secuencial, el tiempo necesario para procesar la tarea total será la suma de los tiempos necesarios para la terminación de cada una de las sub-tareas (véase la figura 5.1)[1].

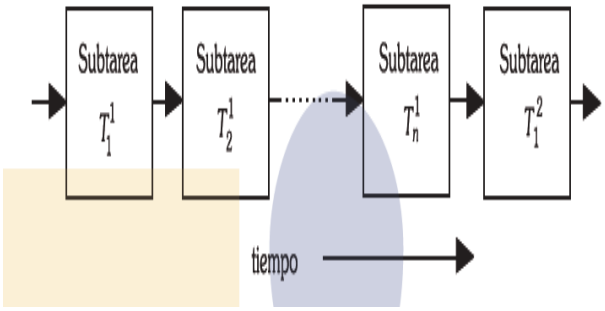


Figura 5.1: Tarea procesada de forma secuencial (no segmentado).Tomada de [1]

Por otra parte si para procesar esa misma tarea se utiliza un procesador segmentado, solamente es necesario que la primera sub-tarea termine para poder comenzar a procesar una nueva tarea (véase la figura 5.2)[1].

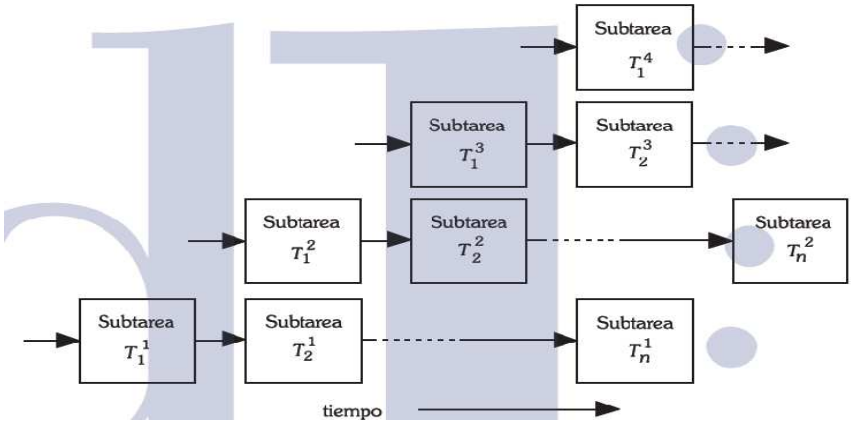


Figura 5.2: Tarea ejecutada mediante un procesador segmentado.Tomada de [1]

Como se observa en la figura 5.2, las tareas tiene un continuo flujo, que se procesa a través de los η segmentos encargados de procesar cada una de las sub-tareas, el tiempo total de procesamiento de una tarea no cambia, lo verdaderamente importante es el ritmo con que salen del procesador las tareas, esto se denomina velocidad de emisión de tareas y η más conocido como profundidad de segmentación[1]. Cabe aclarar que para que el tiempo de latencia no cambie es necesario que el procesador este equilibrado, es decir, que todas sus sub-tareas tarden el mismo tiempo en completarse[1].
Por otra parte para entrar en materia de rendimiento del procesador definiremos lo que se conoce como productividad de un procesador o también llamada ***throughput***.
Se define productividad de un procesador como *el número de tareas que puede completar por unidad de tiempo*[1]. Siendo esta una medida absoluta y mide la potencia global de cálculo del procesador[1]. Podremos calcular el *throughput* dividiendo el numero de tareas emitidas por el tiempo empleado en procesarlas[1].

Como vemos, hay diversos factores que podemos aprovechar para poder optimizar un proceso. Trataremos este concepto y la técnica para poder utilizarlo de la mejor manera.

5.4 DESCRIPCIÓN DEL PROBLEMA

En esta práctica se pretende introducir al estudiante en el manejo de paralelismo y procesadores segmentados, con este fin se plantea un ejercicio práctico de la siguiente manera:

Se deben generar dos números aleatorios A y B de n bits a una frecuencia de f MHz, los cuáles se utilizarán como las entradas de un multiplicador. El cual se debe implementar aplicando los conceptos acerca de pipeline para lograr un sistema eficiente.
Para la optimización del sistema se deben tener en cuenta la máxima frecuencia de trabajo de los circuitos, así como el throughput y la latencia.Tambien el diseño de un circuito que genere dos números aleatorios,al igual que el multiplicador en VHDL, teniendo como guia del diseño la información acerca de Pipeline en la referencia bibliografica[5](pag 135 -142). Luego complete la tabla 5.1 ,donde se deben introducir los valores de frecuencia maxima, throughput y latencia, concluyendo cual es el numero de etapas para que el sistema sea el más eficiente.

Etapas de pipeline	**Fmax**	**Latencia**	**troughput**
1	[MHz]	[ns]	[MHz]
2			
3			
4			

Tabla 5.1: valores caracteristicos del multiplicador

NOTA: Los valores del número de bits n y de la frecuencia f serán indicadas por el profesor.

5.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- Diagrama de bloques del circuito completo.
- Códigos en VHDL
- Diagramas *RTL y Technology Schematic*.
- Tabla con 4 o más valores diferentes de segmentación del circuito multiplicador indicando el throughput y la latencia en cada caso.
- Explicación de qué parámetros se tuvierón en cuenta para la escogencia del número de niveles de segmentación.

5.6 PREGUNTAS DE PRUEBA

1. ¿Que diferencia exite entre paralelismo temporal y espacial.Justifique su respuesta?.
2. ¿Cómo podemos medir la latencia en un sistema digital?.

5.7 REFERENCIAS BIBLOGRAFÍCAS

[1]J. Bastida Ibáñez, "Procesadores Segmentados,"Universidad de Valladolid, url:http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Segment.pdf.

[2]E. M. Tordera, "Segmentación y procesadores segmentados," Universidad politécnica de Cataluña, url:http://studies.ac.upc.edu/EUPVG/ARCO_I/tema2.pdf.

[3]J. A. Somoza Chuay and J. Hernández Palancar, "Estado actual de la paralelización de algoritmos utilizando FPGAs para minería de texto," Ciudad de la Habana, 2010, url:http://www.cenatav.co.cu/doc/RTecnicos/RT%20SerieGris_014web.pdf.

[4]Paco Aylagas Romero, "Arquitectura de Computadores," Universidad Politécnica de Madrid, pp. 1-29, url:http://www.dia.eui.upm.es/asignatu/arq_com/Paco/6-Pipeline.pdf.

[5]XILINX, "XST User Guide," vol. 10.1, pp. 1-600, url:http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf.

LABORATORIO **6**

DISEÑO DIGITAL AVANZADO "UNIDAD DE CONTROL Y DATAPATH"

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

ALBERT EINSTEIN

6.1 INTRODUCCIÓN

Hace algunos años era muy difícil realizar diseños digitales complejos, sin embargo el avance de la tecnología en la actualidad permite desarrollar diseños complejos con mayor facilidad y a un muy buen costo, con la ventaja de crear diseños a la medida de una aplicación con la ayuda de herramientas como las tarjetas programables (FPGAs).

Por otra los procesadores son los más utilizados para diferentes aplicaciones en el campo de los sistemas digitales, es por eso que centros academicos buscan que sus estudiantes de ingeniería electrónica se inicien en el estudio de diseños de procesadores específicos. Existen técnicas de aprendizaje que buscan facilitar la teoría de diseño de procesadores especificos, por medio de la utilizanción del lenguaje de programacion VHDL y el uso de algoritmos de control , diagramas ASM y el *datapath*, los culaes permiten optimizar los diferentes parámetros de diseño, empleando de manera correcta los recurso lógicos y eliminado los problemas de *glitchs* y metaestabilidad.

En esta práctica se coloca en contexto todo el conocimiento teórico sobre diseñar de manera correcta la unidad de control y el *datapath*, de tal manera que se aplique cualquiera de los dos métodos utilizados en el transcurso de la asignatura para diseñar procesadores de proposito especifico.

6.2 OBJETIVOS

- Identificar y comprender los diferentes tipos de procesadores que se presentan en el diseño digital avanzado.
- Conocer las principales técnicas, para facilitar el diseño de procesadores de propósito específico.
- Implementar un procesador de propósito específico(unidad de control y *datapath*) por medio del lenguaje de descripción de hardware, aplicando algunas de las metodologías de diseño vistas en clase.

6.3 MARCO TEÓRICO

EL diseño digital está compuesto por una unidad de control y un *datapath*, en donde el *datapath* se divide en dos tipos de puertos de entrada y salida, un puerto que controla los datos y el otro es un puerto que maneja señales de control. Para desarrollar esta funcion cuenta con una serie de unidades funcionales, por ejemplo $ALUs$ o multiplicadores, donde realiza procesamiento de datos y operaciones[2].
La mayoría de los procesadores tiene como caracteristica dedicar una gran parte de la unidad de control a regular la interacción entre el *datapath* y la memoria (ver figura 6.1)[2].

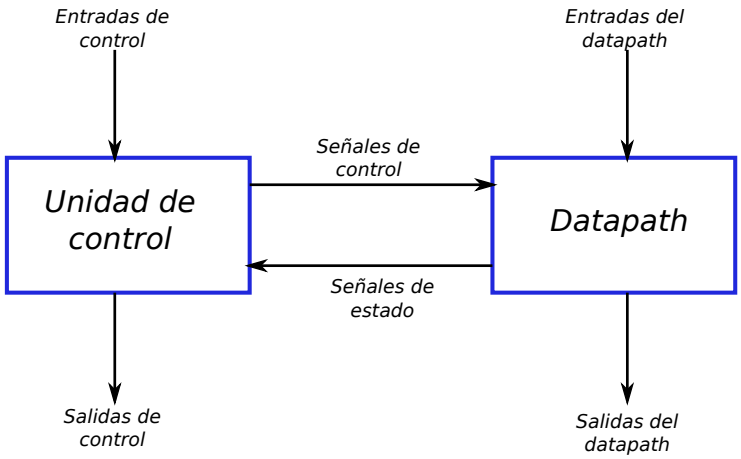


Figura 6.1: Unidad de control y datapath.Tomada de [2]

Por otra parte la unidad de control tiene dos tipos de señales de entrada, las externas y las de estado, las cuales representan los estados del *datapath*. Además también contiene dos tipos de señales de salida, las externas y las de control del *datapath* [2].
Para poder manejar estas señales son necesarias las FSMD, las cuales se entienden como la definición más amplia de las máquinas de estados finitos donde se introducen un conjunto de variables:

- **Entradas y salidas de los *datapath***
- **Un conjunto de entradas, salidas y estados de FSM.**

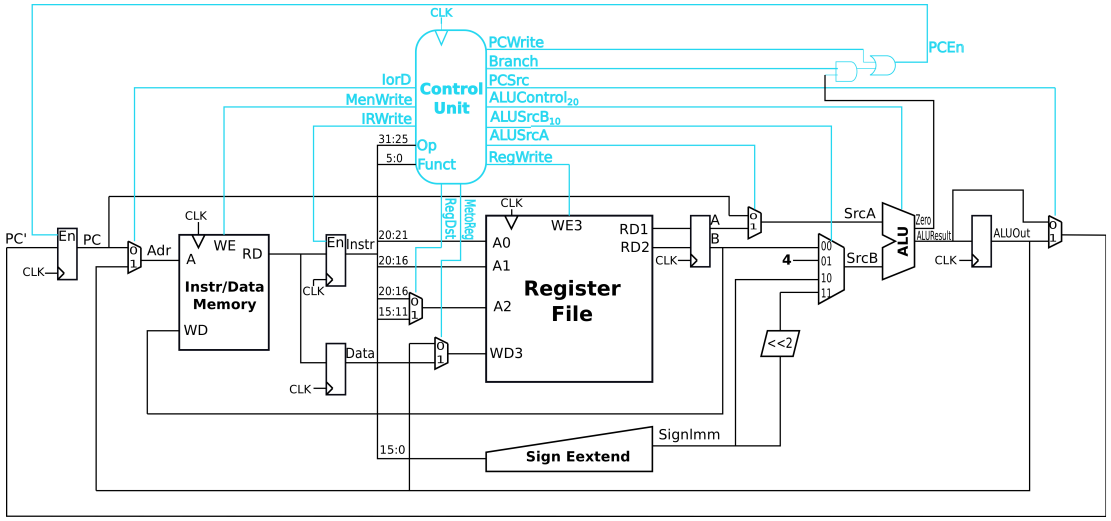


Figura 6.2: *Datapath.*Tomada de [2]

6.3.1 Organización de sistemas

La operación del sistema es sincronizada por una señal de reloj, la cual logra que en cada ciclo de reloj, el subsistema de control envie uno o más registros, de tal manera que para lograr la conexión deseada entre los registros y los agentes que intervienen en una transferencia de registros, se necesite del control del subsistema de datos.

Por otra parte teniendo en cuenta el tiempo de cada ciclo de reloj se determina la longitud de la ruta, desde el registro fuente al registro destino, incluyendo las transformaciones de los datos en el recorrido [2].

Para entender un poco más esta organización hay que clasificarlas de la siguiente forma :

- **Unidades funcionales u operadores**
- **Unidades de control**

Las estructuras de las unidades funcionales son:

- **Sistema no compartido**
- **Sistema compartido**
- **Sistema unimodal**

Y las correspondientes a la estructura de las unidades de control están conformada por :

- **Control centralizado**
- **Control descentralizado**
- **Control semi-centralizado**

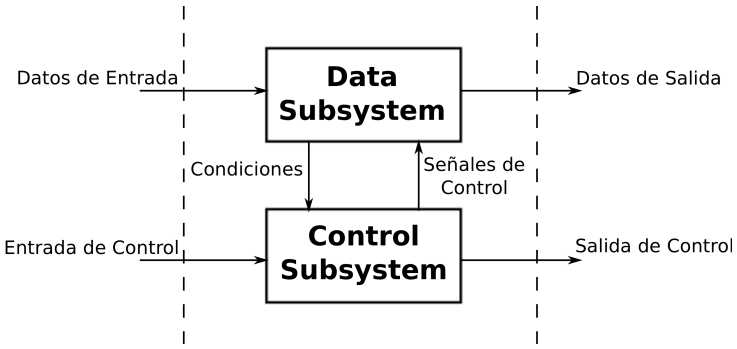


Figura 6.3: Unidades de datos y control.Tomada de [2]

6.4 DESCRIPCIÓN DEL PROBLEMA

En esta práctica se pretende introducir al estudiante en la metodología de diseño de procesadores específicos. Con este fin se planta un ejercicio de la siguiente manera:
Una sucesión matemática es una aplicación definida sobre los enteros naturales, es decir que para cada natural (0, 1, 2, etc.) existe un elemento correspondiente en la sucesión.
En matemáticas los números de Perrin están definidos por la relación de recurrencia:

Perrin(0)=3
Perrin(1)=0
Perrin(2)=2
Perrin(n)=Perrin(n-2)+Perrin(n-3); n>2

Por ejemplo, los 14 primeros números de Perrin son: 3, 0, 2, 3, 2, 5, 5, 7, 10, 12, 17, 22, 29, 39, donde cada término a partir del cuarto fue calculado utilizando lo tres elementos anteriores y la fórmula de Perrin.
Ahora bien, lo interesante de los números de Perrin es que si Perrin(n) (el enésimo número de Perrin) es divisible entre n, entonces n es un número Primo de Perrin.
Por ejemplo, si vemos la posición 11 (Perrin (11)) de nuestra serie, encontramos el número 22, el cual es divisible entre 11; lo anterior indica que 11 es Primo de Perrin.

Diseñe un procesador de propósito específico que reciba como parámetro un número n y retorne el número de Perrin de n. Por ejemplo, si recibe el número 10, debe retornar el número de Perrin de 10, es decir 17 y decir si el número es primo de Perrin en caso de no serlo que muestre el próximo primo de Perrin.
Tenga en cuenta Realizar todos los pasos para el diseño del procesador de propósito general por cualquiera de las dos técnicas vistas en clase y la descripción del hardware, eligiendo los elementos y herramientas a utilizar para el correcto funcionamiento de la práctica. Se debe diseñar el control general de la práctica, implementándolo completamente en la FPGA de la tarjeta SIE.

6.5 INFORME DE LABORATORIO

Se deberá incluir dentro del informe de laboratorio:

- Diagrama de Estados y el *datapath*
- *RTL y Technology Schematic*.
- Códigos en VHDL
- Simulación de los módulos descritos que validen el diseño implementado
- Circuito esquemático completo que incluya todos los dispositivos a utilizar [SIE (como una caja negra), dipswitch, resistencias, sensores, pmod, etc.].

6.6 PREGUNTAS DE PRUEBA

1. ¿Cuales son las diferencias entres los procesadores de proposito especifico y general?
2. ¿Dentro del diseño digital avanzado,que tipo de procesador es mejor utilizar para un tarea determinada?
3. ¿Qué elementos contiene un Procesador. Enumerelos y de una breve explicacion de cada uno de ellos?

6.7 REFERENCIAS BIBLOGRAFÍCAS

[1]C. A. Fajardo and J. H. Ramón, Introducción al Diseño Digital Avanzado, 1.1 ed. Universidad Industrial de Santander, 2010.

[2]R. Aguilar Ponce, “El procesador: Datapath y la unidad de control” Universidad Autónoma de San Luis Potosí, 2012,
`url:http://galia.fc.uaslp.mx/~rmariela/arquitectura/unidad51.pdf`

ANEXO B: PROYECTO FINAL DE LA ASIGNATURA "PROCESADORES"

PROYECTO 1

SISTEMA DE TRATAMIENTO DE EFECTOS DE AUDIO

El trabajo del pensamiento se parece a la perforación de un pozo: el agua es turbia al principio, mas luego se clarifica..

PROVERBIO CHINO

1.1 INTRODUCCIÓN

El proyecto final consiste en construir un sistema de tratamiento de efectos de audio, el cual manejará diferentes efectos como por ejemplo: Delay, Chorus, Flanger y Wah. Se diseñará un módulo de control que funcione como interfaz entre un dispositivo códec y la FPGA de la tarjeta SIE. Mediante un teclado ps/2 el usuario interactúa con el circuito eligiendo el efecto que se desea implementar por medio de la pulsación de una tecla. Debido a que se necesita almacenar una gran cantidad de muestras de audio para realizar los efectos, es necesario interactuar con una memoria externa SRAM que se conecte directamente a la FPGA. En la figura 1 se muestra la composición externa del sistema de tratamiento de efectos de audio. En ella podemos apreciar que el sistema está compuesto por varios dispositivos, a los cuales se le deben realizar las adecuaciones de hardware necesarias para su funcionamiento.

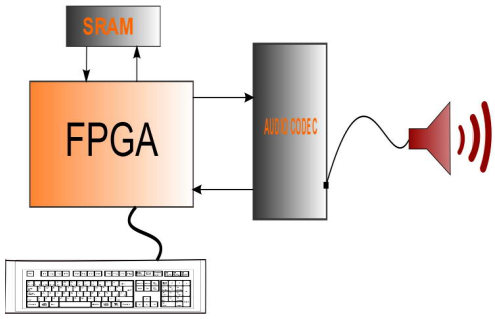


Figura 1.1: Diagrama general del proyecto.Los autores

1.2 OBJETIVOS

- Diseñar e implementar los módulos necesarios para la construción de un sistema de tratamiento de efectos de audio.
- Aplicar las temáticas de la asignatura "procesadores" por medio del diseño e implementación del proyecto final.
- Realizar la construcción del hardware necesario para la implementación del proyecto final por medio de la utilización de la tarjeta SIE.

1.3 MARCO TEÓRICO

Para el desarrollo del proyecto se utiliza la FPGA x3csvq100 de tarjeta SIE y los dispositivos que se enumeran a continuación.

1.3.1 Códec

El códec es un dispositivo que codifica una señal analógica en una señal digital y al contrario, decodifica una señal digital en una analógica. En otras palabras, cuenta con un convertidor analógico-digital (ADC) y un convertidor digital a analógico (DAC) que utilizan el mismo reloj[1].

1.3.2 Memoria SRAM

Tipo de memoria RAM que retienen su contenido el tiempo que reciben energía, a diferencia de las RAM (DRAM) que necesitan refrescarse periódicamente (SRAM[1] no debe ser confundida con las ROM y las memorias flash, dado que es una memoria volátil y preserva los datos sólo cuando recibe energía). Es más rápida y más fiable que las más comunes DRAM. SRAM tiene un tiempo de acceso de 10 nanosegundos,

[1] SRAM: Static random access memory , Memoria de acceso aleatorio estática

en cambio en las DRAM es de 60 nanosegundos. Además, su ciclo es mucho más corto que el de las DRAM porque no necesitan una pausa entre accesos. Una memoria SRAM tiene tres estados distintos de operación: standby, en el cual el circuito está en reposo, reading o en fase de lectura, durante el cual los datos son leídos desde la memoria, y writing o en fase de escritura, durante el cual se actualizan los datos almacenados en la memoria[2].

1.3.3 Teclado ps/2

Un teclado tipo PS2 utiliza para comunicarse con un sistema un protocolo bidireccional serie síncrono de once bits. Los puertos utilizados para la comunicación con el dispositivo son:

- CLK: Reloj de sincronización, generado por el dispositivo ha de tener una frecuencia entre 10 y 30 KHz.
- DATA: Puerto para la transmisión de datos serie.

La transmisión de datos utiliza un protocolo síncrono de once bits el cual consiste en un bit de inicio START (0), Ocho bits de datos (Primero el LSB), un bit de paridad impar y por utlimo un bit de parada STOP (1)[1].

1.4 DISEÑO DE MÓDULOS

En la siguiente sección se presenta una mayor información acerca de cada módulo para llevar a buen término el desarrollo del proyecto final de la asignatura “procesadores”.
Inicialmente son 11 módulos propuestos para el desarrollo del sistema de tratamiento de efectos de audio, pero se deja a libre elección la creación de mas módulos que le den un mejor resultado a la implementación del diseño.
En la figura 2 se muestra el diseño modular concerniente a la arquitectura interna del sistema, cada módulo es representado por una caja con su respectivo nombre y las señales de entrada y salida que los comanda.

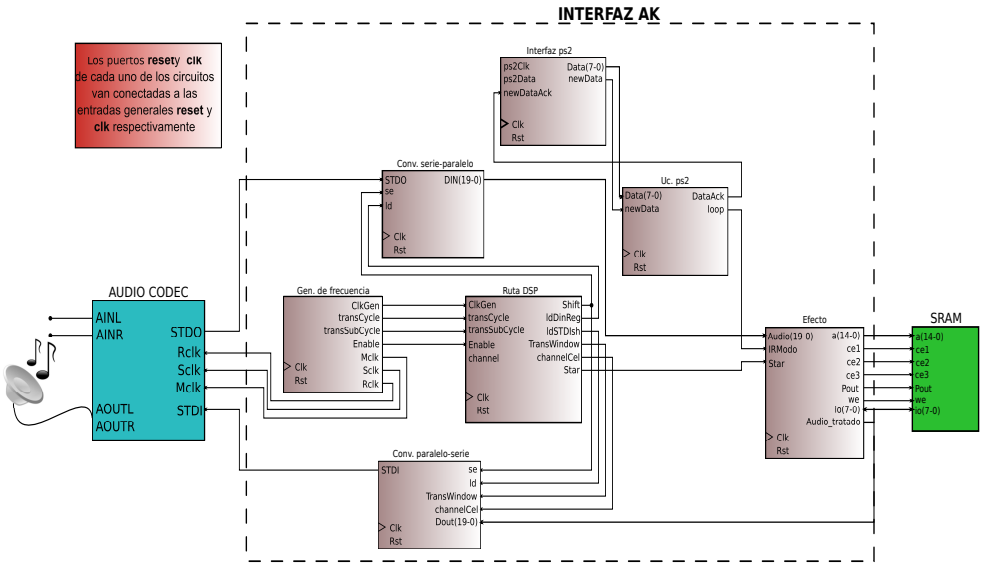


Figura 1.2: Diseño modular. Los autores

A continuación se describe la función principal de cada uno de los 11 módulos del sistema de tratamiento de efectos de audio.

- Interfaz Códec: Es el modulo principal del proyecto, dónde se conectan todos los módulos implicados de manera conveniente para poder realizar la elección de los efecto por medio del teclado.
- Interfaz PS2 : Es el encargado de servir de conexión entre el módulo que realiza los efectos de audio y el teclado, para que el usuario interactué eligiendo el efecto que desea implementar pulsando una tecla determinada
- Unidad de control PS2: Este modulo tiene como finalidad configurar el teclado PS2 para ser conectado en la FPGA de la tarjeta SIE.
- Conversor serie/paralelo: Este módulo se encarga de recibir los bits que componen una muestra de la señal que el códec acaba de discretizar.
- Conversor paralelo/serie: Este módulo toma la muestra tratada y la descompone en bits para realizar la transmisión serie bit a bit al códec y que éste la transforme en una señal analógica.
- Generador de frecuencias: Genera las señales necesarias, con el fin de controlar el a códec y se realicen las conversiones análogo/digital y digital/análogo correctamente.
- Interfaz con SRAM: Este módulo es el encargado de realizar la comunicación entre la FPGA y la memoria SRAM.

- Módulo distribución de efectos: Es el módulo encargado de aplicar los distintos efectos sobre la señal de audio de entrada.
- Efect_Delay: Esta entidad se encarga de almacenar las muestras de audio en la memoria RAM y tratar ésta como una cola FIFO genérica.
- Lfo[2]: Este módulo implementa un oscilador triangular de baja frecuencia. Toma como entrada un tiempo mínimo y máximo de persistencia y lo modula como señal triangular. Su salida la utiliza el módulo Delay como tamaño de la cola.
- Efect_Wah: Este módulo es el encargado de simular el efecto Wah por medio de incrementos y decrementos cíclicos del volumen de la muestra original.

1.5 INFORME

Se deberá incluir dentro del informe :

- Diagrama de bloques del sistema de tratamiento de efectos de auido
- Esquematico de la placa utilizada para la construcción del hardware.
- Códigos en VHDL de cada uno de los módulos
- Circuito esquemático completo que incluya todos los dispositivos a utilizar [SIE (como una caja negra), dipswitch, resistencias, sensores, pmod, etc.].

1.6 REFERENCIAS BIBLOGRAFÍCAS

[1]J. D. Hernández, C. Vásquez Blanco, and E. Duque López, "PROCESADO DE EFECTOS DE AUDIO BAJO FPGAs," 2002.

[2] `url:http://www.alegsa.com.ar/Dic/sram.php`

[2] LFO : low frequency oscillation ; Oscilación de baja frecuencia

ANEXO C: MANUAL DE USUARIO DE LA TARJETA SIE: PROCESADORES

MANUAL DE USUARIO DE LA PLATAFORMA SIE (PROCESADORES)

1.1 Introducción

En el transcurso de la asignatura de Procesadores se utilizaran herramientas importantes para el desarrollo de las prácticas a implementar. Principalmente manejar con destrezas ciertos elementos gráficos que nos brinda el entorno de diseño Xilinx: el ISE WebPack[1].

Este manual es una guía práctica para aprender a utilizar herramientas como: simulación *Post-Route*, diagramas *RTL* y *Technology Schematic*. Mediante el uso del dispositivo lógico programable (FPGA) contenido en la tarjeta SIE. Este manual se ha desarrollado en la escuela de ingeniería eléctrica, electrónica y telecomunicaciones de la universidad industrial de Santander para las prácticas de la asignatura Procesadores. Previamente, los estudiantes han cursado la asignatura de sistemas digitales, donde adquirieron conocimientos básicos sobre los circuitos digitales y han realizado diseño mediante el uso de VHDL y dispositivos lógicos programables. Las prácticas realizadas con FPGA de sistemas digitales están guiadas en el manual de la misma asignatura. En dichas prácticas se enseña a diseñar circuitos electrónicos digitales con esquemáticos y la FPGA de la tarjeta SIE. Por tanto para comprender mejor el manual de la asignatura Procesadores se recomienda tener conceptos básicos de los sistemas de numeración y electrónica digital: diseño con puertas lógicas, bloques combinacionales, elementos de memoria, registros y contadores.

Además de la guía de cómo utilizar ciertas herramientas, este manual contiene información sobre la tarjeta SIE: Periféricos, pines y dispositivos. También se ha añadido el procedimiento de cómo acceder al procesador de la tarjeta SIE, para poder implementar un diseño de hardware en la FPGA.

[1] Herramienta gratuita que se puede descargar en: http://www.xilinx.com/webpack/classics/wpclassic/index.htm

1.2 TARJETA SIE

La tarjeta SIE fue diseñada por un investigador de la Universidad Nacional de Colombia. Creada por la empresa emQbit LTDA para atender a los desarrolladores de hardware. Esta plataforma carece intencionalmente de elementos comúnmente utilizados en las plataformas de desarrollo de PLDs como pulsadores, leds, displays o conectores especiales para obligar al estudiante a crear sus propios periféricos. La tarjeta SIE contiene una FPGA de Xilinx de modelo Spartan-3E XC3S100 con encapsulado VQG100.Tambien diferentes especificaciones que serán nombradas a continuación.

- Procesador Ingenic JZ4725.
- Hasta 64 MB de memoria SDRAM.
- 2 GB de memoria NAND2 GB de memoria NAND.
- 30 pines de propósitos generales entrada / salida de señales digitales (rango 0-3.3V).
- 10 canales analógicos al adaptador digital. Rango de entrada analógica de 0 - 3,3V
- Puerto USB del dispositivo
- Puerto Micro SD.
- Puerto I2C.
- Puerto para conecta el UART Serie RS-232

La capacidad de la FPGA de SIE permite la implementación de procesadores *soft-core* (como plasma y microblaze), posibilitando la implementación de tareas Hardware y Software en ella.

la figura 1.1 muestra la placa SIE, y se señalan varios de sus componentes. La foto de la placa no coincide exactamente con la versión de la placa que tenemos en el laboratorio y hay algunos componentes que no están en el mismo lugar, aunque sí todos los que se han señalado.

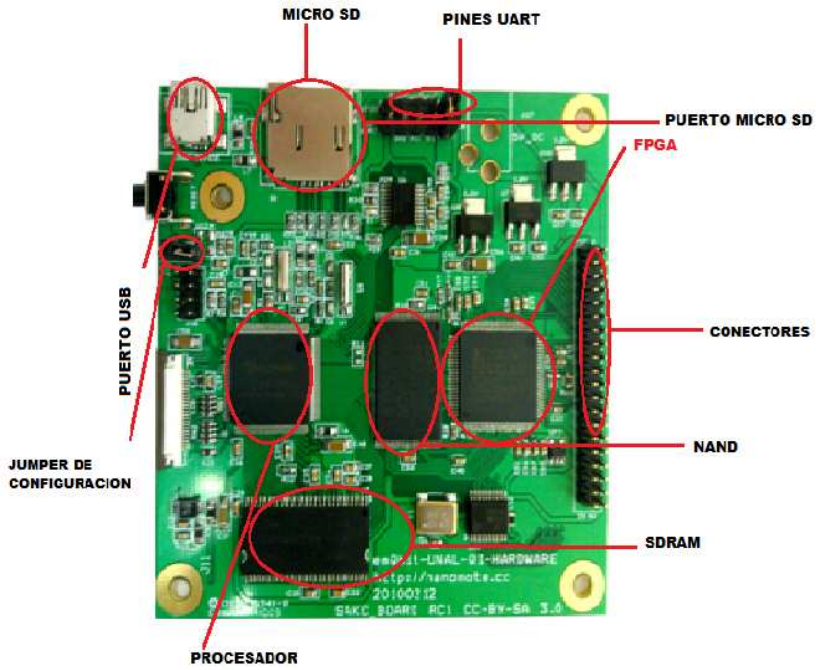


Figura 1.1: Placa SIE de *emQbit LTDA*

A medida que se avance con las prácticas de procesadores iremos aprendiendo la funcionalidad de la tarjeta SIE. Aún así, si tienes curiosidad, en la página web de *emQbit LTDA* puedes consultar todo acerca de la plataforma SIE [2] .

1.3 Diagramas RTL y Technology Schematic.

Aunque es una herramienta muy poderosa para poder comprender mejor nuestro diseño, hay muchos estudiantes que lo pasan por alto. Por ese motivo se plantean pasos para poder obtener los diagramas *RTL* y *Technology Schematic* que nos ofrece el entorno de diseño de Xilinx.

El diseño de los circuitos de las prácticas serán desarrollados bajo el lenguaje de descripción de hardware VHDL, el cual nos permite realizar diseños avanzados de manera más rápida y eficiente y la herramienta *ISE-WebPack* de Xilinx. Habiendo creado el proyecto, colocándole todas las especificaciones necesarias y las líneas de comando para darle forma al código VHDL, los siguiente es proceder a realizar la *"Synthesize"* de nuestra entidad.

En la ventana de procesos, se ordenan los pasos que se toman: *"Synthesize-XST"* , *"Implement De-*

[2] http://en.qi-hardware.com/wiki/SIE

sign" y *"Generate Programming File"*.

Así que en ese orden el primer paso es sintetizar la entidad, lo cual se logra haciendo doble clic en *"Synthesize-XST"* y luego esperar hasta que veamos que se pone una signo verde de correcto (puede colocarse en amarillo indicando que hay advertencias -*warnings*- que a veces no son importantes). Cuando se haya realizado correctamente este paso, la subventana de procesos mostrara el aspecto de la figura 1.2.

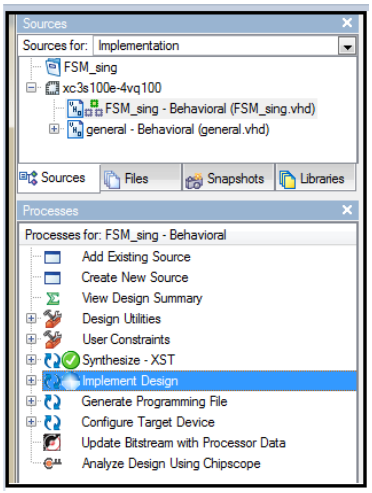


Figura 1.2: Aspecto de procesos una vez sintetizado.

Ahora teniendo sintetizada la entidad, procedemos a obtener el diagrama RTL y el *Technology Schematic*.

Lo primero es situarnos en el símbolo de mas (⊞) que se encuentra a la izquierda de *"Synthesize-XST"* y le damos click, de esta forma desplegamos el menú que está conformado por: *"View Synthesize Report"*, *"View RTL Schematic"* y *"View Technology Schematic"*. En la figura 1.3 se muestra la ventana con el menú desplegado.

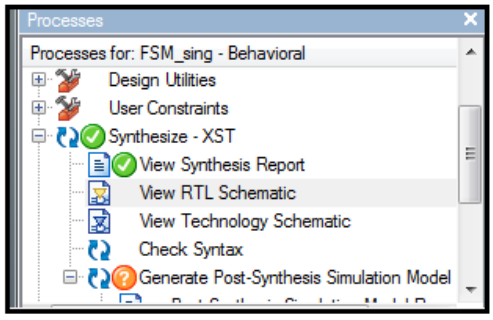


Figura 1.3: Menú desplegado de la Synthesis.

Después de tener el menú desplegado, damos doble clic en *"View RTL Schematic"*, y automáticamente

apareceré en la ventana principal una caja con las entradas y salidas asignadas en la entidad del diseño, en esa instancia ya se encuentra observando el RTL, si le damos doble clic a la caja mostrada nos aparece el diagrama más detallado los cual nos permite visualizar todos las trayectorias y elementos utilizados en su circuito digital(Véase figura 1.4).

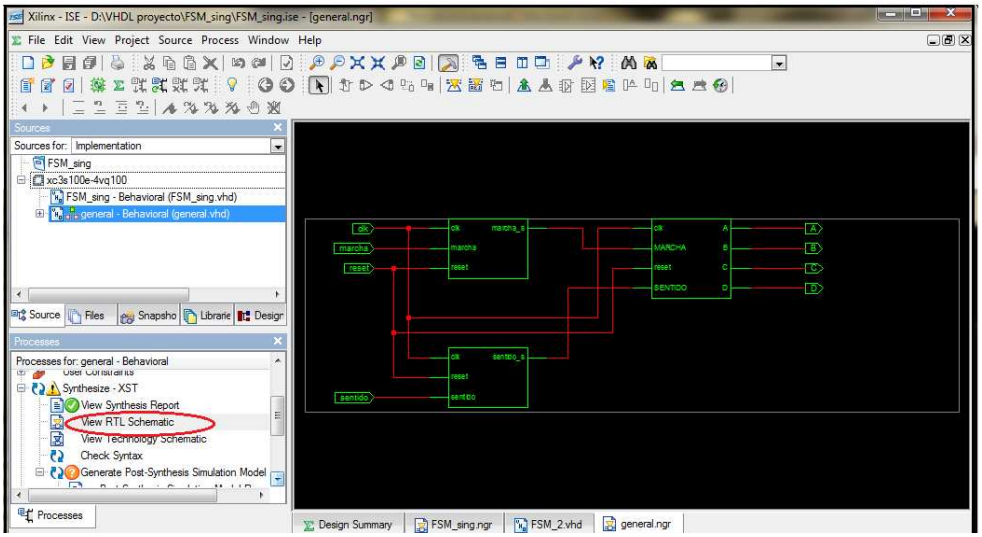


Figura 1.4: *RTL Schematic* de la entidad.

Ahora lo siguiente es obtener el *"Technology Schematic" , haciendo doble clic sobre "*View Technology Schematic" se muestra en la ventana principal una caja que es similar al RTL, si pulsamos doble clic encima, obtenemos el Technology, el cual es un poco más complejo que el RTL ya que está conformado por recursos de la FPGA (*LUT'S, Mux ,Flip Flop, ALU,* etc) en la figura 1.5 se muestra el *"Technology Schematic"*.

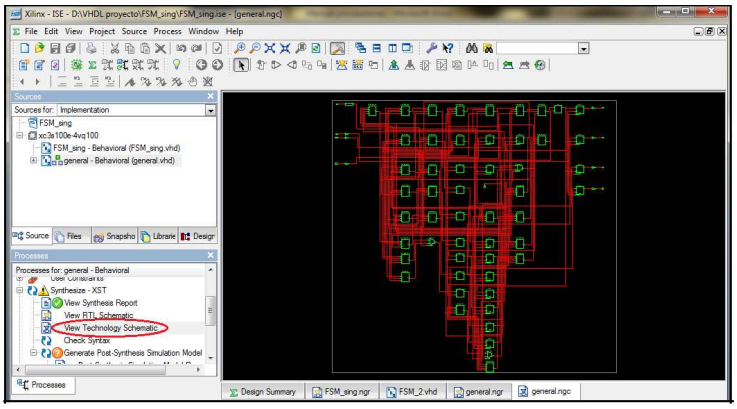


Figura 1.5: *Technology Schematic* de la entidad.

Con esto hemos terminado de analizar una herramienta de *ISE-WebPack* que es muy útil en el diseño avanzado de sistemas digitales. Hemos repasado conceptos del curso anterior. Comprobado que la herramienta funciona, y ya podemos probar otros diseños más complejos.

1.4 SIMULACION POST-ROUTE.

Un aspecto fundamental en el diseño de circuitos es tener la capacidad de simularlos para detectar errores en la descripción del comportamiento, rara vez diseñamos un circuito bien a la primera.
Para solucionar los errores en la descripción, la simulación del circuito resulta de mucha ayuda. Dentro de la practica 1 (Circuitos combinacionales y *Glitchs*) se hace imprescindible utilizar la *"Poust-Route Simulation "*, para poder detectar los *glitchs* generados, debido a que por medio de la *"Behavioral Simulation"* no es posible.
En la simulación se inserta el circuito en un banco de pruebas. El banco de pruebas es un modelo en VHDL que proporciona los estímulos al circuito para así poder ver si funciona como queremos. A diferencia de la simulación normal, la *"Poust-Route Simulation"*, introduce al diseño retardos en los elementos aproximándose a un modelo real y de esta manera la posibilidad de visualizar los *glitchs*. A continuación se propone el procedimiento para conseguir dicha simulación.
Lo primero es seleccionar la entidad a la cual se le va a realizar la simulación, en este caso es el diseño general de glitches1.Teniendo los pasos *"Synthesize-XST"* y *"Implement Design"* realizados correctamente, procedemos a generar la *"Poust-Route Simulation"* haciendo clic sobre el singo de mas (⊞) que se encuentra a la izquierda de *"Implement Design"*, desplegando el menú como se muestra en la figura 1.6.

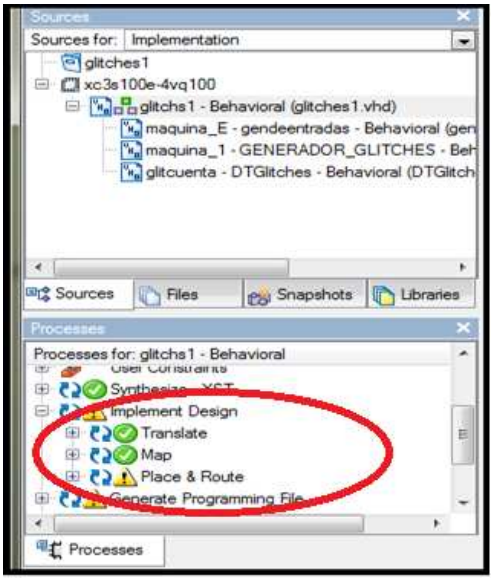


Figura 1.6: Desplegando el menú de "*Implement Design*"

Puede presentarse warnings, lo cual es normal, en este caso se ignoran ya que no afectan el diseño. Dentro del menú se encuentran tres pasos, que se han completado en la *"Implement Design"*: *Translate*, *Map* y *Place & Route*.

Luego damos clic en el signo de mas (⊞) que se encuentra a la izquierda de *Place & Route* y desplegamos el menu que es más extenso que el anterior, debido a que aparecen varios reportes y resultados que por el momento no es de nuestro interés. El menú desplegado de *Place & Route* se muestra en la figura 1.7.

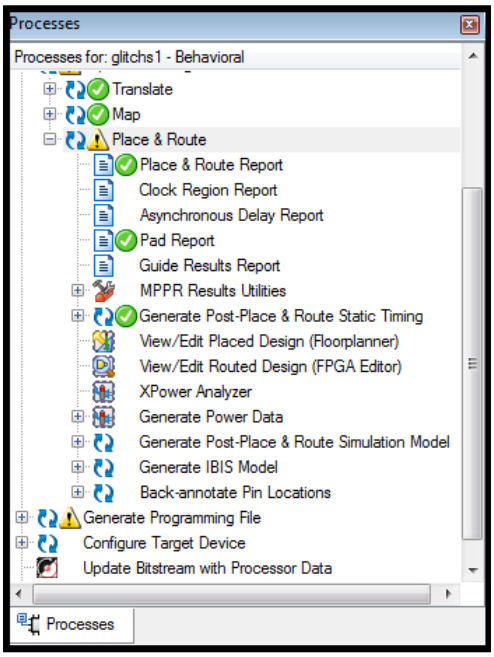


Figura 1.7: Desplegando el menú de "*Place & Route*".

Teniendo el menú desplegado se procede a generar el *Post-Route*, haciendo doble clic en *Generate Post-Place & Route Simulation Model*, esperando que se coloque un signo verde de correcto (Véase la figura 1.8).

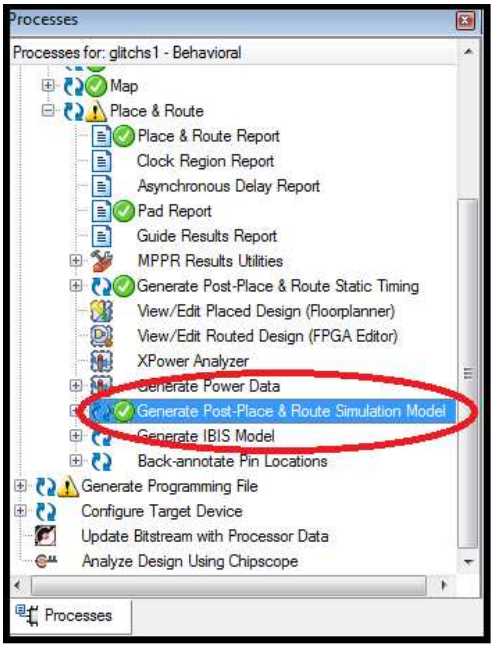


Figura 1.8: Menu de "*Generate Post-Place & Route Simulation Model*".

Luego de haber generado la simulación, nos dirigimos a la ventana de Sources que se encuentra en la parte superior y en la subventana Implementation damos clic en el símbolo de flecha que se encuentra a la derecha ( ) desplegando el menú de simulación.

Después se selecciona *Post-Route Simulation* y quedara anclado el archivo *Post-Route* a la tarjeta SIE. Ahora selecciona el banco de pruebas glitchs1 y en la ventana de Processes, creamos una nueva simulación, siguiendo el procedimiento para simular normalmente.

A continuación haz doble clic en *Simulate Post-Place & Route Model* y se observa la simulación, comprobando que los glitchs se generaron (véase figura 1.9).

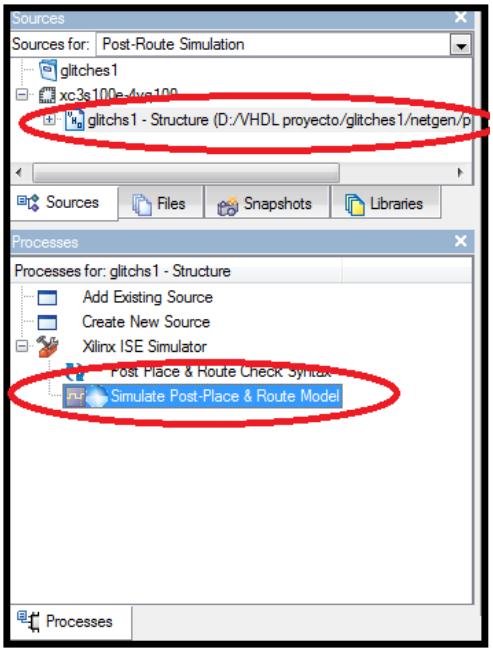


Figura 1.9: Arranque de la simulación.

En este parte hemos aprendido como comprobar que nuestro circuito generador de *glitchs* funcione de manera correcta. La comprobación de los diseños es una tarea que en muchos casos supera en tiempo de desarrollo a la propia tarea del diseño, por tanto, no se debe desestimar su importancia.

1.5 ACCEDIENDO AL PROCESADOR DE LA PLATAFORMA SIE.

En esta parte del manual realizaremos un repaso de como acceder al procesador de la plataforma SIE, para poder sintetizar e implementar códigos VHDL. Debido a que el estudio a fondo de este tema es visto en la materia de Sistemas Digitales que es dictada un periodo antes que la asignatura de Procesadores.
El primer objetivo es poder acceder al procesador de la tarjeta SIE, para tal fin primero se configura un puerto USB que sirva de interfaz , mediante la siguiente línea de código en la terminal de Linux .

```
1 $dmesg
2 $sudo iconfig usb0192.168.1.2 up
```

Script 1.1: Línea de código para configurar puerto USB.

Una vez configurado el puerto, comprobamos que efectivamente hay comunicación con la tarjeta aplicando las siguiente línea.

```
1 $ping 192.168.1.1
```

Script 1.2: Línea de código para comprobar comunicación con la tarjeta SIE.

Finalmente accedemos al procesador de la tarjeta SIE, esto se realiza con la siguiente instrucción.

```
1 $ssh root@ 192.168.1.1
```

Script 1.3: Línea de código para acceder al procesador de la tarjeta SIE.

El password de la tarjeta es “1234”.

ANEXO D: PROCEDIMIENTOS DE LAS PRÁCTICAS DE LABORATORIO PROCESADORES

LABORATORIO 1

IMPLEMENTACIÓN DE CIRCUITOS COMBINACIONALES Y GLITCHS

1.1 PROCEDIMIENTO GENERAL

Se presentará el procedimiento, el cual consiste en la identificación de riesgos que aparecen en la implementación de circuitos combinacionales, los cuales se colocarán en práctica a través de un ejemplo en la FPGA de la tarjeta SIE siguiendo los pasos que se muestran a continuación.

- Seleccionar el circuito combinacional que genere *glitchs*.
- Describir el circuito que detecte los *glitchs* generados
- Simular el circuito generador y detector de *glitchs* para su correcto funcionamiento.
- Implementación en la FPGA de la tarjeta SIE los circuitos generador y detector de glitchs.
- Simulación del circuito generador de *glitchs*.

1.2 PROCEDIMIENTO PASO A PASO

A continuación se muestra cada uno de los pasos para la realización de la correcta implementación de circuitos combinacionales y *glitchs*.

1.2.1 Paso 1: Seleccionar el circuito combinacional generador de *glitchs*

Existe una gran cantidad de circuitos combinacionales que generan *glitchs*. En este caso se opta por un circuito que se halla a partir de un mapa de karnaugh, en el cual se escogen posiciones aleatorias para generar la función qué se emplea como generador de *glitchs*. Este consta de 5 entradas(A, B, C, D, E)

y una salida (S). En la figura 1.1 se muestra el circuito generador de *glitchs* y la ecuación 1.1 la función generada obtenida mediante el análisis de las posibles combinaciones que permiten la aparición de los *glitchs*.

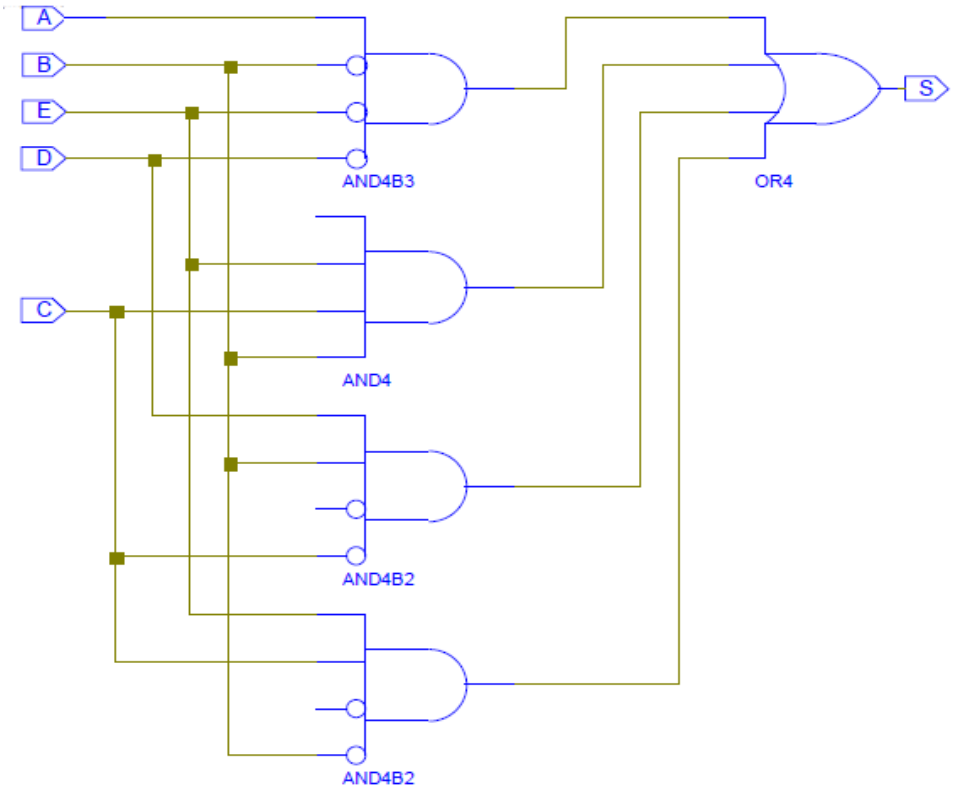


Figura 1.1: Circuito generador de *glitchs*

$$S = (\bar{B}CE\bar{A})OR(B\bar{C}D\bar{A})OR(BCEA)OR(\bar{D}\bar{E}\bar{B}A) \tag{1.1}$$

1.2.2 Paso 2 : Descripción del circuito detector de *glitchs*

A continuación se describe el circuito detector de *glitchs* mediante un *Filp-Flop* y la salida S como reloj del mismo. El circuito generador mantiene la salida en 1 al inicio, luego varia las entradas generando cambios predefinidos en la salida cada cierto tiempo, originando *glitchs static-one*, para así mediante un contador tener la cantidad de *glitchs* generados.

Como se observa en la figura 1.2 se implementa un circuito digital (Alu), el cual realiza la operación aritmética correspondiente, un Flip Flop y varias compuertas AND para su correcto funcionamiento.

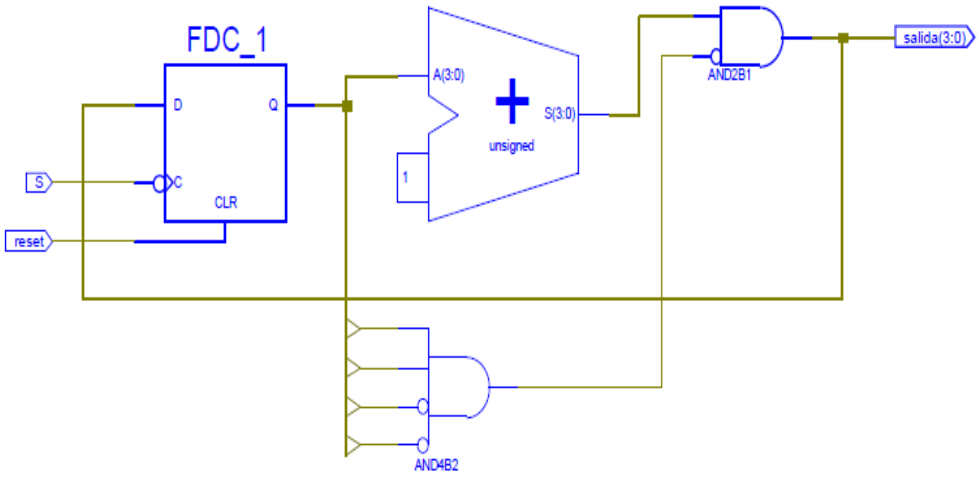


Figura 1.2: Circuito contador de *glitchs*

1.2.3 Paso 3: Implementación del circuito generador de entradas

Luego se hace necesaria la implementación de un circuito que genere los cambios en las entradas, para agregarlos a los circuitos generador y contador de *glitchs*. De esta forma se producen los *glitchs* en el diseño. Las entradas cambian con una frecuencia conocida teniendo una idea de cuantos *glitchs* se generan por unidad de tiempo.

A continuación en la figura 1.3 se muestra el circuito generador de entradas , el cual es un poco extenso debido a la utilización de varios elementos combinacionales y biestables para su funcionamiento, cabe anotar que el diseño de este circuito puede variar según el criterio de cada estudiante y la forma de atacar el problema de *glitchs*.

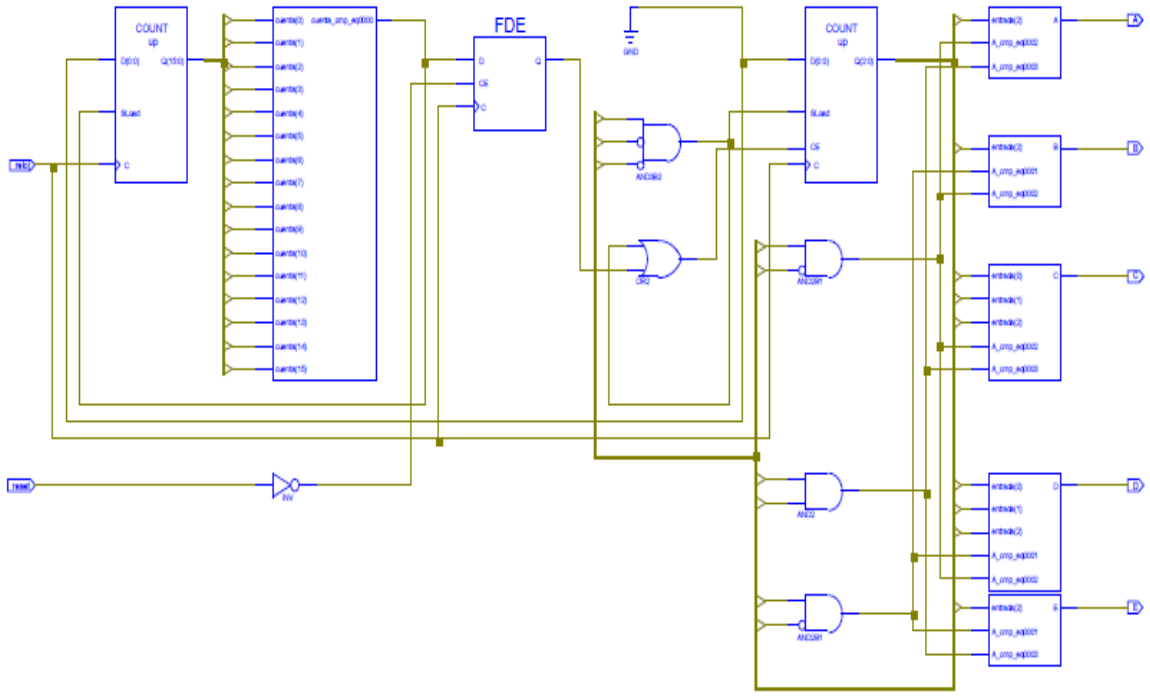


Figura 1.3: Circuito generador de entradas

1.2.4 Paso 4 : Integración de todos los circuitos del diseño

Después de realizar el paso anterior, se procede a integrar todos los circuitos para desarrollar la debida verificación del diseño, implementándolo mediante la simulación para comprobar el funcionamiento y de esta manera constatar que los circuitos que se escogieron cumplen con los objetivos planteados. El circuito para la simulación se puede ver en la figura 1.4.

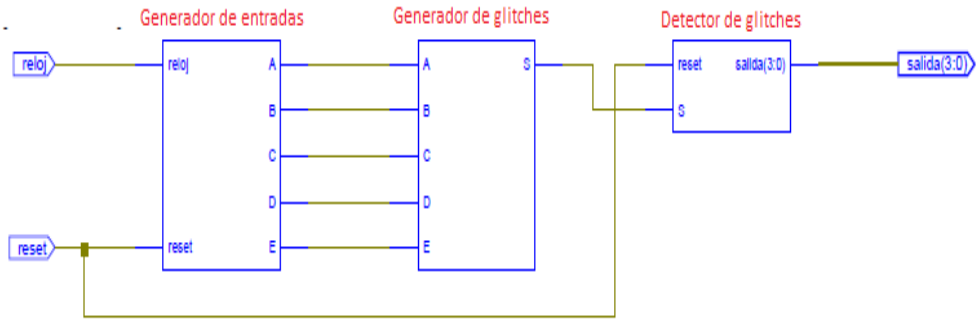


Figura 1.4: Circuito para la simulación

1.2.5 Paso 5: Simulación del circuito generador de *glitchs*

En la figura 1.5 se muestra la simulación del circuito generador de *glitchs*. la señal de salida S siempre debe permanecer en 1, pero como se observa en los circulos rojos se generán los *glitchs*,de esta manera se comprueba que el circuito implementado funciona de la forma esperada,garantizando el buen desarrollo de la práctica.

Figura 1.5: Simulación de *glitchs*

Luego de la generación de los *glitchs* se cuenta por unidad de tiempo como se observa en la señal salida, la cual corresponde a un vector de [6,0], debido a que esta señal se visualizá en un Pmod siete segmento

LABORATORIO 2

IMPLEMENTACIÓN DE MÁQUINAS DE ESTADO EN VHDL

2.1 PROCEDIMIENTO GENERAL

Teniendo en cuenta el ejercicio a realizar se presenta el procedimiento de la práctica, el cual consiste en la implementación de la máquina de estados para la solución de un control de un parqueadero, utilizando la metodología del libro de Chu para la elaboración de la FSM. A continuación se presenta los pasos a seguir para la realización de la práctica.

- Realizar el diagrama de estados.
- Diseñar el contador de los automóviles en el parqueadero.
- Descripción e implementación del código VHDL.
- simular el código VHDL de la máquina de estados para comprobar su correcto funcionamiento.
- construir el hardware requerido para la implementación.

2.2 PROCEDIMIENTO PASO A PASO

A continuación se muestra cada uno de los pasos para la realización del ejercicio propuesto.

2.2.1 Paso 1: Realizar el diagrama de estados

Un diagrama de estados se utiliza para describir el comportamiento de un sistema, es decir nos muestra el comportamiento de un objeto ante un estimulo. Teniendo en cuenta la descripción del ejercicio se realizo el diagrama de estados de la FSM qué ejecutara el control del número de autos dentro de un parqueadero.

El diagrama de estados se observa en la figura 2.1, el cual consta de 6 estados, detallando cómo deben ser las respuestas del sistema en cada uno de ellos. Cabe señalar que se opto por implementar una maquina de estado tipo Mealy qué depende tanto del estado actual como de las entradas D1 y D2, además esta máquina genera 2 salidas una que indica cuando un auto entra y la otra cuando sale.

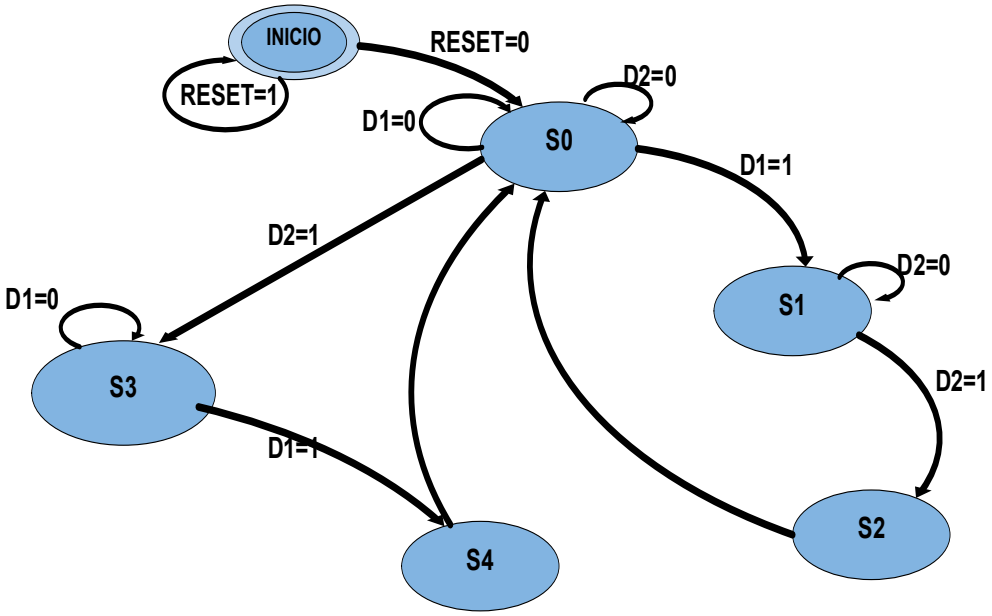


Figura 2.1: Diagrama de estados del parqueadero de autos.

2.2.2 Paso 2 : Diseñar el contador de los automóviles en el parqueadero

En este paso se realiza la implementación del contador, el cual indica el número de automóviles registrados dentro del parqueadero.

El problema de conteo se resuelve con la ayuda de un contador bidireccional, el cual se incrementa cuando se activa la señal E_auto y decrementa cuando la señal S_auto es activada. Teniendo en cuenta que la salida del contador se visualizá en un Pmod siete segmentos, se incluye dentro del contador un modulo que transforme la señal a siete segmentos. A continuación en la figura 2.2 se puede ver el diagrama RTL del contador, el cual ademas de las señales de entrada cuenta también con un reset y un reloj.

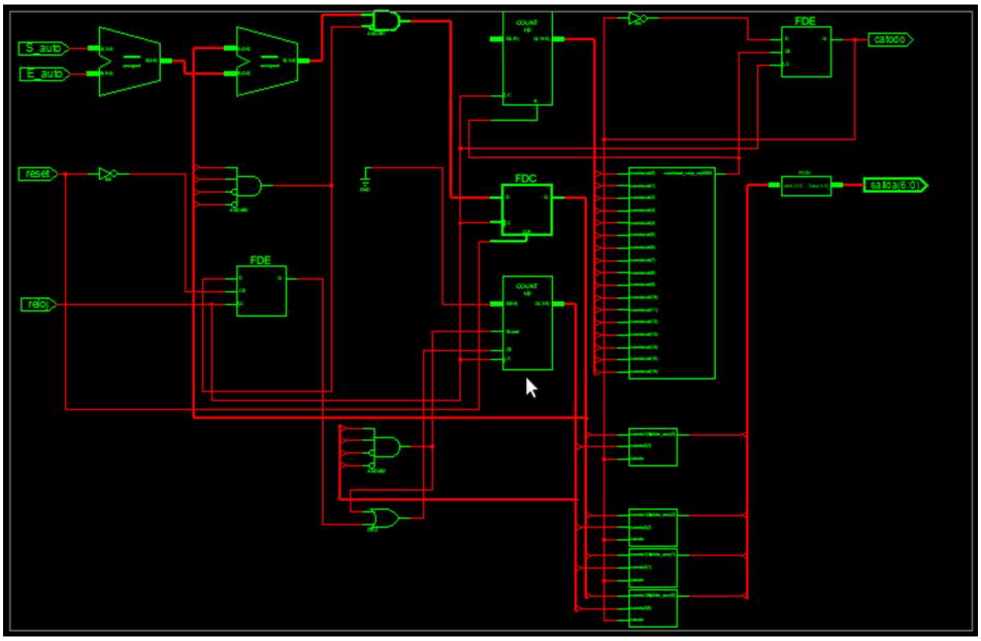


Figura 2.2: RTL del contador

2.2.3 Paso 3: Descripción e implementación del código VHDL

Luego de haber realizado el diseño del ciruito contador de automóviles, se desarrolla la descripción en VHDL de la máquina de estados, teniendo en cuenta los tópicos mencionados dentro del marco teórico de la práctica.

- **Máquina de estados**

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity FSMparq is
7  port(clk, reset      : in std_logic;
8       D1, D2          : in std_logic;
9       S_auto, E_auto :out std_logic);
10 attribute FSM_EXTRACT:string;
11 attribute FSM_EXTRACT of FSMparq:entity is "NO";
12
13 end FSMparq;
14
15 architecture Behavioral of FSMparq is
16
17 constant n_state_bits: integer :=3+2;
18
```

```
19 constant inicio: std_logic_vector(n_state_bits-1 downto 0):="000"&"00";
20 constant s0: std_logic_vector(n_state_bits-1 downto 0):="001"&"00";
21 constant s1: std_logic_vector(n_state_bits-1 downto 0):="010"&"10";
22 constant s2: std_logic_vector(n_state_bits-1 downto 0):="011"&"00";
23 constant s3: std_logic_vector(n_state_bits-1 downto 0):="100"&"00";
24 constant s4: std_logic_vector(n_state_bits-1 downto 0):="101"&"01";
25
26 signal  estado_presente, estado_siguiente: std_logic_vector(n_state_bits-1 downto 0);
27 begin
28 -------asignacion de salidas--------
29
30 S_auto <=estado_presente(0);
31 E_auto <=estado_presente(1);
32
33 ----------registro de estado---------
34 process (clk,reset)
35 begin
36      if (reset = '1') then
37         estado_presente <= inicio ;
38      elsif (clk'event and clk='1')then
39         estado_presente <= estado_siguiente;
40      end if;
41
42 end process;
43
44 -------logica del estado siguiente--------
45 process(estado_presente, D1, D2)
46 begin
47     case estado_presente is
48         when inicio =>
49                  estado_siguiente <= s0;
50         when s0 =>
51             if  (D1 = '1') then
52                  estado_siguiente <= s1;
53                           elsif (D2 = '1')then
54                           estado_siguiente <= s3;
55                  else
56                           estado_siguiente <= s0;
57                  end if;
58          when s1 =>
59             if (D2 = '1') then
60                  estado_siguiente <= s2;
61             else
62                  estado_siguiente <= s1;
63              end if;
64          when s2 =>
65                  estado_siguiente <= s0;
66          when s3 =>
67             if (D1 = '1')then
68                  estado_siguiente <= s4;
69             else
70                  estado_siguiente <= s3;
71              end if;
```

```
72          when s4 =>
73                  estado_siguiente <= s0;
74          when others =>
75                  estado_siguiente <= s0;
76      end case;
77  end process;
78
79  end Behavioral;
```

Script 2.1: Código VHDL de la Maquina de Estados.

2.2.4 Paso 4: Simular el código VHDL de la máquina de estados para comprobar su correcto funcionamiento.

Después de realizar el paso anterior se procede a integrar todos los módulos. Verificando que el funcionamiento del diseño, y de esta manera constatar que los circuitos que se escogieron cumplierón con los objetivos planteados.

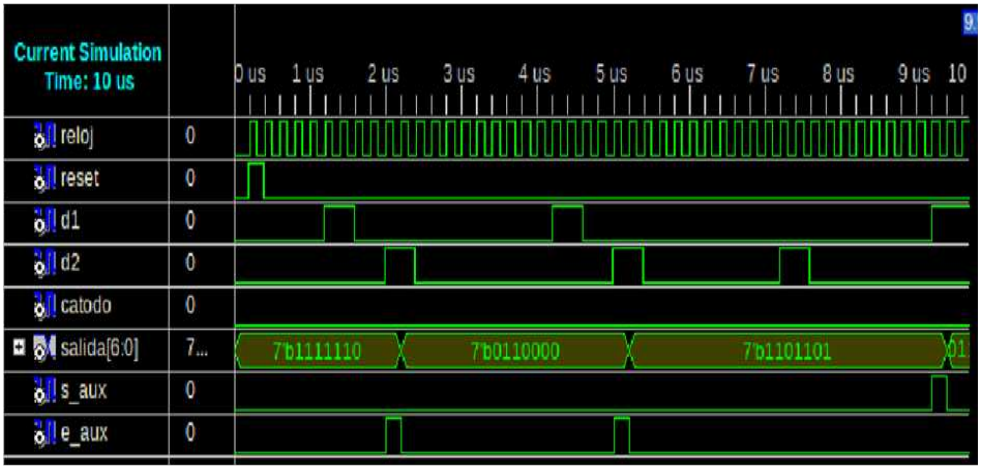


Figura 2.3: Simulación de la implementación FSM.

El la simulación se pude observar las señales S_aux y E_aux que indican cuando un automóvil sale o entra al parqueadero. también la señal salida que nos permite la visualización en el pmod site-segmentos.

2.2.5 Paso 5: construir el hardware requerido para la implementación.

Para cumplir con las necesidades de la práctica en cuanto a la implementación física del contador de automóviles de un parqueadero se hicieron necesarios los siguientes elementos: Tarjeta SIE, 2 encoders, 4 resistencias (2 de 220Ω y 2 de 2.2kΩ), Pmod siete-segmentos.

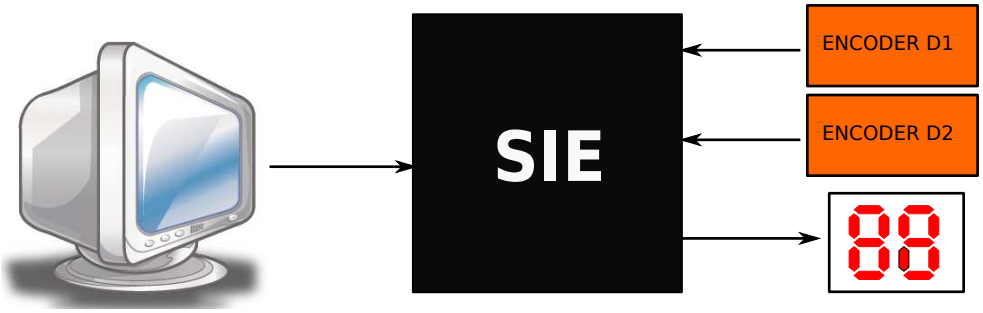


Figura 2.4: Diagrama de implementación.

la salida del encoder es conectada directamente a la tarjeta SIE donde se realizá el procesamiento de la señal. El circuito del encoder es alimentado a una tensión de 5 voltios y dos resistencias, las cuales nos darán una mejor sintonización de la señal de salida del encoder. la figura 2,5 muestra el circuito de alimentación del encoder.

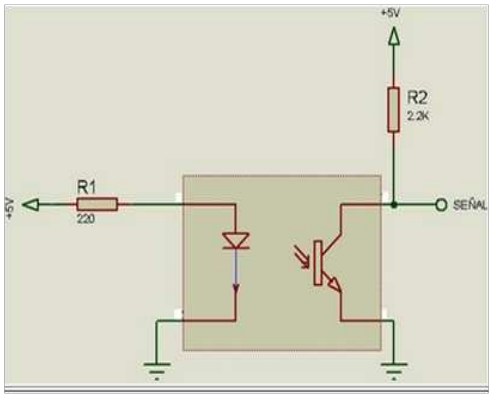


Figura 2.5: Circuito de alimentación del Encoder.

El funcionamiento de los encoder en muy sencillo, al detectar un objeto en medio de su ranura, envia un señal de alto a la tarjeta SIE, reconociendo la entrada o salida de un automóvil dependiendo que sensor es activado.

LABORATORIO **3**

VIOLACIONES DE SETUP Y HOLD "METAESTABILIDAD"

3.1 PROCEDIMIENTO GENERAL.

Se presentará el procedimiento, el cual consiste en visualizar e identificar las causas de un circuito que presenta el fenómeno de metaestabilidad, colocándolo en práctica a través de un ejemplo en la FPGA de la tarjeta SIE siguiendo los pasos que se muestran a continuación.

- Definir las causas por las cuales se va inducir el circuito en metaestabilidad.1
- Investigar los valores de tiempos de *setup*, *hold* y propagación contenidos en la hoja de datos de la FPGA (SPARTAN XCS3100E-4).
- Seleccionar y construir el circuito que será inducido en metaestabilidad
- Simular el circuito para comprobar que se encuentra en metaestabilidad.
- Realizar la implementación del circuito en la FPGA de la tarjeta SIE.

3.2 PROCEDIMIENTO PASO A PASO.

A continuación se muestra detalladamente cada uno de los pasos, para la realización de la correcta simulación del circuito en metaestabilidad.

3.2.1 Paso 1: Definir las causas por las cuales se va inducir el circuito en metaestabilidad.

Teniendo en cuenta la teoría vista sobre metaestabilidad y comprendiendo que no es posible eliminar la metaestabilidad pero si reducir su aparición hemos tomado la decisión de construir un circuito que se encuentre en estado metaestable por medio de violaciones de los tiempos de setup, lo cual se lograra usando el DCM de la FPGA aprovechando qué este nos permite variar el valor del skew. Usaremos esta característica del DCM para lograr qué el circuito entre en estado de metaestabilidad.

3.2.2 Paso 2: Investigar los valores de tiempos de *setup*, *hold* y propagación contenidos en la hoja de datos de la FPGA (SPARTAN XCS3100E-4).

A manera de investigación se hace necesario obtener los valores de tiempo de setup, hold y propagación de los *Flip Flop* de la FPGA, contenidos en la hoja de datos de la SPARTAN XSC3100E-4. Dichos valores se encuentra en el documento que se puede descargar de la siguiente dirección web: `http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf` y se observan en la tabla 3.1.

Elemento	Setup	Hold	Propagación
Flip-Flop	7.1[ns]	3.93[ns]	5.97[ns]

Tabla 3.1: Valores de setup, hold y propagación del Flip-Flop

Cabe anotar que estos valores corresponde a los biestables contenidos en la FPGA, si se utilizan otros elementos como compuertas lógicas en el circuito, se deben averiguar sus propios tiempos.

3.2.3 Paso 3: Seleccionar y construir el circuito que será inducido en metaestabilidad.

Utilizando una maquina de estados se construirá un circuito el cual se encargara de generar una serie de datos conocida que será enviada a un circuito receptor el cual será implementado por medio de un *flip-flop*. Los relojes del *flip-flop* y el circuito generador serán proporcionados con el DCM de la tarjeta SIE, el cual nos brinda versatilidad a la hora de variar características como la frecuencia del o los relojes, así como introducir un valor de skew que puede ser positivo o negativo de acuerdo a las necesidades del usuario. En la figura 3.1 se observa el diagrama RTL circuito.

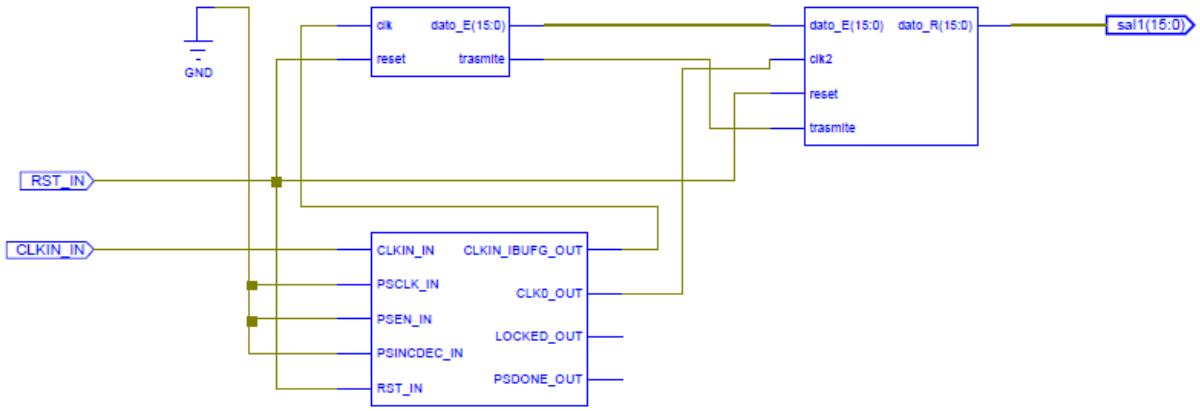


Figura 3.1: Diagrama RTL circuito.

Para lograr que uno de los flip-flop entre en estado metaestable se debe tener en cuenta el valor del tiempo de propagación del circuito generador, de esta manera generar el skew correcto que induzca el circuito en estado de metaestabilidad. En la figura 3.2 se observa un diagrama con el cual se pretende explicar como la contribución del skew del DCM logra llevar el circuito al estado metaestable.

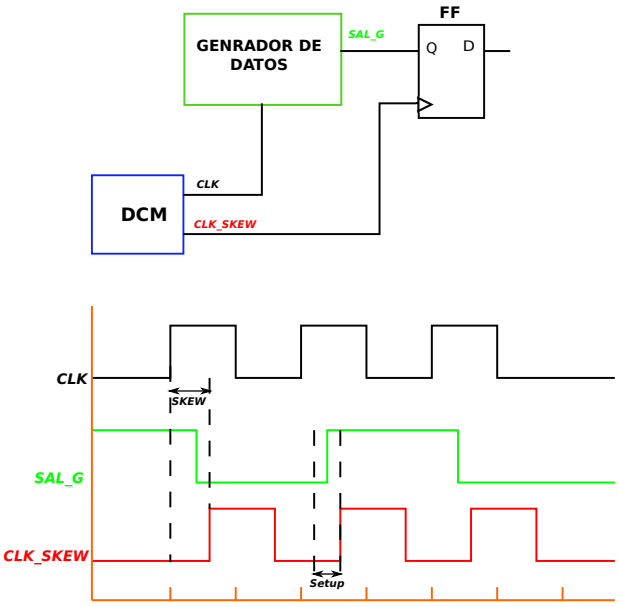


Figura 3.2: Diagrama de contribución del skew del DCM paraa llevar el circuito a metaestabilidad

3.2.4 Paso 4 : Simular el circuito para comprobar que se encuentra en estado metaesble

Realizada la construcción del circuito, se procede a comprobar que efectivamente el sistema se encuentra en metaestabilidad. Despues de realizar las simulaciones para diferentes valores de skew se observa en la figura 3.3 que no es posible visualizar el momento en el cual el circuito entra en estado metaestable, por lo cual la otra opción es obsevar la la metaestabilidad por medio de la implementación física.

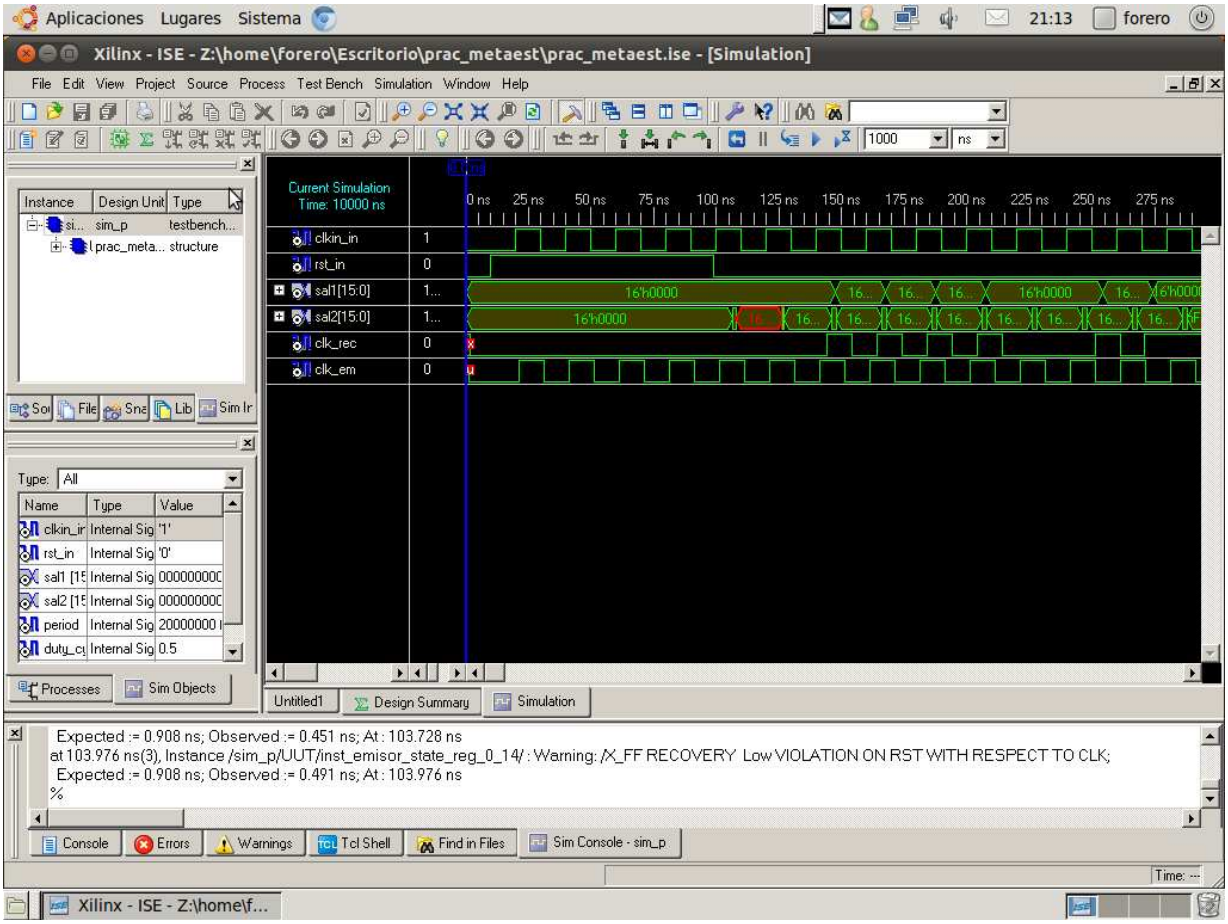


Figura 3.3: Simulación del circuito implementado

3.2.5 Paso 5: Realizar la implementación del circuito en la FPGA.

Para la implementación física en la FPGA se determino la siguiente estrategia para observar si el circuito realmente entra en metaestabilidad.

Primero se implementa un circuito que nos permita establecer de alguna manera si nuestro diseño entra

en metaestabilidad. El circuito utiliza un comparador y un contador el cual se incrementará cuando las estradas que corresponde a la salida del generador y la salida del *filp-flop* receptor sean diferentes, esto nos permitirá identificar que el circuito entra en metaestabilidad, además para garantizar qué nuestros datos se hán comparados correctamente se coloca un *flip-flop* en la entrada del comparador que corresponde a la salida del generador, pero con el mismo reloj del generador es decir sin skew.Debido a la conexion anterior, en el Pmod siete segmente se visualiza un incremento en el valor cada vez que ocurre el fenomeno de la metaestabilidad, asegurarndo que efectivamente el circuito sufre de dicho fenomeno. A continuacion en la figura 3.4 se observa el digrama del circuito implementado.

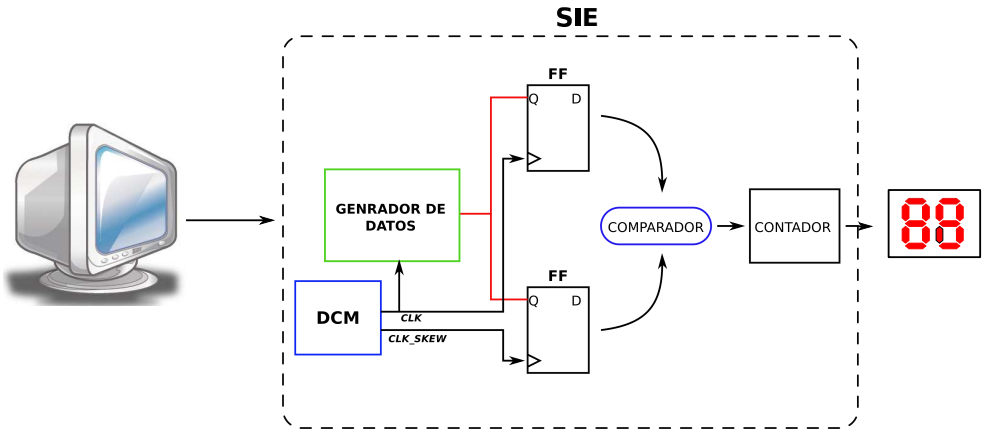


Figura 3.4: Digrama del circuito implementado

LABORATORIO 4

DOMINIOS DE RELOJ Y SINCRONIZADORES

4.1 PROCEDIMIENTO GENERAL.

Para solucionar el ejercicio planteado, se desarrollán una serie de pasos.

- Diseñar el circuito emisor.
- Diseñar el circuito receptor.
- Seleccionar el sincronizador adecuado.
- Implementar el sincronizador en VHDL.
- Simulación de los circuitos del diseño.
- Implementación física del diseño.

4.2 PROCEDIMIENTO PASO A PASO.

A continuación se muestra detalladamente cada uno de los pasos, para la realización del ejercicio propuesto.

4.2.1 Paso 1: Diseñar el circuito emisor.

Para el desarrollo de la práctica es necesario crear un circuito emisor que debe enviar una secuencia de, datos para esto se utiliza una máquina de estados con frecuencia f1 que se describe con el diagrama de estados de la figura 4.1. En el diagrama se puede observar que se tienen tres estados el estado inicio y los estados S0 y S1, en los cuales se generan las salidas.En S0 la salida es 00 y en S1 es FF.

4.2.2 Paso 2: Diseñar el receptor.

En las especificaciones del ejercicio se plantea el diseño de un circuito receptor que trabaje con una frecuencia de reloj f2 diferente a la frecuencia f1 del emisor, dicho circuito es implementado con Flip-Flops

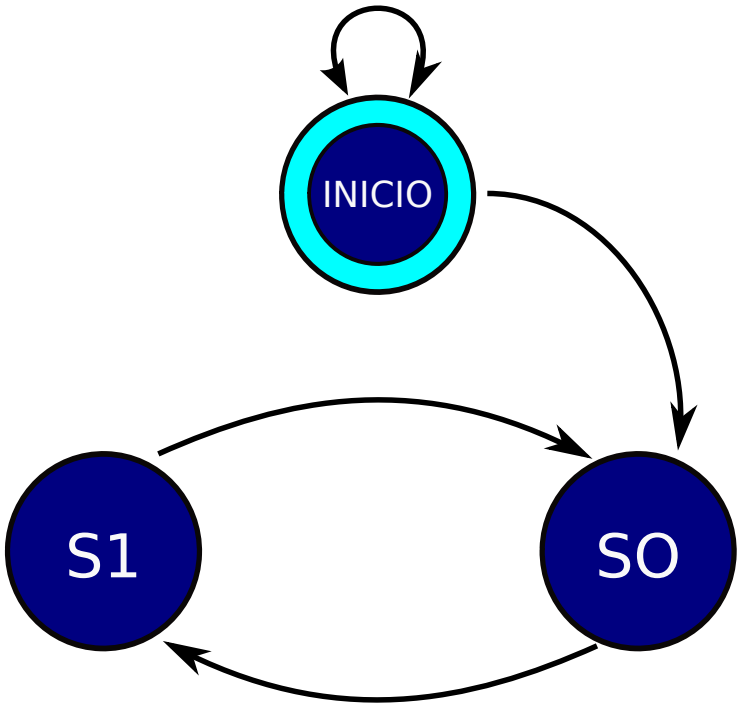


Figura 4.1: Diagrama de estados del emisor

tipo D como un registro de 8 bits. En la figura 4.2 se muestra el diagrama *RTL* del circuito y en la figura 4.3 el Technology schematic.

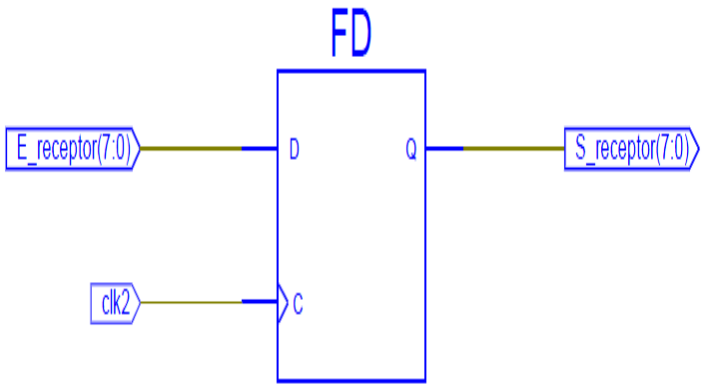


Figura 4.2: *RTL schematic* del receptor.

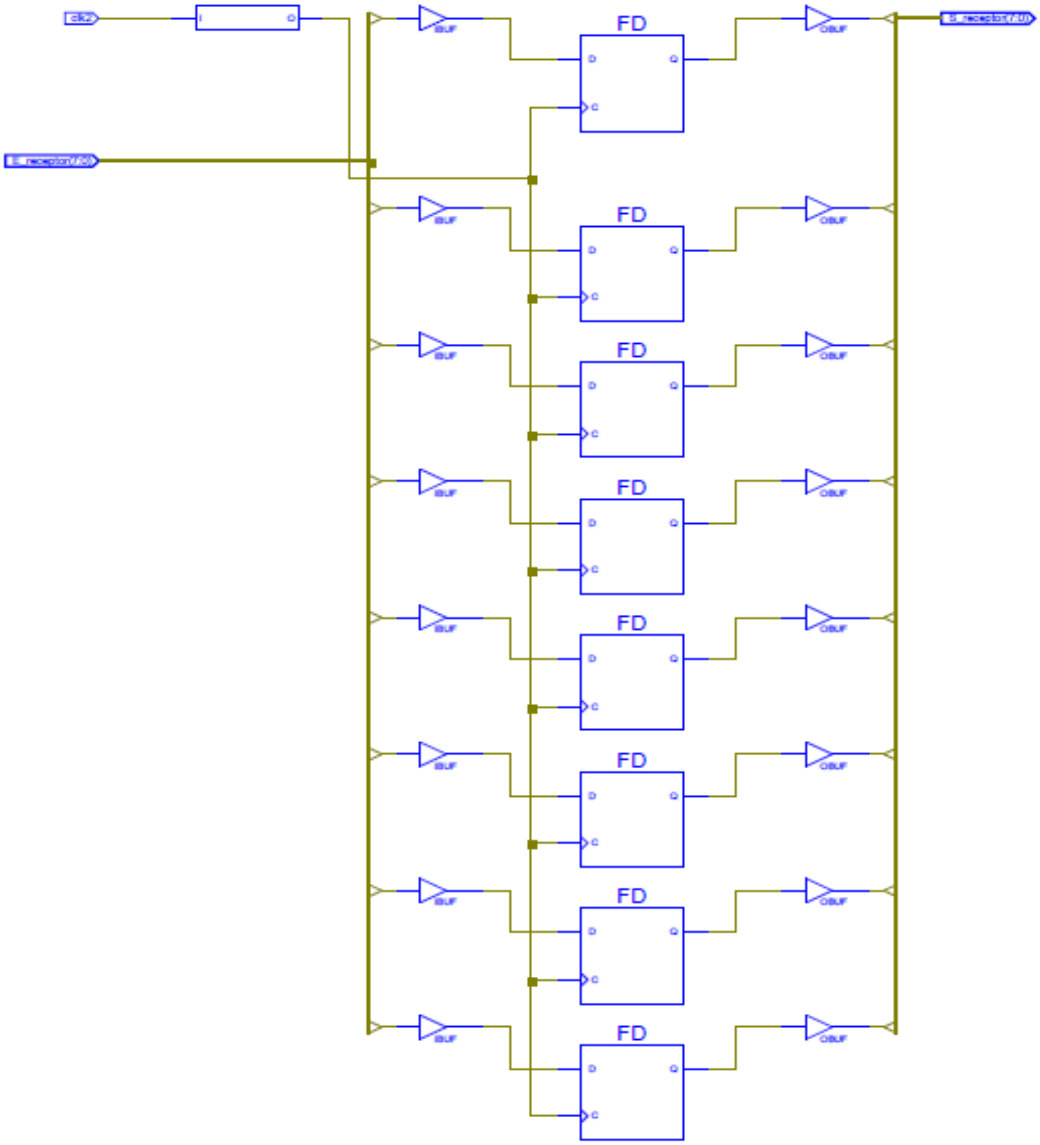


Figura 4.3: *Technology schematic* del receptor.

4.2.3 Paso 3: Seleccionar el sincronizador adecuado.

Para seleccionar el sincronizador tenemos primero que escoger las frecuencias f1 y f2 y asi poder definir el tipo de sincronizador que se debe implementar.
teniendo en cuenta los requerimientos de la práctica, se debe mostrar los datos que obtiene el receptor en un pmod siete segmentos, esto quiere decir qué el problema esta en escoger una frecuencia donde el ojo humano pueda detectar los cambios para mostrar los datos, esta corresponde a la frecuencia f2 y sabiendo que el ojo humano no distingue frecuencias superiores a 30Hz, se escoje a f2 por debajo de este valor.
Ahora como sabemos que la frecuencia f2 tiene que ser una frecuencia baja entonces escogemos la frecuencia f1 como una frecuencia alta. Teniendo toda esta infromación se escoje el sincronizador a implementar y de acuerdo al marco teorico, nos referimos al caso 7 en el cual f1>>f2.

4.2.4 Paso 4: Implementar el sincronizador.

Despues de escoger el sincronizador adecuado realizamos la implemantacion en VHDL, construyendo el circuito del sincronizador del caso 7. En las figuras 4.4 y 4.5 se puede apreciar el diagrama *RTL schematic* y el *Technology schematic* del sincronizador implementado.

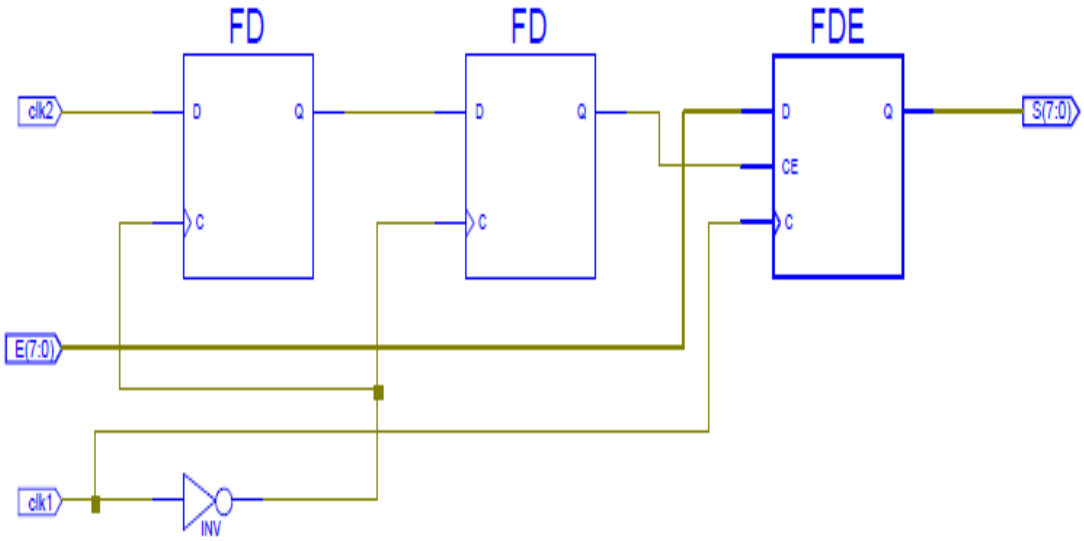


Figura 4.4: *RTL schematic* del sincronizador.

como se puede observar el diagram de la figura 4.4 es muy similar al circuito contenido dentro del marco teorico, esto nos da la confianza de que el circuito esta bien estructurado.

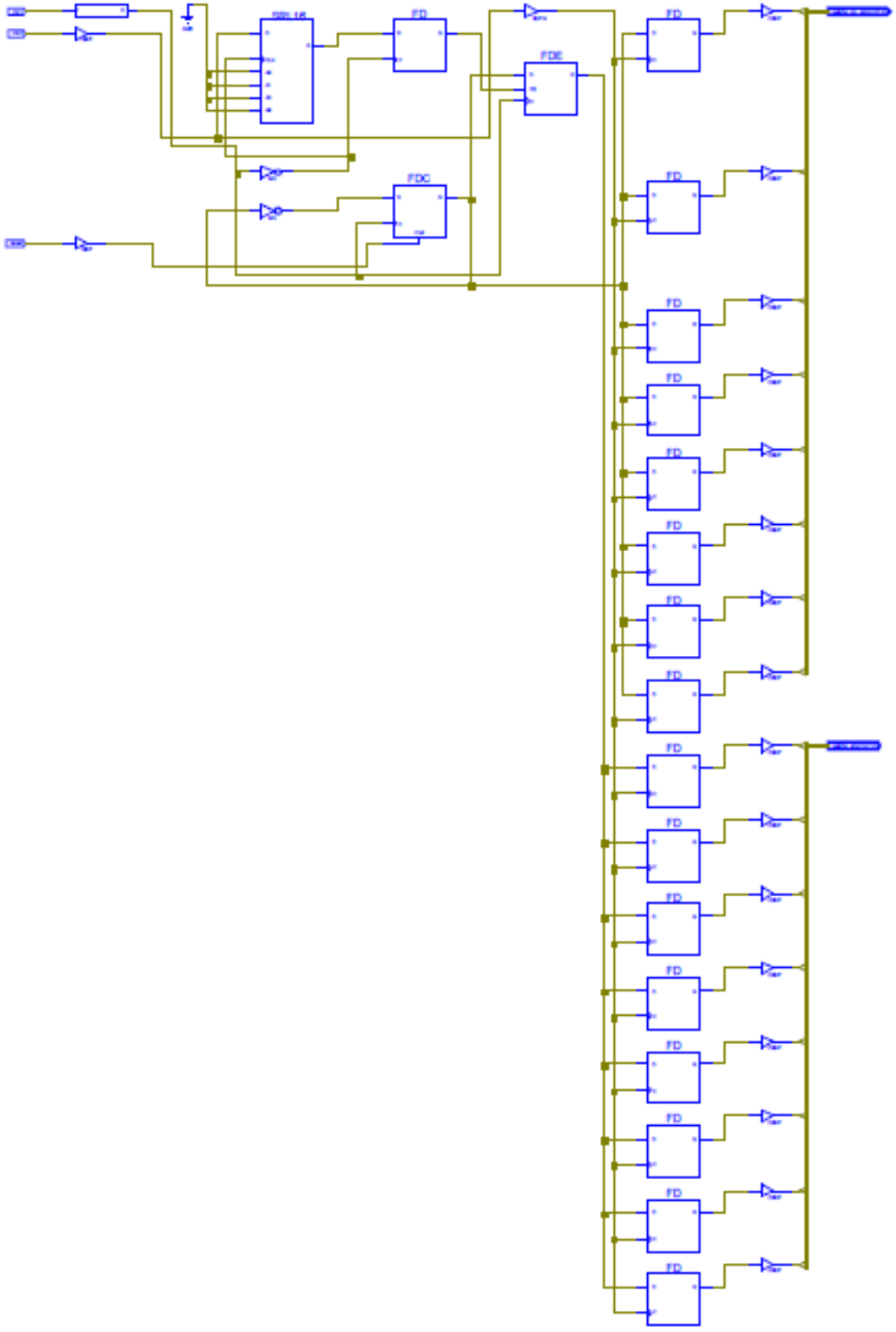


Figura 4.5: *Technology schematic* del sincronizador.

Cabe anotar que las frecuencias f1 y f2 son generadas por medio del DCM, analizado y estructurado en el procedimiento de la práctica 3.

4.2.5 Paso 5: Simulación de los circuitos del diseño.

Dentro de este paso solamente se comprueba el buen funcionamiento de los circuitos empleados, debido a que por medio de la simulación es muy complicado observar la metaestabilidad que genera los dominios de reloj diferentes. Simplemente se observa el circuito del diseño funcionando correctamente con el sincronizador.

Para poder observar la simulacion se creo un archivo VHDL, donde se interconectan todos los modulos del diseño. En la figura 4.6 se observa la simulación del circuito general.

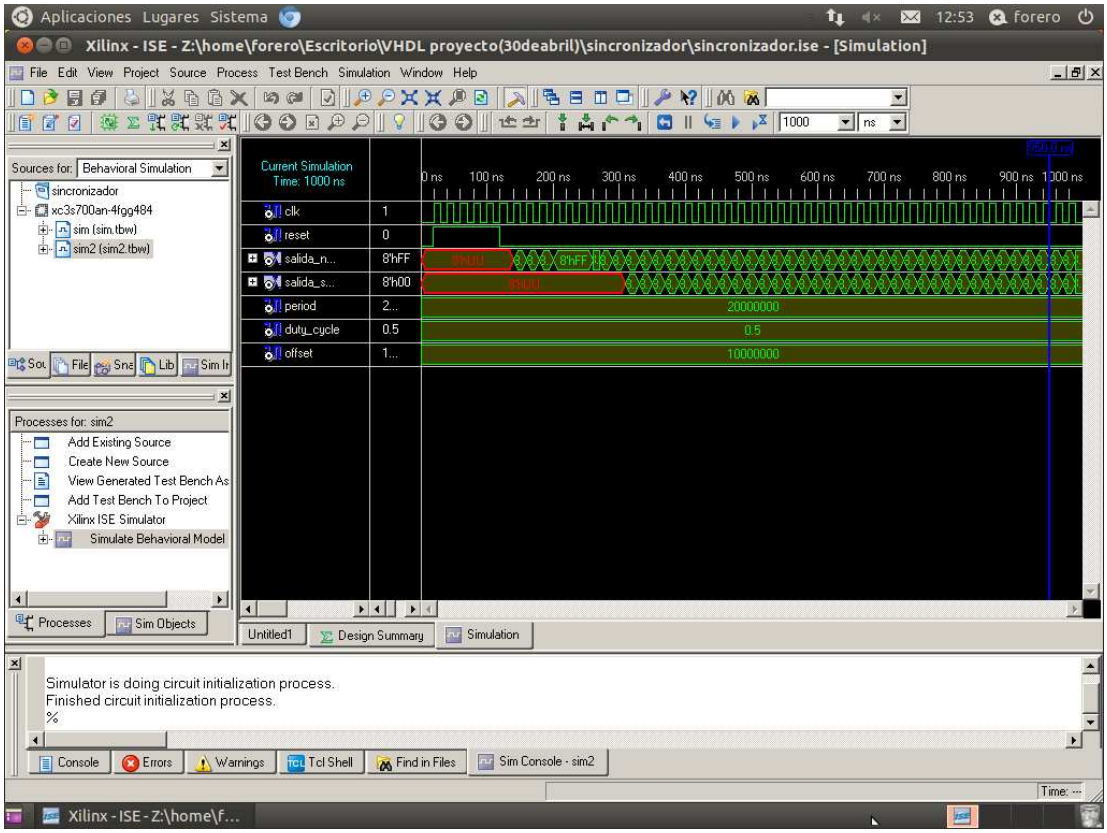


Figura 4.6: simulacion del circuito.

4.2.6 Paso 6: Implementación física del diseño.

Teniendo en cuenta todas las especificaciones de la práctica de laboratorio, se implementa el circuito general completamente en la FPGA de la tarjte SIE y debido a que el unico periferico que se utiliza es

el Pmod siete segmentos, se decidio ilustrar esta parte con un diagrama general de la implementacion, el cual se muestra en la figura 4.7, teniendo en cuenta que la tarjeta SIE se incluye como una caja negra.

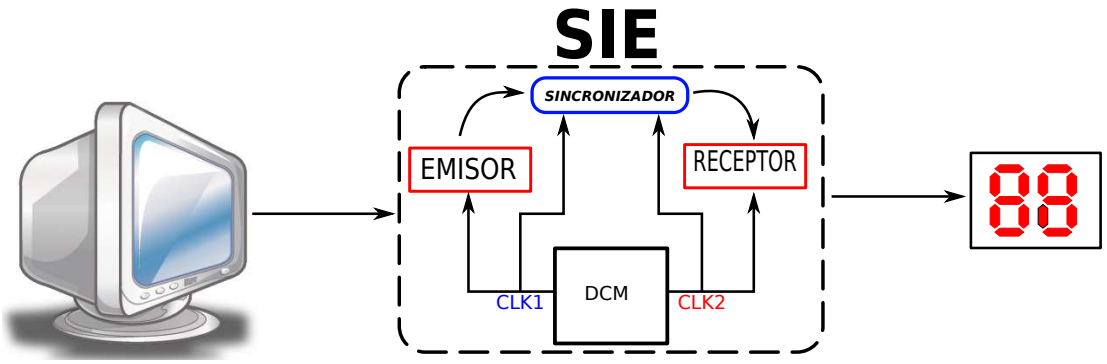


Figura 4.7: Diagrama general de la implementación.

LABORATORIO **5**

PARALELISMO Y PROCESADORES SEGMENTADOS "PIPELINE"

5.1 PROCEDIMIENTO GENERAL.

Se presentará el procedimiento, el cual consiste en definir el número de etapas de pipeline de un multiplicador, colocandolo en práctica a través de un ejemplo en la FPGA de la tarjeta SIE y siguiendo los pasos que se muestran a continuación.

- Seleccionar los circuitos serán necesarios para la implementación del ejercicio.
- implementación de los circuitos en vhdl.
- Diagramas *RTL y Technology Schematic*.
- Seleccionar el número de etapas de pipeline que optimizan el sistema.

5.2 PROCEDIMIENTO PASO A PASO.

A continuación se muestra detalladamente cada uno de los pasos, para la realización de la correcta del ejercicio propuesto en la práctica

5.2.1 Paso 1: Seleccionar los circuitos serán necesarios para la implementación del ejercicio.

Teniendo en cuenta los requerimientos de la práctica, como primera medida se establecen los valores de n y f de la siguiente manera:
f= 100MHz
n= 32bits

Ya que es necesario un reloj de 100 MHz se propone el uso del DCM para generar la frecuencia, además de los circuitos correspondientes al multiplicador y el circuito generador de números aleatorios. En la figura 5.1 se observa el diagrama de bloques completo del circuito.

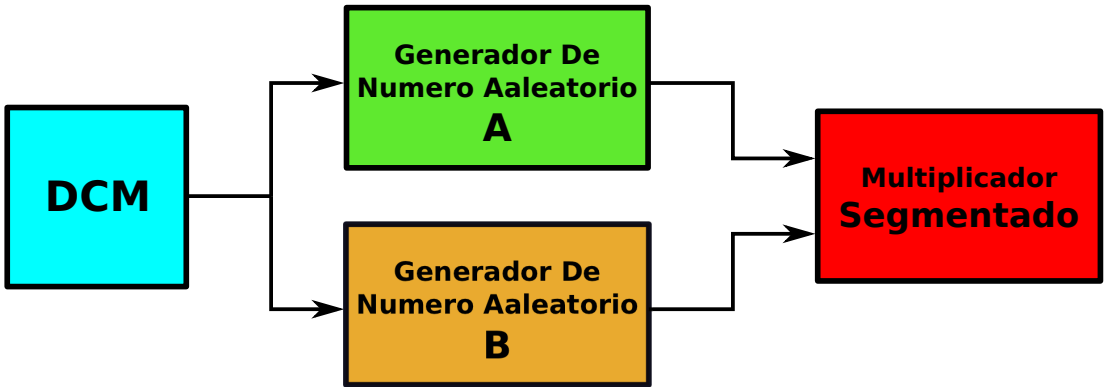


Figura 5.1: Diagrama de bloques del multiplicador

5.2.2 Paso 2:Implementación de los circuitos en VHDL.

El principal objetivo de la actividad, es la de generar un circuito multiplicador que permita optimizar el diseño y desarrollar las competencias necesarias que corresponde a esta experiencia de laboratorio. A continuación se muestra el circuito VHDL de un multiplicador segmentado y el generador de números aleatorios.

```
1  library IEEE;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity mult_outside is
6  port(clk : in std_logic;
7          A : in unsigned (31 downto 0);
8          B : in unsigned(31 downto 0);
9          MULT : out unsigned( 63 downto 0));
10 attribute mult_style: string;
11 attribute mult_style of mult_outside: entity is "pipe_lut";
12 end mult_outside;
13
14
15 architecture Behavioral of mult_outside is
16 signal a_in, b_in : unsigned (31 downto 0);
17 signal mult_res : unsigned( 63 downto 0);
18 signal pipe_1, pipe_2, pipe_3:  unsigned(63 downto 0);
19 begin
20 mult_res <= a_in * b_in;
21 process (clk)
22 begin
```

```
23  if (clk'event and clk='1') then
24  a_in <= A; b_in <= B;
25  pipe_1 <= mult_res;
26  pipe_2 <= pipe_1;
27  pipe_3 <= pipe_2;
28  MULT <= pipe_3;
29  end if;
30  end process;
31
32  end Behavioral;
```

Script 5.1: Código VHDL del circuito multiplicador.

```
1   entity ramdom32 is
2   port (EN  : in std_logic;
3         CLK: in std_logic;
4         RSTz: in std_logic;
5         O   : out unsigned (31 downto 0));
6   end ramdom32;
7
8   architecture Behavioral of ramdom32 is
9   signal ran: unsigned (31 downto 0);
10  begin
11  process (RSTz,CLK)
12  begin
13  if RSTz='0' then
14  ran <= "00000000000000000000000000000001"; --seed
15  elsif CLK'event and CLK='1' then
16  if EN='1' then
17  for i in 31 downto 1 loop
18  ran(i) <= ran(i-1);
19  ran(0) <= ran(31) xor ran(6) xor ran(4) xor ran(2) xor ran(1) xor ran(0);
20  end loop;
21  end if;
22  end if;
23  end process;
24  BOUT: process (ran,RSTz)
25  begin
26  O <= ran;
27  end process;
28
29  end Behavioral;
```

Script 5.2: Código VHDL del generador de numeros aleatorios.

Cabe anotar que el código VHDL del circuito multiplicador fue tomado de *XST User Guide* y adaptado a las necesidades del ejercicio.

5.2.3 Paso 3.Diagramas *RTL y Technology Schematic*

. A continuación se presenta los diagramas *RTL y Technology Schematic*, los cuales se observan en las figuras 5.2 y 5.3 en donde podemos apreciar como ISE realiza la implementación del circuito.

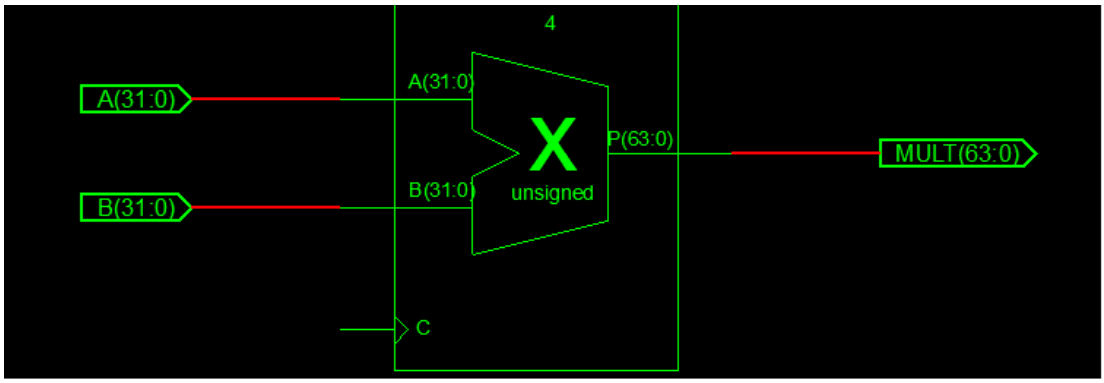


Figura 5.2: *RTL* del multiplicador

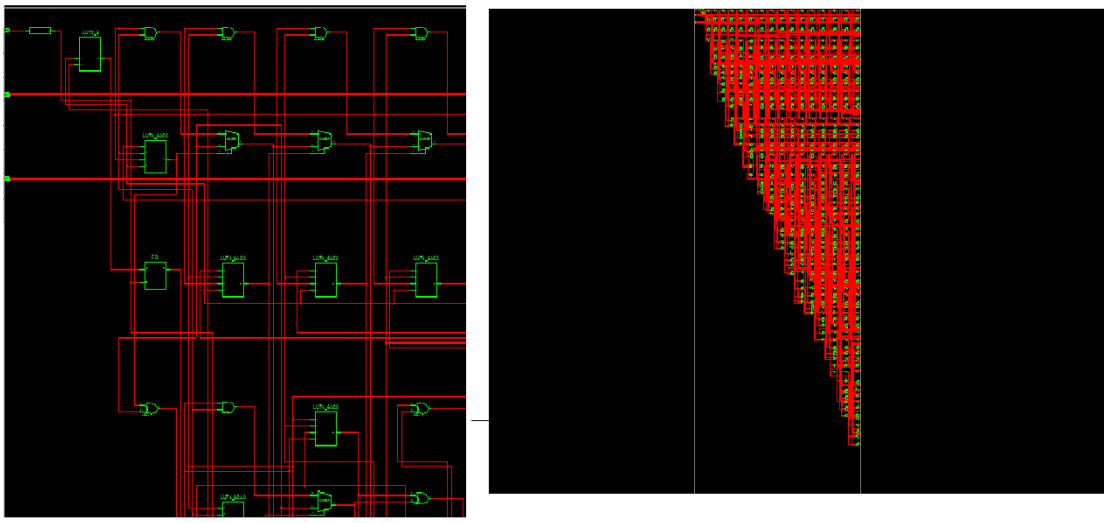


Figura 5.3: *Technology Schematic* del multiplicador

En la parte derecha de la figura 5.3 se observa el *Technology Schematic* del circuito multiplicador con las respectivas etapas de pipeline,además de está topologia Xilinx ofrece otras diferentes que presentan iguales caracteristicas de frecuencia y segmentación. En la parte izquierda se realiza un zoon al *Technology Schematic* y se aprecia mas detalladamente los elementos que conforman el diseño que componen cada etapa de la segmentación como: *LUTs*, *Flip-Flops*, compuertas logicas, *Mux*, etc.

5.2.4 Paso 4. Seleccionar el número de etapas de pipeline que optimizan el sistema.

Luego de haber construido el multiplicador, se procede a tabular los valores de latencia y throughput para diferentes números de etapas de segmentación. En la tabla 5.1 se observan estos valores, los cuales se obtienen apartir de reportes de la síntesis.

Etapas de pipeline	**Fmax**	**Latencia**	**troughput**
3	78.431[MHz]	57.438[ns]	17.41[MHz]
4	97.87[MHz]	66.464[ns]	=15.045[MHz]
5	130.959[MHz]	70.39[ns]	14.206[MHz]
6	130.959[MHz]	84.468[ns]	12.427[MHz]

Tabla 5.1: valores caracteristicos del multiplicador

Teniendo en cuenta los valores consignados en la tabla se toma la decisión de diseñar un multiplicador con 4 etapas de pipeline, aunque se puede obtener una mayor frecuencia con 5 o mas etapas, esto no mejoraría el rendimiento del circuito por qué la frecuencia con la que se generan los datos corresponde a 100MHZ y por esta razón con más de 4 etapas solo se aumentaría el valor de la latencia sin producir ningún beneficio.

LABORATORIO **6**

DISEÑO DIGITAL AVANZADO "UNIDAD DE CONTROL Y DATAPATH"

6.1 PROCEDIMIENTO GENERAL

Para realizar el diseño del procesador que permita la solución del ejercicio planteado se tuvo en cuenta la metodología para implementar algoritmos en hardware utilizando VHDL propuesta por los profesores de la Escuela de Ingeniería Electica, Electrónica y Telecomunicaciones MIE. Carlos A. Fajardo y MsC. Jorge H. Ramón. La cual esta definida en los siguientes pasos.

- Definir la entidad del procesador y el algoritmo a realizar.
- Diseñar el diagrama ASM.
- Diseñar el Datapath.
- Diseñar la FSM.
- Construcción del procesador usando VHDL.
- Simulación de los circuitos del diseño.

6.2 PROCEDIMIENTO PASO A PASO

A continuación se muestra detalladamente cada uno de los pasos para la realización del ejercicio propuesto.

6.2.1 Paso 1: Definir la entidad del procesador y el algoritmo a realizar

En la figura 6.1 se muestra la entidad del procesador que se diseña. Esta consta de 4 entradas: ent_n que es nuestro parámetro de entrada y las señales de inicio reset y el reloj. Además de las salidas: salida1 y

salida2 que son el numero enésimo de perrin y el primo de perrin correspondiente.

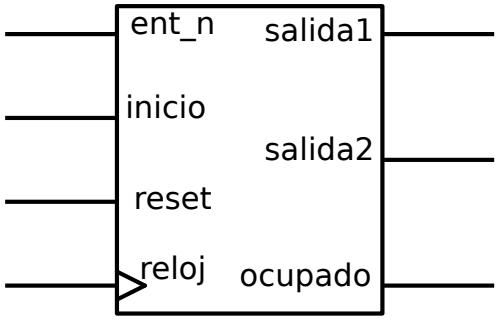


Figura 6.1: Entidad del procesador.

Teniendo la entidad ya definida, procedemos a realizar el algoritmo para desarrollar la sucesión de los números de perrin. En la figura 6.2 se observa el algoritmo paso a paso del ejercicio propuesto teniendo en cuenta las especificaciones de la actividad.

6.2.2 Paso 2: Diseñar el diagrama ASM

En este paso se diseña el diagrama ASM correspondiente al algoritmo de la figura 6.2, este permite hacer la transición de forma sencilla hacia la FSM, teniendo en cuenta que al diseñar el diagrama ASM debe incluir el mayor número de operaciones posibles en cada estado.En la figura 6.3 se observa el diagrama ASM del ejercicio planteado.

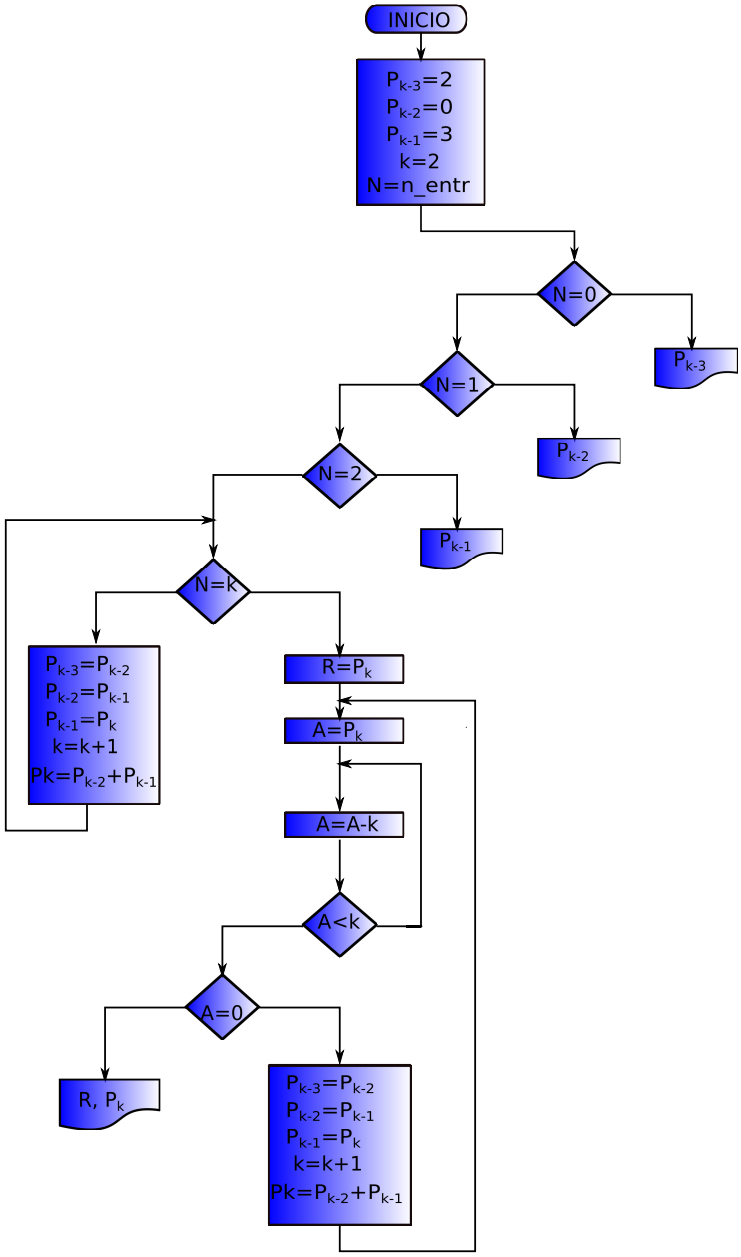


Figura 6.2: Algoritmo para resolver el ejercicio.

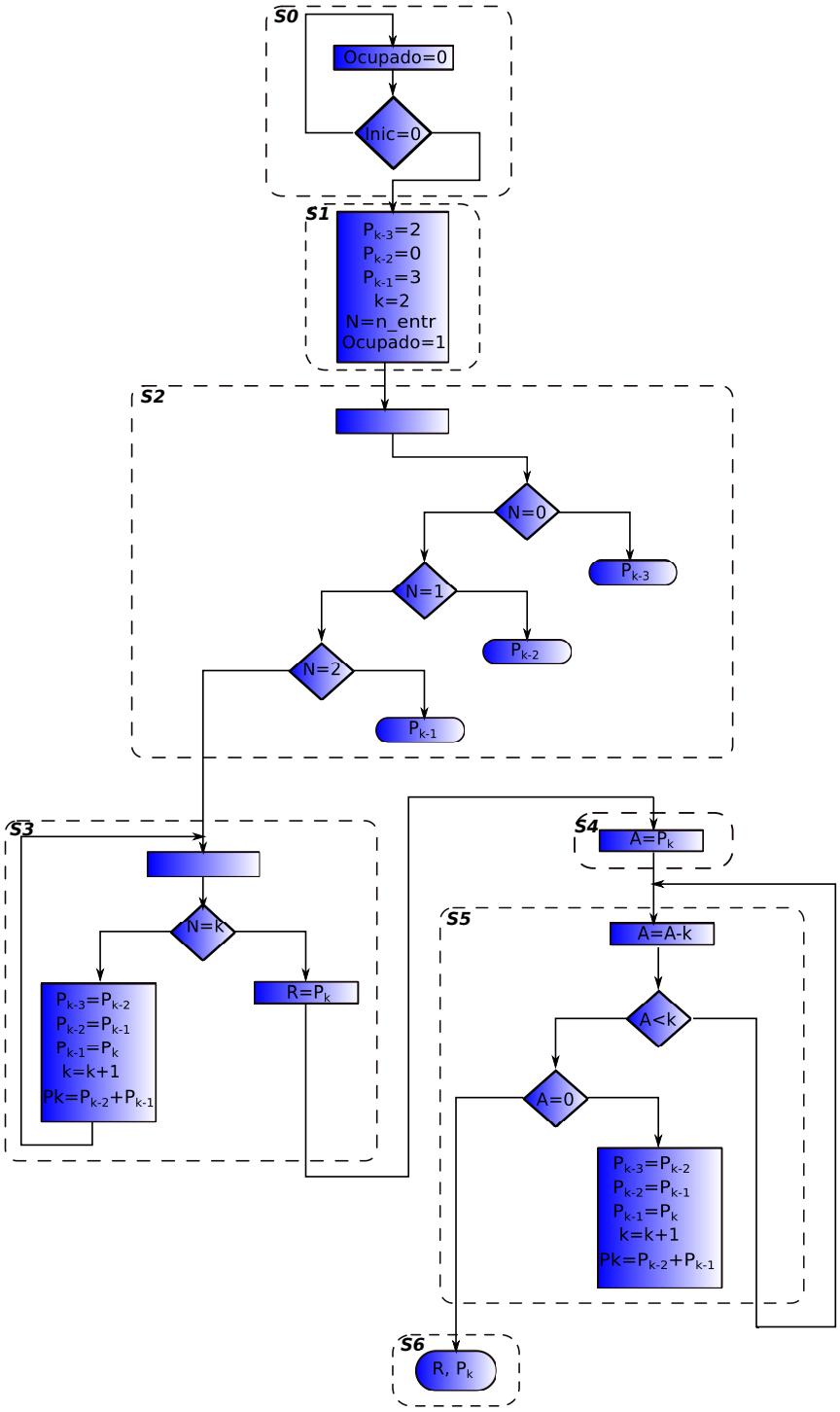


Figura 6.3: Diagrama ASM.

6.2.3 Paso 3: Diseñar el Datapath

Con el fin de desarrollar el diseño del datapath se propone realizar los siguientes ítems.

- **Tabla de uso de variables:** Se indica el tiempo de vida de cada variable con el fin de agrupar aquellas cuyos tiempos de vida no se solapen para usar la menor cantidad de registros.

	S0	S1	S2	S3	S4	S5	S6
N			X	X			
P_k				X	X	X	X
P_{k-3}			X	X	X	X	
P_{k-2}			X	X	X	X	
P_{k-1}			X	X	X	X	
K			X	X	X	X	
A						X	
R					X	X	X

Tabla 6.1: Uso de variables.

- **Asignación de los registros:** Con la información contenida en la tabla 6.1 se hace la asignación teniendo en cuenta que variables se pueden asignar al mismo registro.

  $N = [N, A]$
  $P_k = [\ P_k]$
  $P_{k-3} = [\ P_{k-3}]$
  $P_{k-2} = [\ P_{k-2}]$
  $P_{k-1} = [\ P_{k-1}]$
  $K = [K]$
  $R = [R]$

- **Agrupación de operaciones:** En la tabla de operaciones se determina de una manera sencilla cuantas unidades funcionales son necesaria para construir el datapath.

Operación	S0	S1	S2	S3	S4	S5	S6
+				X		X	
-						X	
+				X		X	

Tabla 6.2: Uso de operaciones.

- **Agrupación de las operaciones *RTL* de acuerdo a su registro de destino:** A continuación se realiza la agrupación de las operaciones para el diagrama ASM de la figura 6.2.

  Operación con el registro N

  $N \leftarrow$ n_entra en S1

  $N \leftarrow$ A en S4

  $N \leftarrow A - K$ en S5

  Operación con el registro P_k

  $P_k \leftarrow P_{k-2} + P_{k-3}$ en S3

  $P_k \leftarrow P_{k-2} + P_{k-3}$ en S5

  Operación con el registro P_{k-3}

  $P_{k-3} \leftarrow$ 2 en S1

  $P_{k-3} \leftarrow P_{k-2}$ en S3

  $P_{k-3} \leftarrow P_{k-2}$ en S5

  Operación con el registro P_{k-2}

  $P_{k-2} \leftarrow$ 0 en S1

  $P_{k-2} \leftarrow P_{k-1}$ en S3

  $P_{k-2} \leftarrow P_{k-1}$ en S5

  Operación con el registro P_{k-1}

  $P_{k-1} \leftarrow$ 3 en S1

  $P_{k-1} \leftarrow P_k$ en S3

  $P_{k-2} \leftarrow P_k$ en S3

  Operación con el registro K

  $K \leftarrow$ 2 en S1

  $K \leftarrow K + 1$ en S3

  $K \leftarrow K + 1$ en S5

  Operación con el registro R

  $R \leftarrow P_K$ en S1

- **Construcción del hardware:** En la figura 6.4 se muestra el datapath construido a partir de los pasos que se generaron anteriormente. Se puede identificar los registros, las unidades funcionales para cada operación asi como multiplexores para seleccionar los datos en los registros que se asocian con multiples operaciones RT.

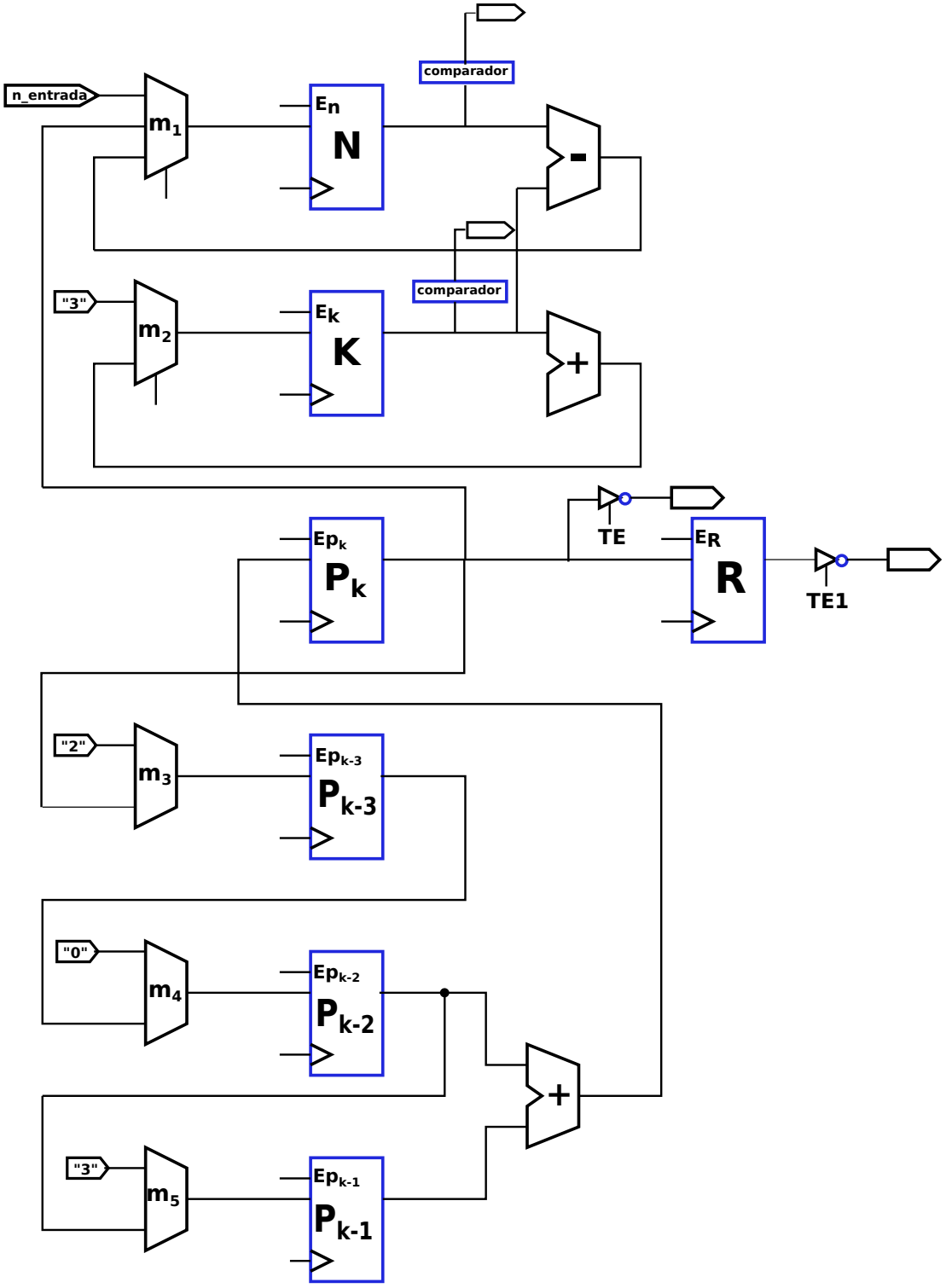


Figura 6.4: Datapath.

6.2.4 Paso 4: Diseño de la FSM

la FSM debe producir las señales de control de los multiplexores y de habilitación de los registros. Estas señales se agrupan en palabras de control que se generan en cada uno de los estados, teniendo en cuenta qué se debe implementar la palabra de control adecuada para realizar las operaciones RT de cada estado. a continuacon se observa en la tabala , cada una de las operaciones con su respectiva funcion en el estado en cual aparace.

OperRT	Ocup	TE	TE1	E_n	E_k	EP_k	EP_{k3}	EP_{k2}	EP_{k1}	ER	m1	m2	m3	m4	m5
$N \leftarrow$ n_entra $P_{k-3} \leftarrow 2$ $P_{k-2} \leftarrow 0$ $P_{k-1} \leftarrow 3$ $K \leftarrow 2$	1	0	0	1	1	0	1	1	1	0	00	0	0	0	0
$P_{k-3} \leftarrow P_{k-2}$ $P_k \leftarrow P_1$ $P_{k-2} \leftarrow P_{k-1}$ $P_{k-1} \leftarrow P_k$ $K \leftarrow K+1$ $R \leftarrow P_K$ $Sal1 \leftarrow P_K$	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1
$N \rightarrow$A	1	0	0	1	0	0	0	0	0	0	01	0	0	0	0
$N \rightarrow A-K$ $P_k \leftarrow P_1$ $P_{k-3} \leftarrow P_{k-2}$ $P_{k-2} \leftarrow P_{k-1}$ $P_{k-1} \leftarrow P_k$ $K \leftarrow K+1$	1	0	0	1	1	1	1	1	1	0	10	1	1	1	1
$Sal1 \leftarrow P_K$ $Sal2 \leftarrow R$	1	1	1	0	0	0	0	0	0	0	00	0	0	0	0

Tabla 6.3: Palabra de control para las operaciones RT.

En la tabla 6.3 $P_1 = P_{K-2} + P_{K-3}$

6.2.5 Paso 5: Construcción del procesador usando VHDL

En este paso se describe cada unidad funcional por separado (registros, sumadores, multiplexores, etc) usando VHDL, después se unen en un archivo en el que se realizan las conexiones para el correcto funcionamiento del procesador. Cabe recordar que es posible usar los *ip-cores* que ofrece ISE si no se desea describir todas la unidades. En el script 6.1 se observa la entidad del archivo donde se encuentrán contenidas todas las unidades funcionales.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity DATAPATH is
7      Port ( reloj                          : in  STD_LOGIC;
8             reset                          : in  STD_LOGIC;
9             En,Ek,Epk,Epk3,Epk2,Epk1,Er    : in  STD_LOGIC;
10            m2,m3,m4,m5                    : in  STD_LOGIC;
11            m1                             : in  STD_LOGIC_VECTOR (1 downto 0);
12            TE,TE1                         : in  STD_LOGIC;
13            ent_n                          : in  STD_LOGIC_VECTOR (7 downto 0);
14            bandera                        : out STD_LOGIC_VECTOR (1 downto 0);
15            c                              : out STD_LOGIC_VECTOR (2 downto 0);
16            sal1,sal2                      : out STD_LOGIC_VECTOR (7 downto 0));
17
18          attribute FSM_EXTRACT:string;
19          attribute FSM_EXTRACT of FSM:entity is "NO";
20 end DATAPATH;
```

Script 6.1: Código VHDL de la entidad del archivo Datapath.

6.2.6 Paso 6: Simulación de los circuitos del diseño

Dentro de este paso se comprueba el buen funcionamiento de los circuitos empleados, se observa el circuito funcionando correctamente, en la figura 6.5 se puede ver el valor N que se ingresa (N=8) y en la figura 6.6 la simulacion del circuito correspondiente al procesador de proposito especifo del problema, teniendo como salida a *salpk*= 29, lo que corresponde al número de perrin de ese valor en especifico.

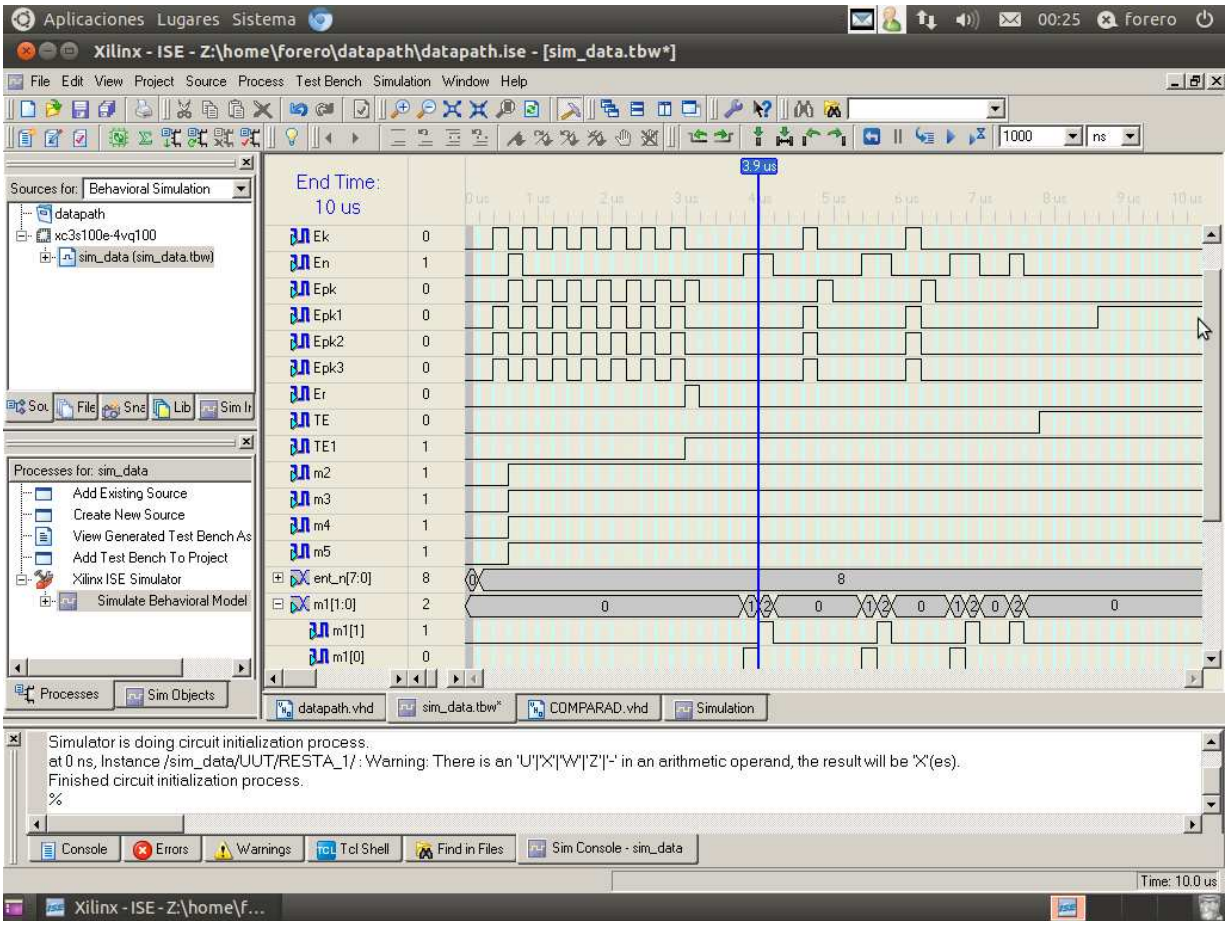


Figura 6.5: Entradas al circuito Datapath.

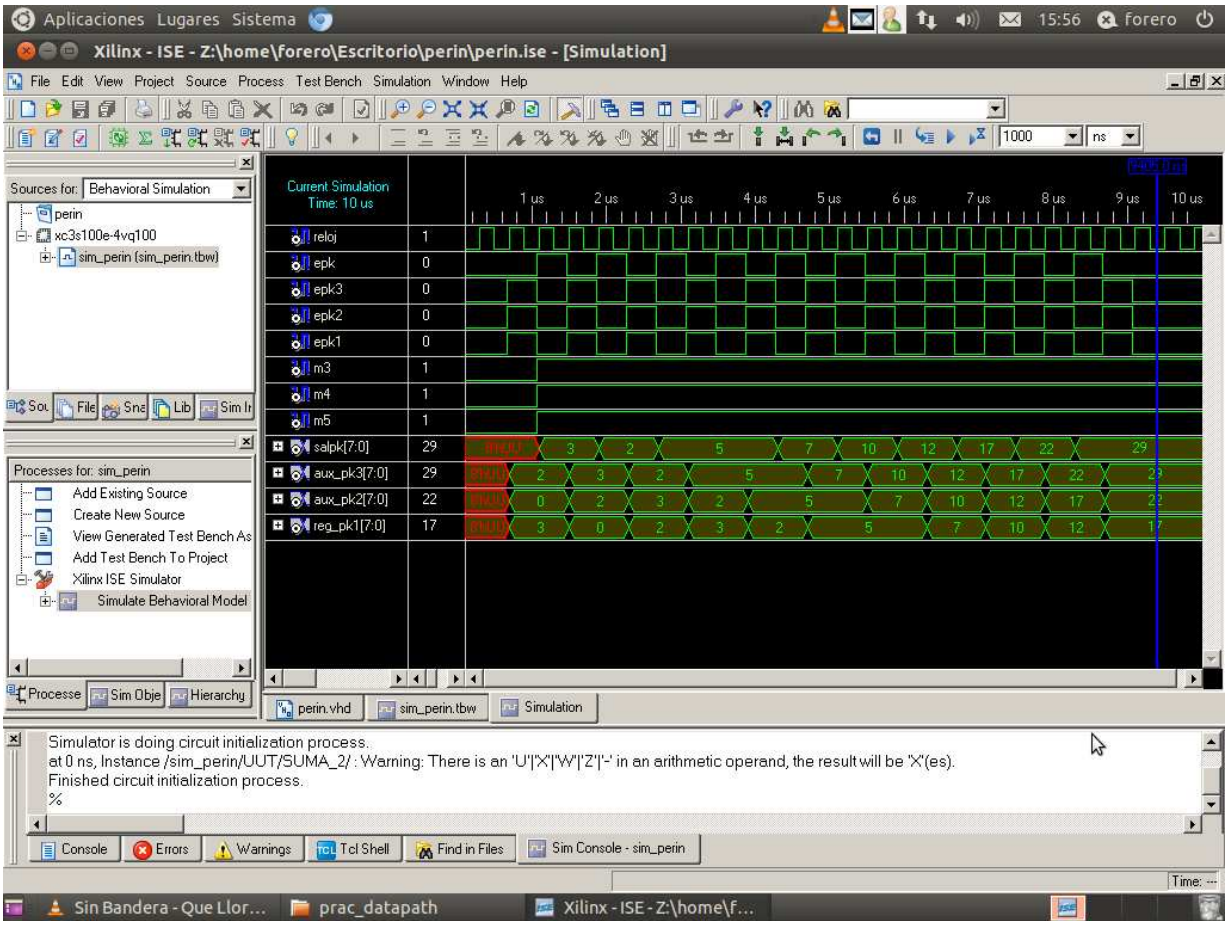


Figura 6.6: Simulacion del Datapath.