PROPUESTA DE PRÁCTICAS DE LABORATORIO PARA EL DISEÑO DE EMBEDDED SYSTEMS, BASADOS EN EL PROCESADOR MICROBLAZE; UTILIZANDO LA PLATAFORMA SPARTAN 3A DSP DE XILINX.

Carlos J. Jerez Acevedo

Sergio Reyes Ramos

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECANICAS

ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Bucaramanga 2009

PROPUESTA DE PRÁCTICAS DE LABORATORIO PARA EL DISEÑO DE EMBEDDED SYSTEMS, BASADOS EN EL PROCESADOR MICROBLAZE; UTILIZANDO LA PLATAFORMA SPARTAN 3A DSP DE XILINX.

Carlos J. Jerez Acevedo

Sergio Reyes Ramos

Trabajo de grado para optar al título de Ingeniero Electrónico

Director:

MIE (c) Carlos A. Fajardo Ariza

Co-Director:
MsC Jorge H. Ramón Suárez

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECANICAS

ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
Bucaramanga 2009

A nuestros padres por su constante apoyo,

A nuestras familias por su incansable aliento,

A nuestros compañeros y profesores por su ayuda, paciencia y motivación.

Agradecimientos

Agradecemos a cada una de las personas que a lo largo del proceso de formación como ingenieros electrónicos nos ayudaron o contribuyeron de alguna forma. De especial manera agradecemos a Carlos A. Fajardo por su orientación, motivación y apoyo durante la realización de este trabajo.

De igual forma a todos aquellos amigos y compañeros que de alguna u otra forma colaboraron con la construcción de este proyecto; entre ellos se destaca el apoyo de Eusebio Delgado, Andrés Cordero, William Méndez, Carlos Nieto, Ana María Olarte y Germán Betancourt.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Índice de Anexos

RESUMEN

TÍTULO: PROPUESTA DE PRÁCTICAS DE LABORATORIO PARA EL DISEÑO DE EMBEDDED SYSTEMS, BASADOS EN EL PROCESADOR MICROBLAZE; UTILIZANDO LA PLATAFORMA SPARTAN 3A DSP DE XILINX[1].

AUTORES: JEREZ ACEVEDO, Carlos Julio

REYES RAMOS, Sergio[2]

PALABRAS CLAVES: Embedded Systems, Sistemas Integrados, FPGA, Spartan, MicroBlaze, Xilkernel.

DESCRIPCION:

Durante la realización del presente trabajo se estudió el desarrollo de aplicaciones integradas implementadas sobre un FPGA y se desarrollaron las prácticas de laboratorio de Embedded Systems según los objetivos planteados en el plan de trabajo. Así mismo se elaboraron los documentos anexos que soportan las prácticas desarrolladas y los documentos de introducción al software de programación EDK (Embedded Development Kit).

El objetivo primordial de estos documentos es servir de apoyo a aquellas personas interesadas en el desarrollo de Embedded Systems (ES), brindándoles las herramientas necesarias para la creación de una serie de aplicaciones cuya complejidad va en aumento. Todo esto encauzado a la adquisición de un aprendizaje que permita al usuario continuar su desarrollo posteriormente.

Además se resalta la importancia que tienen los FPGA's en el desarrollo de ES, al proporcionar funcionalidades como flexibilidad, bajo consumo de potencia, reconfiguración y corto tiempo de desarrollo. Adicionalmente la tecnología de lógica programable cuenta actualmente con la incursión de procesadores soft-core, como el MicroBlaze, los cuales permiten que las aplicaciones integradas encuentren una forma de repartir los problemas de diseño tanto en hardware como en software, lo cual está brindando mayores posibilidades a los diseñadores.

Conjuntamente se da una introducción al uso de un Sistema Operativo en el diseño de ES, el cual facilita el manejo del hardware y permite al programador enfocarse en el desarrollo de la aplicación software.

[1] Trabajo de Grado

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: MIE (c) Carlos A. Fajardo Ariza. Co-Director: MsC Jorge H. Ramón Suarez

ABSTRACT

TÍTLE: PROPOSAL OF LABORATORY PRACTICES FOR EMBEDDED SYSTEMS DESIGNS, MICROBLAZE PROCESSOR-BASED, USING XILINX'S SPARTAN 3A DSP PLATFORM[3].

AUTHORS: JEREZ ACEVEDO, Carlos Julio

REYES RAMOS, Sergio[4]

KEYWORDS: Embedded Systems, FPGA, Spartan, MicroBlaze, Xilkernel.

DESCRIPTION:

During the completion of this work, it studied the development of embedded applications implemented on FPGA and developed embedded system laboratory practices according to the objectives outlined in the roadmap. Also attached documents were created to support the practices developed and documents of introduction to programming software EDK (Embedded Development Kit).

The primary objective of these papers is to provide support for those interested in developing embedded systems, providing the tools necessary to create a series of applications whose complexity is increasing. This channeled to acquire a domain that allows the user to continue its further development.

Besides, emphasizing the importance of FPGA's in the development of embedded systems by providing features such as flexibility, low power consumption, reconfiguration and short development time. Additionally, the programmable logic technology currently has the incursion of soft-core processors, such as MicroBlaze, those are enabling embedded applications to find a way to distribute design problems in hardware and software, which is providing greater opportunities for designers.

In addition, it provides an introduction to the use of an Operating System in the design of ES, which facilitates the management of hardware and allows the programmer to focus on software application development.

[3] Degree Work

[4] Faculty of Phisical-Mechanical. Engineering Electrical, Electronic and Telecommunications School. Director: MIE (c) Cesar A. Fajardo Ariza. Co-Director: Jorge H. Ramón Suarez.

Capitulo 1

1 Introducción

EMBEDDED SYSTEMS INPLEMENTADOS EN FPGA’s

Los desarrollos en la electrónica que hemos observado en los últimos años han impulsado gran número de tecnologías que solucionan los problemas de una humanidad en continua evolución.

Los Embedded Systems (ES) han surgido como una solución viable, estos se encuentran presente en nuestra vida respondiendo a las necesidades actuales, encontrándose en hornos microondas, autos, equipo de audio, celulares, teléfonos, televisor, reproductores de audio, refrigeradores, etc. La orientación de estos sistemas es hacia la minimización de costos y maximizar la confiabilidad [1], se puede decir que un ES es una combinación de hardware y software diseñados para tener una función o tarea específica, es por esto que generalmente presentan beneficios en cuanto a costos y consumo de potencia, comparados con los computadores (PC) tradicionales, donde pocas veces se llega a aprovechar al máximo toda su funcionalidad.

Siguiendo estos lineamientos los ES se han enfocado hacia la flexibilidad en el diseño y bajo consumo de potencia. Es así como las FPGA's se están usando como base de los mismos al responder de manera satisfactoria con los requerimientos que se están presentando.

Los FPGA's al tratarse de dispositivos programables permiten la reconfiguración del sistema si es necesario, esto añadido a la posibilidad de usar bloques de memoria, bloques DSP[5] e incluso hasta procesadores, todo esto dentro del mismo integrado [2].

El uso de procesadores dentro de los ES ha capturado el interés de la industria como de la academia, tanto así que cada día encontramos mas dispositivos electrónicos de este tipo que realizan tareas para nosotros, y las universidades incluyen asignaturas referentes a esta topología de diseño, siendo una parte esencial de estas la elaboración de prácticas de laboratorio que demuestren las capacidades y posibilidades que los ES pueden ofrecer.

[5] Digital Signal Processing

Capitulo 2

2 Conceptos Básicos

2.1. Embedded System

Muchas definiciones pueden darse de lo que es un Embedded System (ES), pero resumiendo estas ideas se puede decir que es un sistema digital basado en un procesado con una combinación entre software y hardware destinada a cumplir con un rango de tareas específicas [3][4][5].

Esta característica los diferencia de los sistemas computacionales convencionales, que poseen gran capacidad de procesamiento pero el usuario no aprovecha todo su potencial. En cambio los ES tienen una capacidad acorde con la tarea a realizar, sin desperdiciar recurso ni dinero, características que ha aprovechado la industria para su producción en masa y para su inserción en todo campo de la tecnología.[6][7].

2.2. Periférico

Un ES tiene que comunicarse con el exterior, esto lo realiza por medio de lo comúnmente se conoce como periférico. Se llama periférico a todo dispositivo o componente que no forma parte del núcleo básico de un ordenado, conectado mediante una interfaz entrada/salida, su propósito es, ofrecer al usuario una representación visual (ejem, monitor, etc),almacenar información en forma permanente (ejem, Disco duro, Memoria USB,DVD-ROM, etc), recibe información específica (ejem, sensores, etc) [8][9].

2.3. Memoria

La memoria es una parte muy importante para los ES, algunas de funciones que esta cumple son: proveer almacenamiento para la aplicación software, almacenamiento para datos como variables, resultados intermedios e información de estado, etc.

Hay varios tipos de memoria: [10]

- RAM (memoria de acceso aleatorio): Esta se define comúnmente como memoria principal. Cuando es utilizada por sí misma, el término RAM se refiere a memoria de lectura y escritura; es decir, se puede escribir datos en RAM como leerlos de RAM. Esto está en contraste a la ROM, que le permite solo hacer lectura de los datos. La mayoría de la RAM es volátil, lo que significa que requiere un flujo constante de la electricidad para mantener su contenido. Tan pronto como el suministro de poder sea interrumpido, todos los datos que estaban en RAM se pierden.

- ROM (memoria inalterable): Los ordenadores contienen casi siempre una cantidad pequeña de memoria de solo lectura que guarde las instrucciones para iniciar el ordenador. En la memoria ROM no se puede escribir.
- PROM (memoria inalterable programable): Un PROM es un chip de memoria en la cual usted puede salvar un programa. Pero una vez que se haya utilizado el PROM, usted no puede reusarlo para salvar algo más. Como las ROM, los PROMS son permanentes.
- EPROM (memoria inalterable programable borrable): Un EPROM es un tipo especial de PROM que puede ser borrado exponiéndolo a la luz ultravioleta.
- EEPROM (memoria eléctricamente inalterable programable borrable): Un EEPROM es un tipo especial de PROM que puede ser borrado exponiéndolo a una carga eléctrica.

2.4. FPGA[6]

Un FPGA es un dispositivo semiconductor que consta de arreglos lógicos en bloques iguales configurables llamados CLB's[7] que se interconectan por medio de una matriz de interconexión programable. La ventaja es que estos bloques se pueden configurar y reprogramar para realizar todo tipo de aplicaciones o diseños combinacionales o secuenciales, según la necesidad del diseñador. La mayoría de FPGA's actúan como una memoria RAM, al momento de retirar la alimentación del sistema la configuración se borra, ya que sus elementos de configuración son biestables, utilizando los bloques lógicos configurables CLB's [11].

[6] Field Programmable Gate Arrays
[7] Configurable Logic Blocks

Paralelamente con los dispositivos lógico programables simples (PAL[8], GAL[9]) y con los dispositivos lógicos programables complejos (CPLD's[10]) surgió el FPGA por medio de Signetics[11] a finales de los setenta, permitiendo gran funcionalidad y flexibilidad en dispositivos compuestos de interconexiones lógicas.

El número de bloques CLB's, precio y características varían dependiendo del dispositivo y del fabricante. La arquitectura de CLB's puede contener desde 64 hasta 8 millones de celdas lógicas idénticas; estas celdas lógicas están definidas para crear funciones lógicas programables y están compuestas de un Look Up Table (LUT) de cuatro entradas y un Flip-Flop.

Son tres los elementos esenciales de un FPGA, los Bloque Lógicos Configurables (CLB's), los bloques Entradas/Salida (IOB's) y la Matriz de interconexión Programables (PSM's), además dependiendo del diseño del FPGA algunos utilizan bloques RAM e incluyen otros bloques [12].

El FPGA se programa por la carga de celdas de memoria de configuración, que controla la lógica e interconexiones de los bloques CLB's, IOB's y PSM's [13]. Existe herramientas CAD[12] que programan y definen las conexiones y las celdas lógicas, estos traducen el código de lenguaje (HDL[13]) de programación y lo compilan.

[8] Programmable Array Logic
[9] Generic Array Logic
[10] Complex Programmable Logic Device
[11] Empresa dedicada a la creación de semiconductores, fundada por en 1961 y una de las primeras compañías creadas para fabricar circuitos integrados.
[12] Computer Asisted Design
[13] Hardware Description Languaje

Para configuración el FPGA se puede realizar de diversas maneras, Master, Slave, o incluso otras. Estas configuraciones se pueden realizar con comunicación serial, paralela o incluso, dependiendo del dispositivo, usar puerto USB para la configuración si se cuenta con el soporte y el software necesario.

Como se observa el FPGA es una buena opción al momento del diseño y la implementación de un ES, por contar con características indispensables que han hecho que la industria se interese en estos dispositivos, apoyando así, la revolución tecnológica que estamos viviendo.

2.5. Plataforma de Desarrollo SPARTAN 3A DSP 1800

La SPARTAN 3A DSP 1800 es una plataforma de desarrollo de bajo costo y gran desempeño que ofrece la empresa Xilinx[14], puede ser utilizada en diversas aplicaciones donde se requiera flexibilidad, desempeño y bajo consumo de potencia. Esta tarjeta contiene periféricos internos y puertos o ranuras estándar de expansión para la inclusión de periféricos externos, esta característica los hace ideal para diseños académicos.

El FPGA XC3SD1800A es el componente principal de la plataforma con 37440 celdas lógicas, estas puertas lógicas son biestables, razón por la cual lo hace reconfigurable, se compone de cinco elementos que son el bloque XtremeDSP DSP48A, bloque RAM[15], bloque lógico configurable (CLB´s), bloque entradas/salida (I/O), bloque administradores de reloj digital (DCM´s).

[14] Empresa líder en soluciones lógicas programables.
[15] Random Access Memory

Otros componentes principales que posee la plataforma, cuatro fuentes de reloj, Oscilador 125 MHz, Oscilador 25.1725Mhz (para Video) oscilador 25 MHz (para Ethernet) y un socket para oscilador externo; tres memorias, memoria DDR2 SDRAM 128MB, Memoria Parallel/BPI Flash 16Mx8, memoria serial/SPI Flash 64Mb; 8 leds; 8 Dip Switches; 4 Botones; Boton de reset; Conector de entrada/salida, dos conector 6 pines, 2 conectores de expansión, conector GPIO de 30 pines.

2.6. Procesadores IP (Propiedad Intelectual)

La propiedad Intelectual IP comprenden la protección legal sobre los productos intelectuales, la propiedad intelectual comprende las patentes, los derechos de autor, las marcas registradas y los secretos comerciales [14].

Los módulos IP, ofrece una solución para superar la barrera existente entre el numero de bloques lógicos disponibles y la productividad de los diseños. Esos módulos son reutilizables permitiendo a las empresas avanzar más rápidos y eficientemente en sus investigaciones.

Entre los bloques IP se encuentra los bloques hard-core, son bloques pocos flexibles integrados en el silicio del dispositivo; los bloques firm-core la descripción es a nivel de puertas y biestables (netlist), optimizada para dispositivos o familias de dispositivos en particular; y finalmente los bloques soft-core que están descritos en lenguaje de descripción de hardware a nivel entre registros y su código representa entidades (hardware) sintetizable.

“Podemos clasificar los procesadores de acuerdo a su construcción. Procesador soft-core, el núcleo del microprocesador no se encuentra predefinido, así que no tiene un área dedicada dentro de la FPGA, se define configurando el FPGA y permite la modificación e integración de más funcionalidades. Procesador Hard-core el núcleo del microprocesador esta empotrado en la FPGA, y ocupa un área fija” [15]. Xilinx emplea procesadores soft-core (MicroBlaze y PicoBlaze) y procesadores hard-core basados en Power PC®.

2.7. Procesador Soft-core

Son procesadores escritos en lenguaje de descripción de hardware (HDL) preparados para ser sintetizados, por ejemplo en una FPGA o CPLD, los hay de de diversos tipos y tamaños, encontrándose procesadores complejos hasta de 32 bit’s. Su característica principal es que son flexibles, utilizando solo los recursos necesarios, por ejemplo realiza la configuración de una parte de la FPGA.

2.7.1. MicroBlaze

El MicroBlaze es un procesador soft-core de 32 bits tipo RISC[16], repertorio de instrucciones reducido y simples que se ejecutan en un ciclo de maquina haciendo más rápida la ejecución, en que las instrucciones de carga y almacenamiento acceden a la memoria por datos, y arquitectura Harvard[17], donde se separa físicamente el almacenamiento de datos e instrucciones por buses separados. Este procesador es propiedad intelectual de Xilinx y dedicado para sus plataformas de desarrollo, optimizado para implementarlo en una FPGA en sus plataformas Spartan y Virtex, la ventaja de este procesador es que es altamente

[16] Reduced Instruction Set Computer
[17] El término se origina en las computadoras Harvard Mark I.

configurable, permitiendo flexibilidad en los diseños en aplicaciones industriales y académicas. [16]

Su estructura básica consta de 32 registros de propósitos generales, una unidad de desplazamiento, interfaz de bus de dos niveles de interrupción, una unidad aritmética lógica (ALU), decodificador de instrucciones, unidad de administración de la memoria (MMU), contador de programa.

Los buses de instrucciones y de datos se encuentran separados y son: el bus LMB (bus de memoria local), PLB (bus local del procesador), el OPB (bus de periféricos en chip) y el XCL (Xilinx Cache Link). Contiene además una interfaz maestra (MFSL) y una interfaz esclava (SFSL).

2.8. Sistema operativo

El sistema operativo es el software principal e importante de un computador su propósito es el reconocimiento y administrador de recursos (procesador, memoria, discos, ratón o archivos) [17]. Es el intermediario entre el usuario (Usuario de órdenes, programador, diseñador, administrador de sistema) y el hardware; ejemplos de sistemas operativos, Windows, Unix, Linux, DOS, Mac OS, etc.

Cuando se instala un nuevo hardware a la CPU se debe instalar un controlador (Driver) para el reconocimiento de la aplicación, el sistema operativo se encarga de administrar este driver para una buena interacción y funcionamiento, permitiendo fiabilidad ya que si se produce algún error o ciertos errores el sistema se encarga de arreglarlos sin que el usuario final se entere o informe, puede adicionalmente administrar simultáneamente diferentes aplicaciones.

Los Sistema operativos se clasifican de la siguiente forma:

Multiusuario: Permite que dos o más usuario interactúen con el sistema simultáneamente.

Multiprocesador: Soporta el abrir un mismo programa en más de una CPU[18].

Multitarea: Dos o más programas se ejecutan simultáneamente.

Multitramo: Un programa se divide en partes, el multitramo permite la ejecución simultánea de estas partes del programa.

Tiempo real: Permite la ejecución en tiempo real, respondiendo inmediatamente a las entradas.

2.8.1. Kernel

Es el núcleo del sistema operativo, actúa como un programa individual instalado en el sistema operativo que siempre se encuentra cargado en la memoria principal del ordenador, ejecutándose permanentemente, se encarga de la comunicación entre el software y el hardware, decide el acceso y manejo de los recursos del sistema de comunicación entre los dispositivos, encargándose de todas la operaciones y dialogo que se dan entre el hardware y software.

Para poder dialogar con las aplicaciones el Kernel utiliza una interfaz de sistema llamado syscall[19], mediante esta llamada las aplicaciones podrán pedir al sistema operativo que realice tareas en su nombre, haciendo así el único punto de entrada que tiene los procesos de espacio de usuario al Kernel [18]

[18] Unidad Central de Procesamiento
[19] Interface para que las aplicaciones de espacio de usuario puedan comunicarse con el hardware.

El Kernel consta de dos partes fundamentales, la sección de control de procesos, encargada de asignar recursos, programas, procesos; y el control de dispositivo se encarga de supervisar la transferencia de datos entre memoria y los periféricos [19].

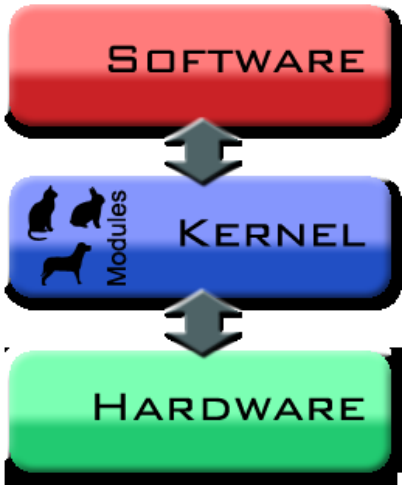

Figura 1. Posición del Kernel dentro de un Sistema **[20]**

Existen diferentes tipos de Kernel, estas dependen las funcionalidades y del tamaño del mismo [21].

- Monolítico; todas las funciones están integradas al sistema principal, por tal motivo se trata de un Kernel robusto y de tamaño considerable.
- Modular; Kernel que contiene las funciones esenciales para el funcionamiento del SO, dejando que controladores (Drivers) de periféricos u otros sistemas se adhieran al Kernel posteriormente, según sea necesario. Esta es la línea hacia donde se están desarrollando esta tecnología.
- Microkernel; Kernel consistido en una abstracción muy simple sobre el hardware, con un conjunto de funciones primordiales para implementar los

servicios mínimos en un SO, como la gestión de hilos (Multithreading) y la comunicación entre procesos.

2.8.1.1. XILKERNEL

Es un microkernel modular y robusto provisto por Xilinx, para trabajar desde el Xilinx Plattform Studio (XPS), actualmente está disponible la versión 4.00.a y está integrado con la descarga del software del EDK. [22]

- Permite un alto nivel de modificación, dejando cambiar a gusto del usuario para optimizarlo en tamaño y funciones.
- Con un POSIX API, soporta núcleos esenciales para un microkernel como trabajo con hilos (Thread's), sincronización (semáforos y mutex, timers e interrupciones).
- Trabaja en procesadores MicroBlaze, PowerPC405 y PowerPC440.

2.9. Embedded Development Kit (EDK)

Paquete de software suministrada por Xilinx para el trabajo con ES sobre sus plataformas de desarrollo (SPARTAN, VIRTEX). Compuesto por archivos y librerías destinados a facilitar la configuración e implementación de la plataforma, el procesador, periféricos a usar, aplicación software, etc.

La interfaz de trabajo, Xilinx Platform Studio o XPS, brinda un entorno agradable para el desarrollo de las funciones mencionadas anteriormente, permitiendo además la programación de plataformas.

Capitulo 3

3 Arquitectura General y Flujo de Diseño

3.1. Arquitectura de trabajo

El objetivo general de estas prácticas de laboratorio es guiar al estudiante para que esté en la capacidad de diseñar un Embedded System, sobre un FPGA. La arquitectura general seleccionada para el desarrollo de estos ES consta de un procesador soft-core al cual se le adicionan periféricos de acuerdo a las necesidades del usuario, algunos de ellos por medio de controladores descritos en VHDL.

En el siguiente diagrama se muestra un esquema general de la arquitectura utilizada:

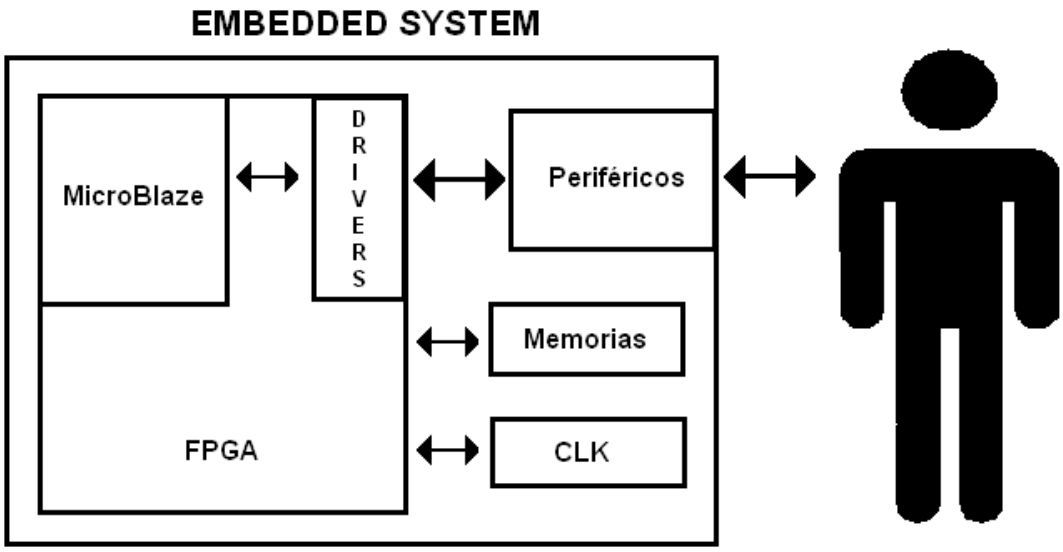

Figura 2. Arquitectura General de Diseño

Como se observa en la figura anterior el FPGA es el núcleo del sistema al contener el procesador y los controladores necesarios para los periféricos usados, además maneja la comunicación con partes esenciales del sistema como memorias y osciladores.

Los controladores de periféricos juegan un papel importante para el usuario, ya estos hacen de puente o comunicación entre el procesador o el programa y el usuario, permitiendo así la interacción del mismo con el ES. Sin olvidarnos de las memorias, osciladores y demás complementos esenciales para el funcionamiento del sistema.

Las plataforma de desarrollo, ofrecen todos estos recursos necesarios a la hora de implementar un ES, incluyendo periféricos básicos como LED's, botones,

comunicación serial, Ethernet, etc. Aprovechando además el soporte de software y recursos que ofrecen los fabricantes para el trabajo con las mismas.

Como se mencionó en al capítulo anterior la plataforma seleccionada para el trabajo fue la SPARTAN 3A DSP 1800A de Xilinx, sistema de desarrollo disponible en la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones, en conjunto con el software suministrado Xilinx ISE y EDK. Esta plataforma contiene las siguientes características:

• FPGA XC3SD1800A (Xilinx)

• Relojes

- Oscilador 125 MHz, LVTTL
- Socket para oscilador LVTTL
- Oscilador 25.175 MHz LVTTL (Video clock)
- Oscilador 25 MHz Ethernet

• Memorias

- 128 MB (32M x 32) DDR2 SDRAM
- 16Mx8 Parallel / BPI Flash (Configuración)
- 64 Mb SPI Flash (Almacenamiento/Configuración)

• Interfaces

- 10/100/1000 PHY (Ethernet)
- Puerto JTAG (Configuración/Programación)

- Puerto Serial (RS232)
- Puerto VGA
- 4 Líneas SPI

• Botones e Interruptores

- 8 LED's
- 8 DIP Switch's
- 4 Botones
- Botón RESET

• Entradas/Salidas (I/O)

- Conector 6-pines Digilent (2)
- Conector de expansión (2)
- Conector GPIO de 30 pines

El microprocesador soft-core soportado por la plataforma es el MicroBlaze, compuesto por un set de instrucciones reducido (RISC) desarrollado por Xilinx, y optimizado para el desarrollo de aplicaciones en sus FPGA's (Spartan, Virtex, etc).

El microprocesador MicroBlaze es altamente configurable, permitiendo gran flexibilidad a la hora de realizar todo tipo de diseño. Dentro de las principales características de este Soft-Core están:

- 32 registros multipropósito de 32 bit's de longitud
- Instrucciones con palabras de 32 bit's, incluyendo hasta tres operando con dos tipos de direccionamiento

- Bus de dirección de 32 bit’s
- Arquitectura interna tipo Harvard
- Funcionalidad Pipeline (posibilidad de operar varias tareas al mismo tiempo de ejecución)

3.2. Flujo de Diseño

La metodología utilizada en el desarrollo de cada uno de los ES implementados en este proyecto, con el fin de ser propuestos como prácticas de la laboratorio que faciliten el aprendizaje significativo de las competencias necesarias a la hora de diseñar un ES sobre un FPGA, se presenta de forma estructurada y secuencial, buscando que el estudiante se familiarice rápidamente con la metodología y así pueda desarrollar aplicaciones complejas con mayor facilidad.

A continuación se da una breve descripción de los pasos usados para la implementación de los ES en las prácticas:

3.2.1. Definición de Objetivos.

En esta etapa se define y limita los alcances del sistema a diseñar. Además se identifican que periféricos son necesarios para cumplir con los requerimientos del sistema y se tiene en cuenta la necesidad de describir los periféricos no disponibles en la plataforma.

3.2.2. Separar Hardware y Software en el ES.

Aquí se determina que procesos y funciones del sistema se implementaran hardware y cuáles en software. Teniendo en cuenta principalmente los requerimientos de tiempo y el descargar al procesador de algunas tareas específicas.

3.2.3. Configuración del MicroBlaze.

En la creación del sistema es necesario configurar algunas características del microprocesador, entre ellas se encuentran la velocidad del reloj de procesamiento, memorias, controladores de memoria, módulos de punto flotante, entre otros.

3.2.4. Descripción de Periféricos Externos.

Si dentro del sistema se usa uno o más periféricos externos y este precisa algún tipo especial de control, es necesario describir en VHDL un modulo para implementarse en la FPGA que se encargue de la comunicación entre el exterior y el procesador.

3.2.5. Adición de Periféricos.

Luego de tener los controladores de los periféricos externos, estos se agregan al procesador, así como los periféricos locales, por medio de los buses destinados para este fin (PLB, Processor Local Bus). Con los periféricos añadidos se procede a su configuración y a realizar las respectivas conexiones necesarias.

3.2.6. Síntesis[20] del Proyecto.

La síntesis es la creación de los archivos que contienen la información necesaria para hacer las interconexiones en el FPGA, correspondientes al hardware del ES. Estos contienen la información del microprocesador, los módulos de los periféricos, la distribución de memoria local, interconexiones entre periféricos y el microprocesador, etc. Es muy importante al generar estos archivos asegurarse de incluir todo lo necesario para el sistema.

[20] Método que procede de lo simple a lo compuesto, integración de las partes aisladas en un conjunto que unifique todos los elementos [23]

3.2.7. Aplicación Software.

Esta aplicación software realizada en lenguaje C, es la interfaz de control entre el usuario y la máquina, es en este punto donde se inicializan los periféricos y se programan para realizar las tareas requeridas. Además al tratarse de un lenguaje de alto nivel, facilita al usuario la programación de diferentes tareas.

3.2.8. Implementación

Completada las partes hardware y software del sistema se procede a la implementación del sistema para su ejecución. Por medio del cable de programación se descargan los archivos de programación a la plataforma para que el software se encargue de implementar el sistema en el FPGA.

Capitulo 4

4 Practicas a Implementar

El propósito de estas prácticas junto con los documentos de apoyo generados en este proyecto[21], es guiar al estudiante a través de una serie de ejercicios de diseño que le permitan ir adquiriendo conocimientos claves a la hora de diseñar un ES sobre un FPGA.

Las prácticas has sido concebidas de tal forma que vayan cubriendo las diferentes etapas que componen el diseño de ES sobre esta tecnología, sin que se pretenda cumplir con todas las posibilidades de diseño que ofrece la plataforma de desarrollo utilizada en este proyecto, sino mas bien facilitar al estudiante adquirir un aprendizaje significativo básico que mas adelante él mismo pueda ampliar. La filosofía principal de estas prácticas es aprender haciendo.

[21] En proyecto junto con las cinco guías de las practican se generaron los documentos: Ins

A continuación se describe cada una de las prácticas seleccionadas.

4.1. Introducción al EDK y MicroBlaze.

Las tendencias de diseño integrado están usando el recurso de un microcontrolador o microprocesador soft-core encargado de realizar las operaciones de cálculo, sincronización, etc. Estos implementados sobre un dispositivo lógico programable como CPLD's o FPGA's están respondiendo a las necesidades de hardware y software que plantean los problemas de hoy en día.

El MicroBlaze, microprocesador proporcionado por Xilinx para el desarrollo en sus plataformas SPARTAN, VIRTEX, etc; es una herramienta muy útil y flexible para el desarrollo de ES, y este, al ser altamente configurable desde el software de EDK permite un diseño más intuitivo y simplificado.

La primera meta del estudiante es conocer e identificar las configuraciones del MicroBlaze desde el EDK, por medio de la creación de un proyecto sencillo como lo es el encendido y apagado de los LED's disponibles en la plataforma.

Al finalizar esta práctica el estudiante estará en capacidad de:

- Crear un sistema base dentro del software XPS, para la implementación de un ES en la plataforma de desarrollo SPARTAN 3A DSP.
- Realizar la configuración básica del procesador MicroBlaze para su utilización dentro de un diseño.
- Desarrollar aplicaciones software (Lenguaje C) para el funcionamiento de sistemas sencillos.

- Implementar en la plataforma el sistema creado mediante el cable de programación USB-JTAG.

4.2. Adición y uso de periféricos locales Microblaze.

Es fácil encontrar en los ES, desde los más simples a los de mayor complejidad, el uso de botones, interruptores e indicadores visuales; por este motivo y con el fin de ampliar las posibilidades en los diseños se propone una serie de aplicaciones para el uso de estos periféricos, los cuales están disponibles en la plataforma.

La primera aplicación incluye los LED's y botones, al pedir crear una aplicación donde se controle el encendido de los mismos por medio de los botones. Seguido se pide al estudiante que use como control del encendido de los LED's los interruptores (DIP Switch) de la plataforma.

Para finalizar se propone la creación de un contado ascendente/descendente visualizado nuevamente en los LED's, esta vez deberá tener tres controles dirigidos por botones, uno para incrementar, otro para el decremento y uno final para reiniciar cuenta a cero (Reset).

Al finalizar esta práctica el estudiante estará en capacidad de:

- Disponer de los recursos internos de la plataforma (Periféricos Locales) para el desarrollo de ES más elaborados.
- Configurar estos periféricos para el uso particular y eficiente de los mismos.
- Realizar interconexiones entre el procesador MicroBlaze y los periféricos utilizados.

4.3. Uso de periféricos externos (Módulo Display de 7 Segmentos).

Ya con un dominio formado en uso de periféricos añadidos al MicroBlaze, se propone como siguiente paso la adición de un periférico externo. Esta adición se realiza por medio de los puertos entrada/salida (I/O) de la plataforma, en este caso los conectores de 6 pines.

El periférico a integrar es el Digilent PmodSSD el cual es un display 7 segmentos doble, para esto es necesario que el estudiante mediante el asistente incorporado en el EDK para la creación de periféricos, describa en VHDL el control para la conversión BCD[22] - 7 segmentos. La meta de esta práctica consiste en el diseño de un proyecto en EDK con el cual se visualice en el modulo un contador ascendente de 00 a 99 con reinicio automático.

Al finalizar esta práctica el estudiante estará en capacidad de:

- Utilizar el asistente provisto por el software XPS para la creación un nuevo controlador (Driver).
- Manejar y modificar los archivos generados por el asistente con el fin de realizar el control necesario sobre el periférico añadido.
- Distinguir de acuerdo a las necesidades, la implementación hardware y software para aplicaciones sencillas.

4.4. Uso de periféricos externos (Módulo ADC).

Existen muchos ES en los cuales es indispensable la adquisición y procesamiento de datos, con el fin de implementar un sistema con estas características se propone la integración de periféricos internos y externos.

[22] Binary Coded Decimal

El sistema contiene el modulo PmodAD1 de Digilent, convertidos análogo – digital de 12 bit's, y uso de periféricos internos como botones, LED's, memoria RAM y el puerto de comunicación serial (RS232). Como se realizó en la práctica anterior, es necesario generar el controlador para el modulo ADC usado.

El sistema finalizado debe estar en la capacidad de recibir una señal análoga del exterior, por medio del modulo y su controlador enviar la señal binaria correspondiente a los LED's para su visualización en tiempo real. Además contar con dos controles por botones, uno para realizar un guardado finito de datos continuos en la memoria RAM y otro para iniciar el envió de las datos guardados a un puesto de trabajo (PC) por medio de la conexión serial.

Al finalizar esta práctica el estudiante estará en capacidad de:

- Integrar de forma correcta los periféricos locales disponibles en la plataforma con periféricos externos dentro del diseño de un ES.
- Dominar el procedimiento de generación de un controlador para periférico externo.
- Comunicar la plataforma de desarrollo con un sistema exterior (Computador).

4.5. Uso de periféricos desde el procesador a través de un sistema operativo (Xilkernel).

La complejidad en los diseños de sistemas integrados ha inducido a los programadores a usar un recurso que facilita la comunicación entre el hardware del sistema y la aplicación creada por el usuario. Este es el caso

de la utilización de un sistema operativo, cuya función es realizar una comunicación sincronizada entre el hardware y la interfaz de usuario, además de gestionar recursos y funciones por medio de estándares de programación.

Por este motivo cada vez se encuentra con más frecuencia el uso de sistemas operativos livianos, en su mayoría gratuitos, que ofrecen solución a la problemática anteriormente expuesta. Para estar al tanto de esta situación, Xilinx viene desarrollando desde hace unos años un microkernel, conocido comercialmente como Xilkernel, el cual está altamente adaptado al trabajo con sus sistemas de desarrollo, operando bajo microprocesadores como el MicroBlaze o PowerPC. Este microkernel consta de algunas de las funciones más relevantes del estándar POSIX de programación, como lo son el manejo de hilos de ejecución (Thread's), semáforos, memoria compartida, temporizadores, etc; características que abren un nuevo horizonte de diseños en ES.

Con el fin de introducir al estudiante en este tipo de diseño y analizar las ventajas que suministra el uso de un sistema operativo se propone la utilización del Xilkernel para observar mediante la comunicación por puerto serial, la operación de varios hilos de ejecución (Multithreading). Para esto se creara un proyecto en EDK que cumpla con este fin, mediante una aplicación software que controle el arranque del 41icros y la impresión de los mensajes en un computador por medio del puerto serial (RS232).

Al finalizar esta práctica el estudiante estará en capacidad de:

- Realizar la configuración básica para el funcionamiento del Xilkernel.
- Identificar las funciones que este microkernel ofrece para el trabajo con ES.
- Reconocer las ventajas que ofrece el uso de un sistema operativo.

- Implementar diseños en los cuales se utilice la característica multihilo que ofrece Xilkernel.

Los documentos de apoyo (Anexos) entregados junto con las prácticas son indispensables para la realización de las mismas, ya que estos proveen de guías de instalación, introducción y soporte de las herramientas software necesarias para la ejecución de los diseños propuestos en las practicas. Dentro de estos se encuentran los documentos “Instalación del software ISE y EDK”, “Introducción al XPS” y “Creación de un IP Core”. Como complemento a la quinta practica se entrega un documento donde se expone una descripción del funcionamiento del microkernel provisto por Xilinx, además se exhibe la configuración básica para el funcionamiento del mismo, esto se encuentra en el anexo titulado: “Xilkernel”.

Capitulo 5

5 Resumen de Prácticas

Nombre de la Práctica	Objetivos	Documentos y Referencias
Introducción al EDK y MicroBlaze.	**Objetivo General** Describir los principales pasos para el diseño de un sistema digital con la herramienta EDK, incorporando el procesador MicroBlaze. **Objetivos Específicos** Instalar el software requerido para el diseño de ES suministrado por Xilinx para sus diversas plataformas. Implementar un proyecto sencillo que permita reconocer y profundizar el manejo de la herramienta de software EDK (Xilinx Platform Studio). Construir un programa en lenguaje C que permita apagar y encender	• Anexo F, Instalación del software ISE y EDK. • Anexo G, Introducción al software XPS. • http://www.xilinx.com/43icro/microblaze.htm • http://www.xilinx.com/products/devkits/HW-SD1800A-DSP-SB-UNI-G.htm

	secuencialmente (Auto fantástico) los LED´s de la Plataforma SPARTAN 3A DSP. Programar la Plataforma de desarrollo SPARTAN 3ª DSP 1800, mediante el cable USB-JTA.	
Adición y uso de periféricos locales al Microblaze.	**Objetivo General** Usar los recursos internos con lo que cuenta la plataforma en el diseño de aplicaciones sencillas. **Objetivos Específicos** Seleccionar los IP Core (Periféricos) necesarios para la creación de tres proyectos a realizar, utilizando LED´s, interruptores y Pulsadores. Adicionar los periféricos locales al PLB (Processor Local Bus), de la plataforma Spartan 3A DSP. Escribir los códigos en lenguaje C para la implementación de los proyectos con los periféricos internos de la plataforma.	• Anexo G, Introducción al software XPS. • http://www.xilinx.com/support/documentation/ip_documentation/xps_gpio.pdf
Uso de periféricos externos (Modulo Display de 7 Segmentos).	**Objetivo General** Crear mediante el asistente de creación de periféricos que proporciona la herramienta EDK, un modulo IP para el control de un periférico externo. **Objetivos Específicos** Realizar e implementar un diseño en la	• Anexo H, Creación de un IP Core. • http://www.digilentinc.com/Data/Products/PMOD-SSD/Pmod%20SSD_rm.pdf

	plataforma que permita el control del modulo Digilent PmodSSD (Display 7 Segmentos Doble). Describir en VHDL un módulo que permita hacer la conversión BCD-7 segmentos, e introducirlo al Driver generado	
Uso de periféricos externos (Modulo ADC).	**Objetivo General** Crear un sistema que reciba una señal analógica, usando el ADC digitalice esta misma y se visualice en los LED's de la plataforma. Además el sistema debe contener la opción de almacenar un rango de datos para su posterior trasmisión a un computador, esto controlado por medio de Pulsadores. **Objetivos Específicos** Diseñar un código en VHDL que controle el modulo ADC para la adquisición de datos. Diseñar en EDK un proyecto que permita la adquisición de los datos, almacenaje, control y trasmisión de los mismos. Escribir en lenguaje C el código correspondiente para la implementación del proyecto descripto Integrar el modulo externo ADC PmodAD1 con módulos internos de la plataforma SPARTAN 3A DSP 1800A	• Anexo H, Creación de un IP Core. • http://www.dea.icai.upco.es/romano/sp/sisper_adc.pdf • http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,499&Prod=PMOD-AD1 • http://www.national.com/ds/DC/ADCS7476.pdf • http://www3.fi.mdp.edu.ar/electronica/tesis/Resumen-Carrica.pdf • http://www.digilentinc.com/Data/Products/PMOD-AD1/Pmod%20AD1_rm.pdf • http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf • http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf

Uso de periféricos desde el procesador a través de un sistema operativo (Xilkernel).	**Objetivo General** Crear un sistema en donde se aprecie el trabajo multihilo (Multithreading) del microkernel proporcionado por Xilinx (Xilkernel), operando con el procesador MicroBlaze. **Objetivos Específicos** Diseñar un proyecto en EDK que permita visualizar en el Hyperterminal [1], por medio del puerto serial (RS232), la ejecución de varios hilos de impresión. Configurar el Xilkernel para operar varios hilos de ejecución (Thread's) Escribir el código en C para controlar la ejecución del Xilkernel y la impresión de los thread's mediante comunicación serial.	• Anexo I, Xilkernel. • http://technet.microsoft.com/es-es/library/cc784492(WS.10).aspx • http://en.wikipedia.org/wiki/Operating_system • http://es.wikipedia.org/wiki/N%C3%Bacleo_(inform%C3%A1tica) • http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node60.html • http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

Tabla 1. Resumen de Prácticas

Capitulo 6

6 Conclusiones

- El aprendizaje mediante la realización de ejercicios de diseño que incrementen su complejidad se muestra como una opción de aprendizaje significativo en las competencias requeridas a la hora de diseñar un ES.
- La posibilidad de sintetizar el procesador MicroBlaze dentro de un FPGA ofrece la gran ventaja de dividir un ES en hardware y en software, con esto se logra usar los recursos disponibles de forma eficiente, logrando un alto rendimiento en el diseño implementado.
- Las posibilidades en el diseño de ES que actualmente ofrecen los dispositivos lógicos programables en conjunto con un sistema de desarrollo son muy amplias y no pueden ser cubiertas en número reducido de prácticas. Por lo tanto las prácticas entregadas forman parte de los primeros peldaños de trabajo con ES implementados en un FPGA, generando las herramientas necesarias y brindando una base para continuar la profundización en el diseño de ES.
- Los FPGA´s son alternativa eficiente a la hora de implementar hardware, ya que estos al configurarse mediante software es fácilmente reconfigurable y

reusable, además posee características que dentro del diseño de ES son muy relevantes como un alto desempeño, flexibilidad, etc.

- El uso de un Sistema Operativo (SO) dentro de un ES proporciona nuevas funcionalidades que apoyan al desarrollo de aplicaciones complejas. Estas funciones como el procesamiento multihilo, sistemas de sincronización o temporizadores, alivian la carga del diseñador hacia el hardware y manejo de recursos, permitiéndole concentrarse en la aplicación software.

- Se describió de forma sencilla las características y funciones del procesador MicroBlaze, así mismo para la plataforma de desarrollo con la que se trabajó, SPARTAN 3AA DSP 1800.

- La experiencia adquirida acerca del diseño de ES implementados sobre un FPGA durante la realización de este proyecto fue de gran valor al realizar la estructura de las practicas, tomando como ejemplo nuestro proceso de aprendizaje y plasmando de la mejor forma un camino para que los estudiantes se guíen dentro del diseño de ES.

- El soporte entregado por Xilinx en su pagina web, compuesto por documentos, ejemplos y foros, forman una herramienta esencial a la hora de diseñar sobre sus plataformas. Ya que por medio de estos recursos el usuario puede agilizar el proceso de diseño, corregir y entender posibles errores de programación, conocer experiencias de trabajo de otros usuarios, etc.

7 Referencias Bibliográficas

[1] http://www.ati.es/article.php3?id_article=772.

[2] http://en.wikipedia.org/wiki/Fpga

[3] http://www.idose.es/sistemas-embebidos

[4] http://www.actc-control.com/glossary/glossary_E.asp

[5] Peter Marwedel. Embedded System Design

[6] Steve Heath. Embedded Systems Design. Second Edition

[7] Michael Barr, Anthony Massa. Programming Embedded Systems with C and GNU Development Tools. Second Edition.

[8] http://es.kioskea.net/contents/pc/peripherique.php3

[9] Diccionario de la Real Academia Española. http://www.rae.es/rae.html

[10] http://www.monografias.com/trabajos/memoria/memoria.shtml

[11] http://www.xilinx.com/company/gettingstarted/index.htm.

[12] A. Castillo, J. Ortegón,C, Rodriguez, Prácticas de laboratorio para estudiantes de ingeniería con FPGA. IEEE LATIN AMERICA TRANSACTIONS, VOL. 6. Junio 2008.

[13] Gabriel Sánchez Suarez, Que es una FPGA. Universidad Francisco de Paula Santander.

[14] E. Castillo Morales. “PROCTECCION DE LA PROPIEDAD INTELECTUAL DE CIRCUITOS DIGITALES PARA LA SÍNTESIS DE DESCRIPCIONES DE ALTO NIVEL”. Tesis Doctoral, Departamento de Electrónica y tecnología de Computadores. Universidad de Granada. 2008.

[15] Moisés Serra, Oscar Navas, Jordi Escrip, Martí Bonamusa, Pere Marti, Jordi Carrabina. Metodología de prototipado rápido desde Matlab: herramientas visuales para flujo de datos. Departamento de Electrónica y Telecomunicaciones. Universidad Autónoma de Barcelona

[16] http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[17] http://www.frm.utn.edu.ar/soperativos/Archivos/procesos.pdf

[18] http://www.kernel-labs.org/?q=syscalls

[19] http://www.monografias.com/trabajos/unix/unix.shtml

[20]http://radio.flujos.org/index.php?option=com_content&view=article&id=116:kernelinfo&catid=61:sesiontrescapa&Itemid=91

[21] http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node60.html

[22]http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

[23] manuelgalan.blogspot.com/2008/12/definicion-de.html.

8 ANEXOS

Anexo A

PRÁCTICA NO 1

INTRODUCCIÓN AL EDK Y MICROBLAZE

INTRODUCCIÓN

El soporte software suministrado por Xilinx para el desarrollo de ES, para sus diversas plataformas, se convierte en una herramienta necesaria al proporcionar una manera sencilla para la implementación de todo tipo de proyectos mediante un diagrama de flujo de diseño simple.

La herramienta de software EDK (Embedded Development Kit) condensa todas las características y posibilidades que ofrecen las plataformas de diseño, para este caso la SPARTAN 3A DSP, y de una manera fácil seleccionar cuales de estas se van a utilizar. Mediante una forma didáctica se quiere dar a conocer los pasos necesarios para la creación de un proyecto (ES), permitiendo conocer la configuración y recursos básicos con los que cuenta plataforma SPARTAN 3A DSP 1800.

OBJETIVOS

OBJETIVO GENERAL

- Describir los principales pasos para el diseño de un sistema digital con la herramienta EDK, incorporando el procesador MicroBlaze.

OBJETIVOS ESPECIFICOS

- Instalar el software requerido para el diseño de ES suministrado por Xilinx para sus diversas plataformas.
- Implementar un proyecto sencillo que permita reconocer y profundizar el manejo de la herramienta de software EDK (Xilinx Platform Studio).
- Construir un programa en lenguaje C que permita apagar y encender secuencialmente (Auto fantástico) los LED´s de la Plataforma SPARTAN 3A DSP.
- Programar la Plataforma de desarrollo SPARTAN 3A DSP 1800, mediante el cable USB-JTAG.

TRABAJO PREVIO

Leer y estudiar los documentos: “Instalación del software ISE y EDK” (ANEXO F), e “Introducción al software XPS” (ANEXO G).

Revisar los documentos existentes en la página web oficial de Xilinx acerca del procesador MicroBlaze [1] y la plataforma de desarrollo SPARTAN-3A DSP [2].

Conocimiento básico en programación con lenguaje C; declaraciones, funciones, punteros, etc.

TIEMPO ESTIMADO

El tiempo estimado para la realización de esta guía de laboratorio es de 2 (dos) secciones de 2 (dos) horas, o equivalente a esta duración.

PROCEDIMIENTO

1. Con una copia del documento “Introducción al software XPS” (ANEXO G), realizar el procedimiento propuesto en este para la elaboración de un proyecto en EDK.
2. Identificar dentro del programa en lenguaje C entregado, sintaxis y declaraciones especiales para el trabajo en EDK.
3. Revisar dentro de la información suministrada por el IP Core GPIO (Api Documentation), el uso de las funciones encontradas y los parámetros que estas emplean.
4. Modificar el código entregado para implementar la secuencia “Auto Fantástico” en los LED’s de la plataforma.
5. Comprobar el funcionamiento del proyecto mediante la implementación del mismo sobre la plataforma SPARTAN 3A-DSP.

REFERENCIAS

[1] http://www.xilinx.com/tools/microblaze.htm

[2] http://www.xilinx.com/products/devkits/HW-SD1800A-DSP-SB-UNI-G.htm

Anexo B

PRÁCTICA NO 2

ADICIÓN Y USO DE PERIFÉRICOS LOCALES AL MICROBLAZE

INTRODUCCIÓN

En los ES, los puertos de entrada y salida juegan un papel determinante dentro del sistema al convertirse en la interacción del usuario o el exterior con el sistema. Dentro de estos puertos, la plataforma SPARTAN 3A DSP cuenta con 8 LED´s, 4 Pulsadores y 8 Interruptores (Dip Switch), todos destinados a la realización de aplicaciones que requieran de estos.

Para comprender la importancia que tienen estos puertos, se propone la construcción de varios ES en los cuales se usan estos recursos disponibles.

OBJETIVOS

OBJETIVO GENERAL

- Usar los recursos internos con los que cuenta la plataforma en el diseño de aplicaciones sencillas.

OBJETIVOS ESPECIFICOS

- Seleccionar los IP Core (Periféricos) necesarios para la creación de tres proyectos a realizar, utilizando LED´s, interruptores y Pulsadores.
- Adicionar los periféricos locales al PLB (Processor Local Bus), de la plataforma Spartan 3A DSP.
- Escribir los códigos en lenguaje C para la implementación de los proyectos con los periféricos internos de la plataforma.

TRABAJO PREVIO

Para realizar esta práctica correctamente es necesario que se haya completado satisfactoriamente la práctica “Introducción al EDK y Microblaze”, es decir, haber comprendido y analizado todos los tópicos que se incluyeron en esta primera práctica.

Además es necesario revisar el documento que contiene la información referente a los puertos de I/O de propósito general “xps_gpio” [1], el cual es el controlador de los IP Core encargados del manejo de los periféricos a usar en esta práctica.

TIEMPO ESTIMADO

El tiempo estimado para la realización de esta guía de laboratorio es de 2 (dos) secciones de 2 (dos) horas, o equivalente a esta duración.

DIAGRAMA DE BLOQUES DE LA PRÁCTICA

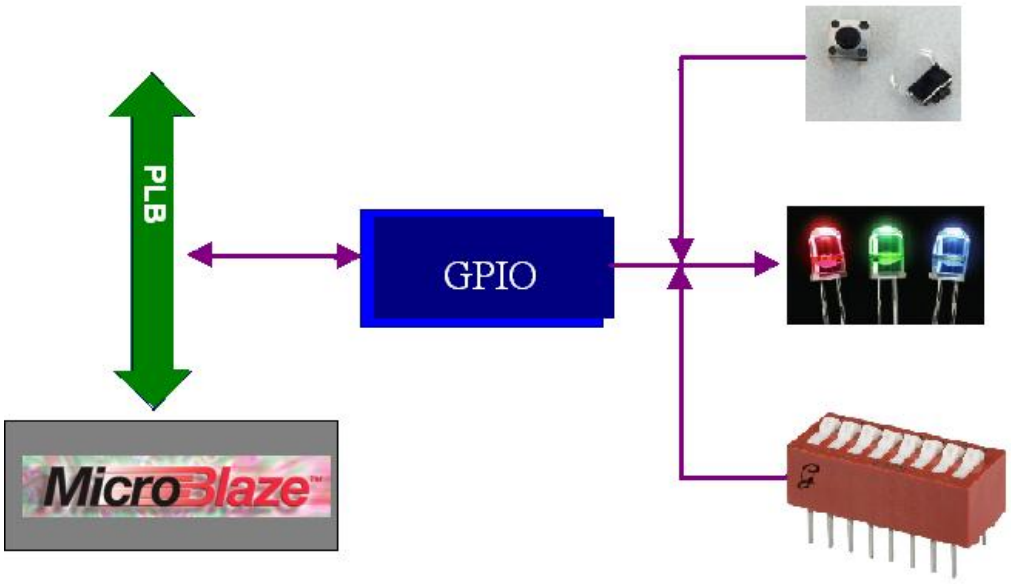

Figura 1. Diagrama de interconexiones del sistema

PROCEDIMIENTO

1. Crear un proyecto en EDK que permita el control del encendido de los LED´s disponibles mediante la utilización de un pulsador.

Asegúrese de incluir los periféricos necesarios para la implementación del sistema y realizar las configuraciones y conexiones antes de realizar el Bitstream.

Genere la aplicación software y realice el código en lenguaje C para cumplir con el objetivo del proyecto. Verifique el correcto funcionamiento del mismo mediante la implementación en la plataforma.

2. Incluir en un nuevo proyecto en EDK los interruptores y los LED's de la plataforma. Diseñar una aplicación que permita el encendido de los LED´s controladamente utilizando los interruptores, cada interruptor enciende un único LED.

   Observe el resultado del mismo en la plataforma para su comprobación.

3. Crear un proyecto en EDK que permita la visualización de un contador (ascendente/descendente) de 8 bit´s controlado por pulsadores, con opción de reset en cualquier momento.

   Agregue en la creación del proyecto los LED's y los pulsadores necesarios para el funcionamiento del sistema, configure el tamaño del periférico de los pulsadores a 3 bit's.

   Genere en lenguaje C el código para la ejecución del sistema deseado. Al implementar el proyecto se deberán encontrar las respuestas esperadas al oprimir el pulsador de incremento, decremento y reset a cero.

REFERENCIAS

[1] http://www.xilinx.com/support/documentation/ip_documentation/xps_gpio.pdf

Anexo C

PRÁCTICA NO 3

USO DE PERIFÉRICOS EXTERNOS

(Modulo Display de 7 Segmentos)

INTRODUCCIÓN

En los diseños de ES con la SPARTAN 3A DSP 1800A es necesario incorporar módulos externos, estos módulos pueden ser motores, sensores, pantallas LCD, display siete segmentos, etc. Para ello, esta práctica se pretende dar los pasos necesarios para la inclusión de un periférico externo que permita dimensionar la flexibilidad de esta plataforma.

Debido a la profundidad y complejidad que pueden llegar a tener los diseños con periféricos externos, esta práctica sólo es una guía de introducción y enseñanza que sirve como inicio a futuros proyectos agilizando el procedimiento para la creación de nuevos periféricos.

OBJETIVOS

OBJETIVO GENERAL

- Crear mediante el asistente de creación de periféricos que proporciona la herramienta EDK[23], un modulo IP Core (Driver[24]) para el control de un periférico externo.

OBJETIVOS ESPECÍFICOS

- Realizar e implementar un diseño en la plataforma que permita el control del modulo Digilent PmodSSD (Display 7 Segmentos Doble).

- Describir en VHDL el código que permita hacer la conversión BCD[25]-7 segmentos, e introducirlo al driver generado.

TRABAJO PREVIO

Lea el documento suministrado “Creación de un IP CORE” (Anexo H), consta de una guía para la creación y utilización de un nuevo IP CORE (Driver) para el manejo de periféricos externos, requisito esencial para la elaboración de la presente práctica.

TIEMPO ESTIMADO

El tiempo estimado para la realización de esta guía de laboratorio es de 3 (tres) secciones de 2 (dos) horas, o equivalente a esta duración.

[23] Embedded Development Kit
[24] Controlador o manejador de dispositivo
[25] Binary Coded Decimal

MÓDULO PERIFÉRICO DIGILENT PmodSSD[1]

El modulo Digilent PmodSSD cuenta con dos Display siete segmentos independientemente en configuración cátodo común, donde todos los cátodos de los diodos led se encuentran unidos y conectados a un sistema de conmutación que controla cada display 7 segmentos (Cat1 y Cat2), y los ánodos van a los pines del conector. En la figura 1 se muestra su diagrama de conexión.

Si se pretende encender el diodo led B1, el ánodo de B1 debe de estar en un ¨1¨ lógico mientras que el Cat1 debe de estar en ¨0¨ lógico (tierra) ya que todos los cátodos del primer siete segmentos se encuentra conectados a él, por tal motivo en el pin P4 debe entrar un ¨1¨ lógico que después de pasar por el negador llega a cat1 un ¨0¨ lógico, mientras en el cat2 entra un ¨1¨ lógico encendiendo el primer siete segmentos. Si se pretende encender el segundo display 7 segmentos, en P4 debe entrar un ¨0¨ para que en Cat2 llegue un ¨0¨logico.

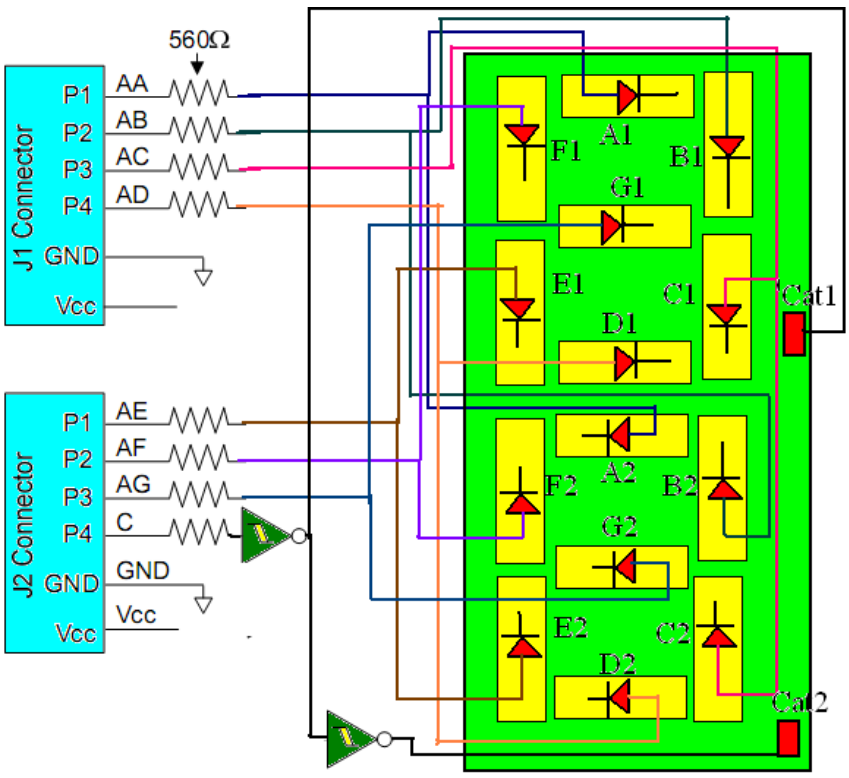

Figura 1. Diagrama de conexión del Display 7 segmentos doble (PmodSSD).

DIAGRAMA DE BLOQUES DE LA PRÁCTICA

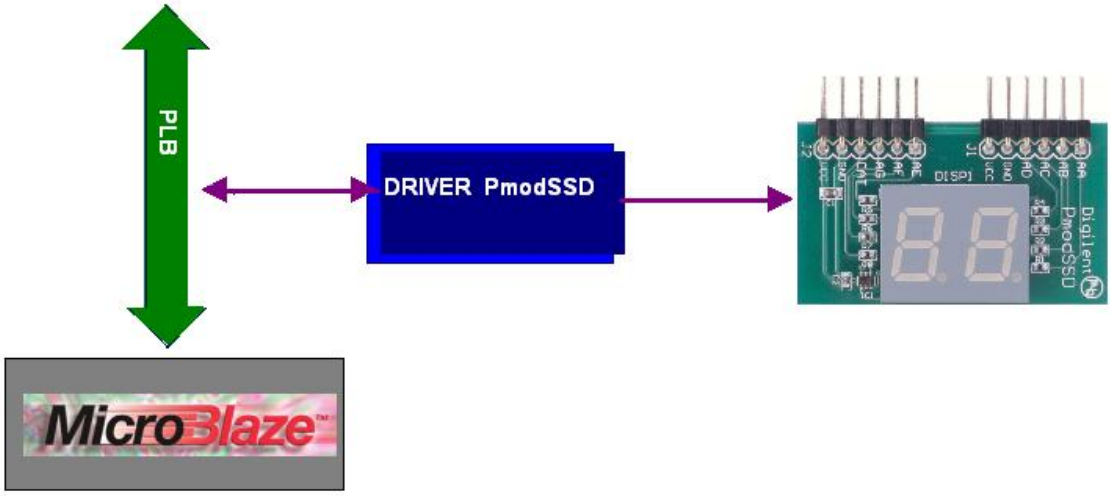

Figura 2. Diagrama de interconexión del sistema.

PROCEDIMIENTO

1. Crear un proyecto en EDK en el cual se utilice un GPIO para controlar los conectores de 6 pines como salidas, asegúrese de configurar el tamaño del periférico y de realizar las respectivas conexiones. Adhiera al archivo de restricciones (UCF) las líneas necesarias para este fin.

2. Realice una aplicación software donde se controle el encendido de cada uno de los display del modulo PmodSSD de Digilent, utilizando el periférico GPIO agregado en el punto anterior. Genere en lenguaje C las funciones necesarias para la conversión BCD – 7 segmentos. Además deberá observarse en el modulo, el encendido intercalado de cada uno de los display's.

3. Modifique el código anterior para lograr el efecto visual del encendido de los display's simultáneamente. Esto se logra disminuyendo el tiempo de

encendido de cada display y por ende aumentando la velocidad del intercalado.

4. Realice en lenguaje C un contador ascendente de 00 a 99, este debe acoplarse a las funciones anteriormente realizadas de conversión BCD – 7 segmentos y velocidad de transición de los display's para la correcta visualización del contador en el modulo PmodSSD display 7 segmentos doble.

5. Ejecute paso a paso el procedimiento contenido en el documento anexo "Creación de un IP CORE", en este se encuentran los pasos necesarios para la generación del control de un periférico externo.

6. Emulando el procedimiento anterior cree un nuevo periférico para el control del modulo 7 segmentos doble. Recuerde que debe escribir en VHDL el código para controlar la conversión BCD-7 segmentos, además de definir y mapear los puertos necesarios (7 segmentos).

7. Genere un nuevo proyecto en EDK en el cual se incluya el nuevo IP Core creado para la visualización en el modulo y un periférico GPIO de un bit para el control de la conmutación.

8. Realice en lenguaje C la programación de un contador ascendente de 00 a 99. Recuerde que los números generados deben ser enviados al IP Core de control del modulo por medio de los registros S/H de control. Además se debe generar la señal de conmutación en sincronía con el envió de los números para la visualización de los números.

REFERENCIAS

[1]http://www.digilentinc.com/Data/Products/PMOD-SSD/Pmod%20SSD_rm.pdf

Anexo D

PRÁCTICA NO 4

USO DE PERIFÉRICOS EXTERNOS

(Modulo ADC[26])

INTRODUCCIÓN

Como complemento a los trabajos anteriores con los periféricos internos y adicción de un periférico externo, en esta práctica se busca crear una integración para asemejar de mejor forma un ES y así permitir un mayor dominio en el uso de periféricos (internos y externos) hacia proyectos posteriores. Para esta práctica se utilizara el modulo ADC [1] PmodAD1 de Digilent como adquisición de datos, memoria DDR2_SDRAM[27] como almacenamiento temporal , puerto RS232 como transmisión de datos al PC ,Pulsadores de control, LED`s como visualización de los datos y Interruptores (Dip Switch).

[26] Analog-to-Digital Converter
[27] Double Data Rate Synchronous Dynamic Random Access Memory

OBJETIVOS

OBJETIVO GENERAL

- Crear un sistema que reciba una señal analógica externa, usando el ADC digitalice esta misma y la visualice en los LED's de la plataforma. Además el sistema debe contener la opción de almacenar un rango de datos para su posterior trasmisión a un computador, esto controlado por medio de pulsadores.

OBJETIVOS ESPECÍFICOS

- Describir en VHDL el código que controle el modulo ADC para la adquisición de datos.
- Diseñar en EDK un proyecto que permita la adquisición de los datos, almacenaje, control y trasmisión de los mismos.
- Escribir en lenguaje C el código correspondiente para la implementación del proyecto descrito.
- Integrar el modulo externo ADC PmodAD1 con módulos internos de la plataforma SPARTAN 3A DSP 1800A.

TRABAJO PREVIO

Realizar la práctica No 3 "Uso de Periféricos Externos" (Modulo Display de 7 Segmentos).

Comprender el funcionamiento del ADC (Convertidor Análogo a Digital) y memoria SDRAM (Memoria Dinámica Síncrona de Acceso Aleatorio).

TIEMPO ESTIMADO

El tiempo estimado para la realización de esta guía de laboratorio es de 4 (cuatro) secciones de 2 (dos) horas, o equivalente a esta duración.

Módulo ADC PmodAD1 [2][5]

Este es un modulo del fabricante Digilent (Figura 1) para plataforma de desarrollo, que se pueden conectar a uno de los puertos (I/O) de 6-pines de la Plataforma SPARTAN 3A DSP 1800A, permitiendo la conversión de señales analógicas a digitales.

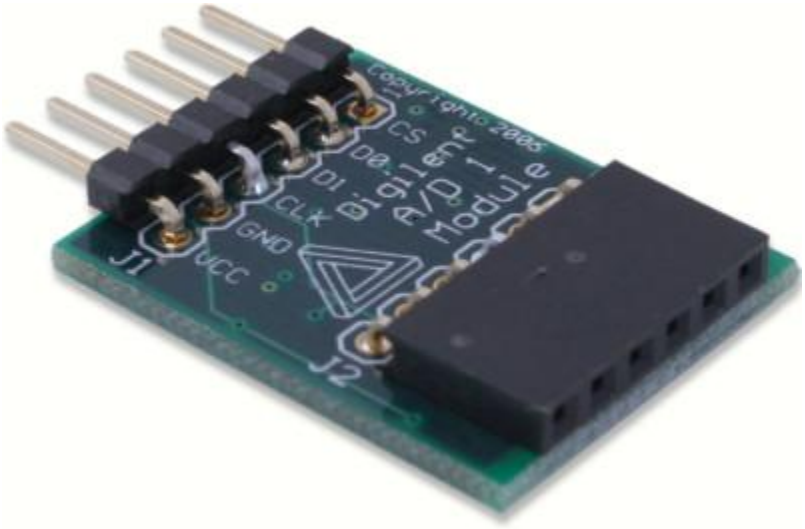

Figura 1. Modulo ADC PmodAD1 de Digilent

El PmodAD1 tiene dos conectores de 6 pines, uno que envía los datos digitales y recibe señales de control (macho) y el otro recibe los señales analógicas (hembra), dos ADC (ADCS7476MSPS[3] de National Semiconductor) de 12 bit, dos filtros antialiasing para evitar solapamientos, este es un filtro pasa bajas que

elimina las frecuencias que sobrepasan las frecuencia critica, atenuando lo mas posible el contenido armónico superior a la frecuencia de Nyquist[28][4].

Las señales analógicas entran por los pines P1 y P3 del conector J2 y van al filtro antialising F1 y F2 respectivamente, continuando hacia los convertidores (ADC1 y ADC2), cada ADC tiene un rango de voltaje de entrada de 0 a 3.3 voltios y salida digital de 12 bit y rango de 4095 (2^{12-1}), los dos ADC trabajan por separado operando señales simultáneamente.

El conector J1 está compuesto por las señales binarias en los pines P2 y P3, estas se trasmiten de forma serial; la señal de control CS (pin P1), señal del reloj (pin P4), el pin P5 es la tierra y el pin P6 es la alimentación del ADC, es recomendable alimentar el modulo directamente desde la plataforma utilizando el pin P6 del conector J1, ya que si se alimenta por los dos conectores J1 y J2 simultáneamente o si la fuente externa es mayor a 3.3 V puede ocasionar daños el ADC. El diagrama del circuito del modulo se muestra en la figura 2.

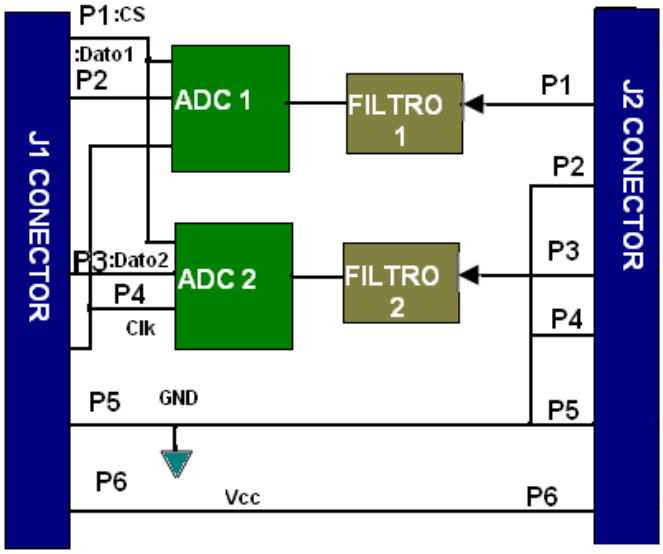

Figura 2. Diagrama del circuito ADC

[28] Teorema de muestreo formulado por Harry Nyquist en 1928

Como se observa en la figura 3, cada ADC emplea la señal de control CS para iniciar la conversión, esta se realiza por medio de un flanco descendente de la misma. A partir de este punto se generan salidas por medio de los pines de datos, son necesarios 16 ciclos de reloj para completar una muestra y es necesario además que la señal de control CS se mantenga en nivel bajo por lo menos 10 de los 16 ciclos. Para el ADCS7476 los datos de la muestra se generan a partir del cuarto ciclo de reloj luego de iniciar la conversión, por defecto el convertidor genera tres ceros (“0”) al comenzar la conversión mientras procesa la muestra presente. Para iniciar una nueva conversión es indispensable cambiar el nivel de CS a alto, preferiblemente luego de generar la muestra, y regresarlo a bajo e iniciar de nuevo el ciclo de conversión.

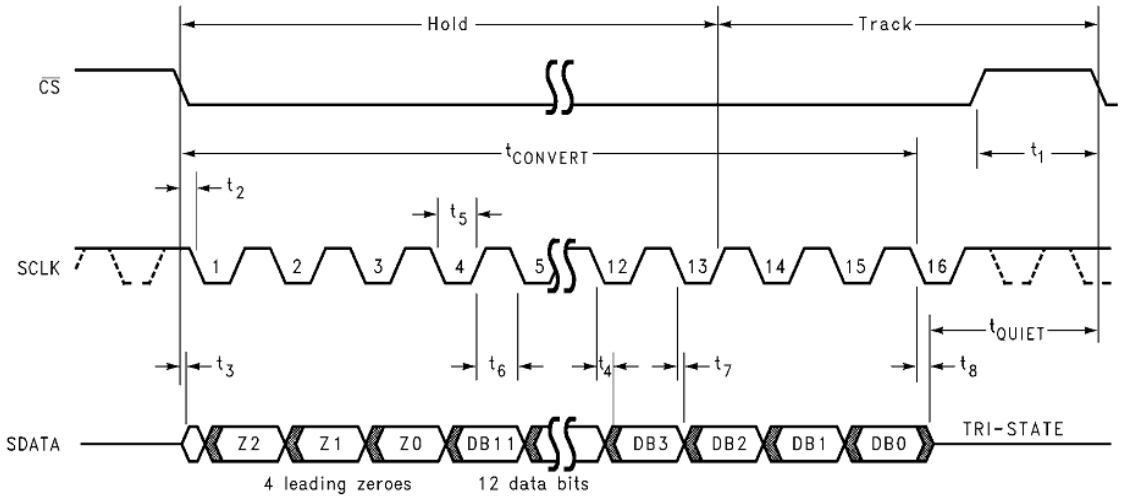

Figura 3. Diagrama de tiempos de funcionamiento del ADC.

DIAGRAMA DE BLOQUES DE LA PRÁCTICA

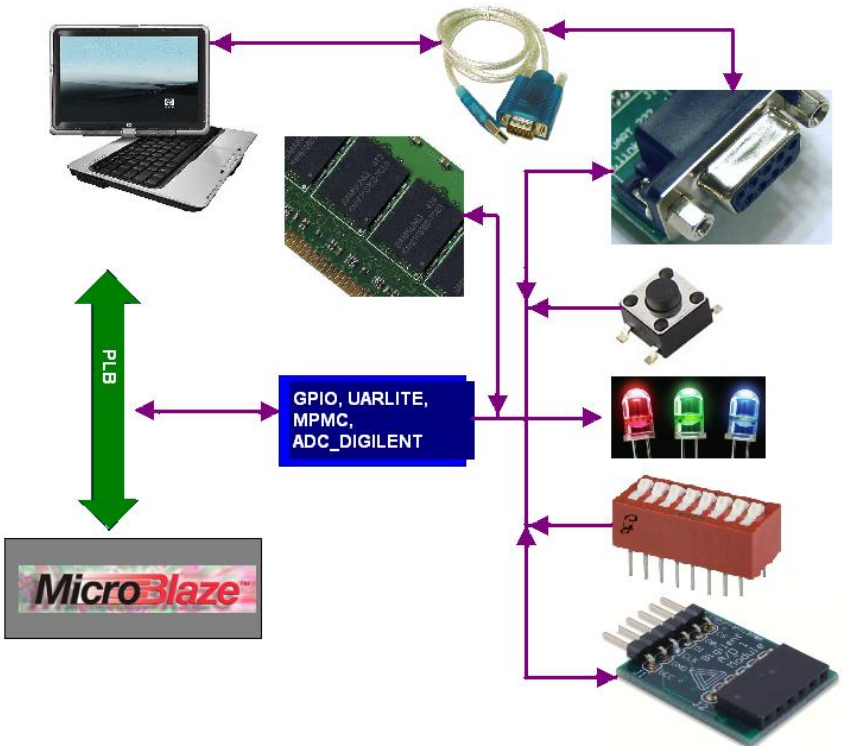

Figura 4. Diagrama general a realizar en esta práctica.

PROCEDIMIENTO

Para realizar el diseño propuesto previamente de creación de un sistema que contenga periféricos internos y externos (modulo ADC PmodAD1, memoria DDR2_SDRAM, LED's, Pulsadores y Puerto RS232), en necesario realizar los siguientes pasos:

- **Fase 1**

1. Realice el código en VHDL del controlador para el modulo ADC PmodAD1 de Digilent. Tenga presente que este debe generar la señal de control CS

para el ADC y la señal de reloj para el mismo. Pensando en la visualización de la señal en los LED's disponibles es imperativo una conversión serial-paralelo de los datos recibidos y una división sobre 16 para pasar de 12 a 8 bit's.

2. Mediante el asistente de creación de periféricos, genere el IP Core para el control del modulo PmodAD1, incorporando en él, la descripción en VHDL realizada en el punto anterior y los puertos necesarios para este fin.

3. Realice en EDK un proyecto en el cual se visualice en los LED's de la plataforma todo el rango de excursión del ADC. Esto es, para una entrada de 0 voltios los LED's tomaran un valor de 0 (00000000), y para una entrada máxima de 3.3 voltios (VCC) deberá observarse el valor de 255 (11111111).

- **Fase 2**

1. Como se piensa usar trasmisión serial (RS232)[6] y almacenamiento temporal (RAM)[7], se recomienda revisar los documentos proporcionados por Xilinx acerca de estos componentes (IP Core). Esto es necesario para conocer la forma de configurar estos periféricos y así adecuarlos a los requerimientos del proyecto. Recuerde que la configuración utilizada en el periférico RS232 debe coincidir con la configuración del Hyperterminal para realizar una conexión exitosa entre la plataforma y el computador.

2. Realice un proyecto en EDK que permita por medio de un pulsador almacenar en la memoria RAM un Byte (8 bit's) desde los interruptores (Switch), y por medio de otro pulsador muestre el dato almacenado en los LED's de la plataforma. Se recomienda el uso de punteros en la programación de la aplicación software para el manejo de la memoria,

teniendo la dirección base de la misma y por medio de punteros se puede acceder fácilmente a cualquier locación de almacenamiento.

3. Apuntando hacia el proyecto final, modifique el código anterior para almacenar una serie de datos finita y nuevamente al pulsar un botón trasmitir toda la serie de datos al computador usando el puerto RS232.

- **Fase 3**

1. Como fase final se propone la creación de un sistema en el cual se capture una señal analógica por medio del modulo PmodAD1, esta señal debe visualizarse en tiempo real en los LED's. Un botón que permita el almacenamiento de 100 muestras consecutivas y otro botón que inicie la trasmisión de los datos hacia un computador.

2. Identifique que periféricos son necesarios para realizar el diseño propuesto.

3. Realice en el EDK el proyecto descrito usando los periféricos seleccionados. Recuerde realizar las conexiones necesarias además de hacer las modificaciones al archivo de restricciones.

4. Escriba la aplicación software (Lenguaje C) para manejar los periféricos añadidos y cumplir las funciones establecidas.

5. Verificar el correcto funcionamiento del diseño mediante la implementación y visualización de resultados en la plataforma y en el cmputador.

REFERENCIAS

[1] http://www.dea.icai.upco.es/romano/sp/sisper_adc.pdf

[2]http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,499&Prod=PMOD-AD1

[3] http://www.national.com/ds/DC/ADCS7476.pdf

[4] http://www3.fi.mdp.edu.ar/electronica/tesis/Resumen-Carrica.pdf

[5] http://www.digilentinc.com/Data/Products/PMOD-AD1/Pmod%20AD1_rm.pdf

[6]http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf

[7] http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf

Anexo E

PRÁCTICA NO 5

USO DE PERIFÉRICOS DESDE EL PROCESADOR A TRAVÉS DE UN SISTEMA OPERATIVO (XILKERNEL)

NTRODUCCIÓN

La importancia que los ES han tomado durante la última década, ha liderado un desarrollo continuo en esta área, creando cada vez productos más complejos y capaces. Estos sistemas complejos impulsan a los programadores a usar un Sistema Operativo (SO), el cual ayuda con el manejo del hardware, permitiendo concentrarse concretamente en el desarrollo de la aplicación.

Otra ventaja que pueden ofrecer los Sistemas Operativos es el trabajo concurrente (Multithreading), operando varios procesos al mismo tiempo cuando sea necesario. Además pueden ofrecer sistemas para el uso eficiente de la energía.

OBJETIVOS

OBJETIVO GENERAL

- Crear un ES en donde se aprecie el trabajo multihilo (Multithreading) del microkernel proporcionado por Xilinx (Xilkernel), operando con el procesador MicroBlaze.

OBJETIVOS ESPECÍFICOS

- Diseñar un proyecto en EDK que permita visualizar en el Hyperterminal [1], por medio del puerto serial (RS232), la ejecución de varios hilos de impresión.
- Configurar el Xilkernel para operar varios hilos de ejecución (Thread's)
- Escribir el código en C para controlar la ejecución del Xilkernel y la impresión de los thread's mediante comunicación serial.

TRABAJO PREVIO

Leer y estudiar el documento complemento "Xilkernel" (Anexo I), con el fin de conocer las funciones, capacidades y la forma de configuración del microkernel que Xilinx ofrece para implementar en sus plataformas, Xilkernel.

TIEMPO ESTIMADO

El tiempo estimado para la realización de esta guía de laboratorio es de 3 (tres) secciones de 2 (dos) horas, o equivalente a esta duración.

SISTEMA OPERATIVO [2]

Se puede definir un Sistema Operativo como la interfaz que comunica el hardware con el usuario final. Este es el encargado de gestionar y coordinar los recursos disponibles de la maquina, como lo son como la CPU, memorias, puertos de red y comunicación, periféricos de entrada y de salida.

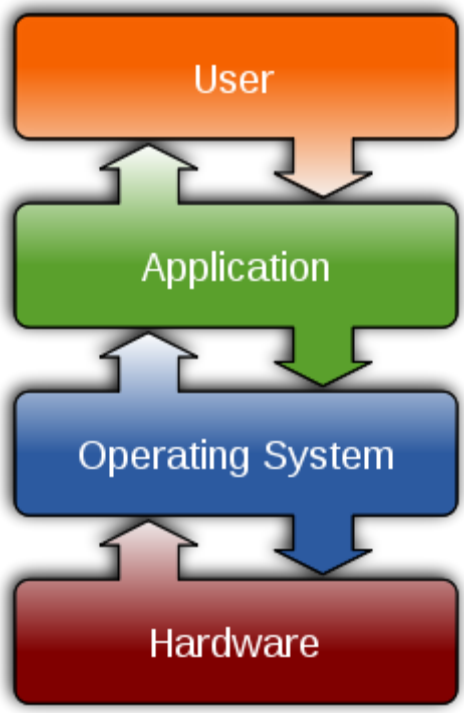

Figura 1. Diagrama de Bloques de un Sistema Computacional [2]

Dentro de los Sistemas Operativos más conocidos encontramos Windows, Linux, Solaris, Mac OS, etc.

KERNEL [3]

El Kernel o núcleo es la parte fundamental de un SO, es el código responsable de facilitar a las aplicaciones software los recursos de hardware necesarios para su

ejecución. Además posee la capacidad de seleccionar la utilización de estos recursos por parte de diferentes procesos que se ejecutan simultáneamente, esto consiste en definir qué proceso se ejecuta y durante que rango de tiempo.

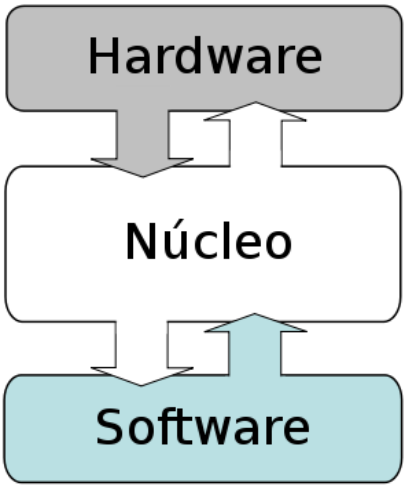

Figura 2. Posición del Kernel dentro de un Sistema [3]

Existen diferentes tipos de Kernel, estas dependen las funcionalidades y del tamaño del mismo [4].

- Monolítico; todas las funciones están integradas al sistema principal, por tal motivo se trata de un kernel robusto y de tamaño considerable.
- Modular; kernel que contiene las funciones esenciales para el funcionamiento del SO, dejando que controladores (Drivers) de periféricos u otros sistemas se adhieran al kernel posteriormente, según sea necesario. Esta es la línea hacia donde se está desarrollando esta tecnología.
- Microkernel; kernel consistido en una abstracción muy simple sobre el hardware, con un conjunto de funciones primordiales para implementar los servicios mínimos en un SO, como la gestión de hilos (Multithreading) y la comunicación entre procesos.

XILKERNEL [5]

Es un microkernel modular y robusto provisto por Xilinx, para trabajar desde el Xilinx Plattform Studio (XPS), actualmente está disponible la versión 4.00.a y está integrado con la descarga del software del EDK.

- Permite un alto nivel de modificación, dejando cambiar a gusto del usuario para optimizarlo en tamaño y funciones.
- Con un POSIX API, soporta núcleos esenciales para un microkernel como trabajo con hilos (Thread's), sincronización (Semáforos y Mutex, timers e interrupciones).
- Trabaja en procesadores MicroBlaze, PowerPC405 y PowerPC440.

PROCEDIMIENTO

1. Revisar el documento anexo "Xilkernel" (Anexo I), preste gran atención a los capítulos Requerimientos Hardware, Configuración Software y Programación, estos son necesarios para la realización de esta guía.
2. Crear un proyecto en EDK donde se incluya el periférico para trasmisión serial (RS232), así como los IP Core necesarios para la configuración del Xilkernel, Timer y controlador de interrupciones.
3. Realice las configuraciones y conexiones necesarias para los periféricos añadidos en el punto anterior. Proceda a realizar el Bitstream del proyecto.
4. Generar una aplicación software donde incluya el Xilkernel, recuerde agregar la librería –lxilkernel en las opciones del compilador, configurar el timer del sistema (selección, frecuencia e intervalo) y la selección de las configuraciones de entrada y salida estándar (RS232) y el controlador de interrupciones (xps_intc).

5. Active el modulo de thread’s estáticos y declare un thread llamado “Thread_1”, la prioridad por defecto puede usar el valor 1.

6. Copiar el código en lenguaje C entregado en el anexo y adherirlo a la aplicación software, este código imprime en el computador un mensaje desde el thread generado.

7. Configure el Hyperterminal para realizar la conexión entre la plataforma y el computador. Implemente el proyecto realizado y compruebe los resultados esperados.

8. Con el fin de visualizar el procesamiento multihilo del Xilkernel, incluya en la tabla de Thread’s dos nuevos thread’s (Thread_2 y Thread_3).

9. Adicione al código dos nuevas funciones de impresión con los nombres respectivos para los thread’s añadidos. Cree para cada uno de los thread’s una secuencia de impresión, puede ayudarse por medio de una sentencia FOR.

10. Implemente el proyecto mediante el cable de programación y observe los resultados en la ventana del Hyperterminal.

11. Puede observar de manera mas clara el funcionamiento del procesamiento multihilo, que puede compararse con un multiplexado por división de tiempo, si cambia el tiempo dedicado a cada proceso por parte del Xilkernel en el parámetro systmr_interval dentro de la configuración systmr_spec en la configuración de los modulos del Xilkernel. Revise la diferencia en los resultados entre usar un tiempo de intervalo de 1 (ms) y un tiempo de 40 (ms).

REFERENCIAS

[1] http://technet.microsoft.com/es-es/library/cc784492(WS.10).aspx

[2] http://en.wikipedia.org/wiki/Operating_system

[3] http://es.wikipedia.org/wiki/N%C3%BAcleo_(inform%C3%A1tica)

[4] http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node60.html

[5]http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

Anexo F

INSTALACIÓN DEL SOFTWARE ISE Y EDK

El continuo y rápido avance de la electrónica ha llevado a la realización de sistemas reconfigurables que permiten un rápido aprendizaje, teniendo presente que se requiere agilizar algunos procesos en la enseñanza de los Embedded Systems para poder dar camino a diseños más complejos; se pretende dar los pasos básicos como es la adquisición e instalación de las herramientas necesaria para el aprendizaje, como son el *ISE WebPACK* y el *EDK (Embedded Design Kit).*

Procedimiento de instalación de ISE

Si no se cuenta con el DVD de instalación del Software ISE 11, este se podrá adquirir en la página de Xilinx gratuitamente entrando a *www.xilinx.com*.

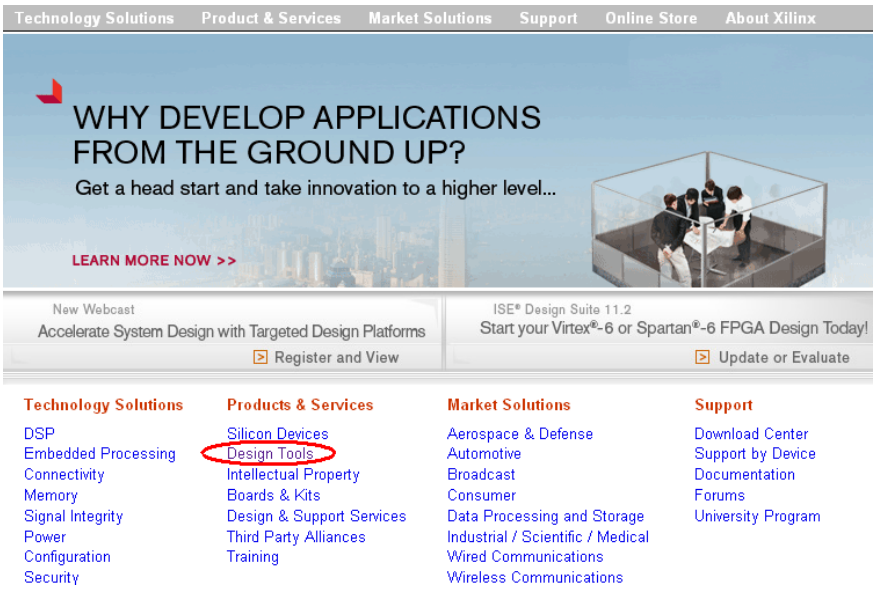

Figura 1. Pagina Web Xilinx

En la segunda columna *Products & Services*, se encuentra *Design Tools* (Figura 1), dando clic aquí aparecen las versiones disponibles de ISE.

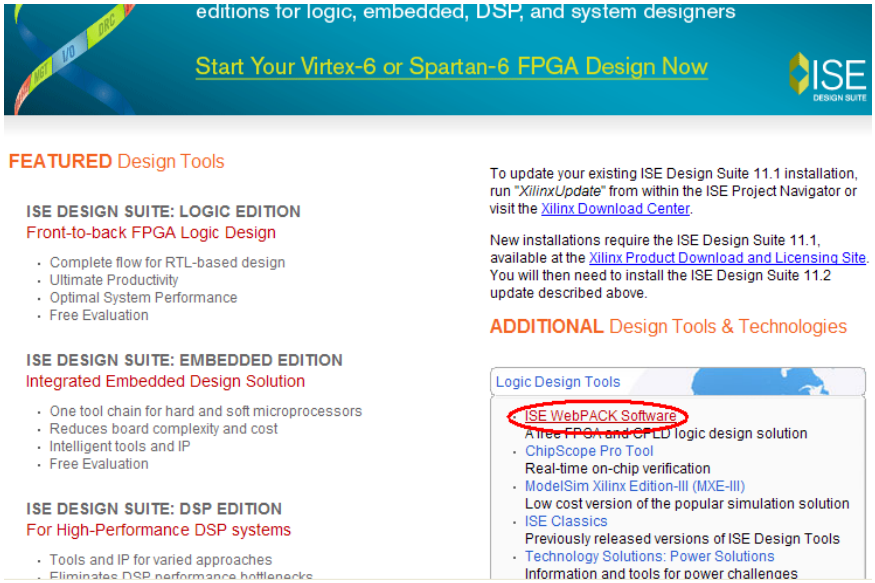

Figura 2. Listas de software disponibles por Xilinx.

Se selecciona la versión ISE WebPACK ya que es gratuita (Figura 2), a continuación aparece el link *Download ISE WebPACK software for Windows and Linux* (figura 3). Dar clic.

Figura 3. Link de descarga del ISE WebPACK

Para poder adquirir el software es necesario registrarse llenando un formulario que se encuentra en *Create Account* (Figura 4).

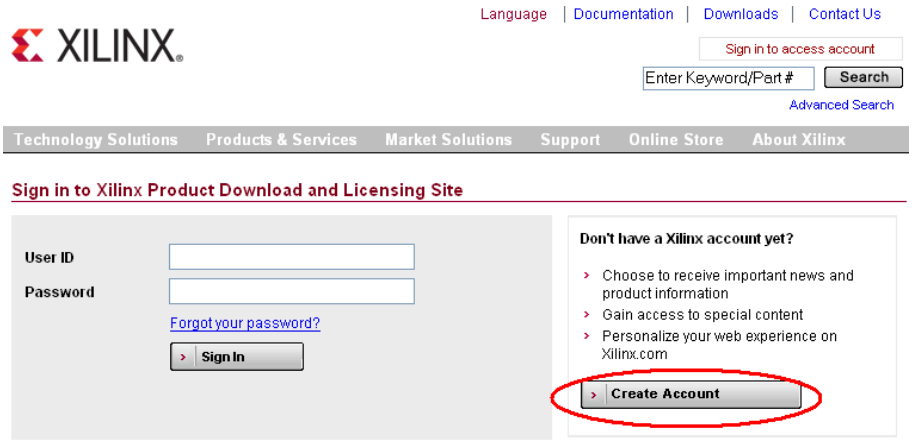

Figura 4. Link para acceder al formulario de registro.

Es obligatorio llenar todos los campos del formulario y es indispensable que la dirección de correo sea válida para poder recibir después la confirmación. (Figura 5).

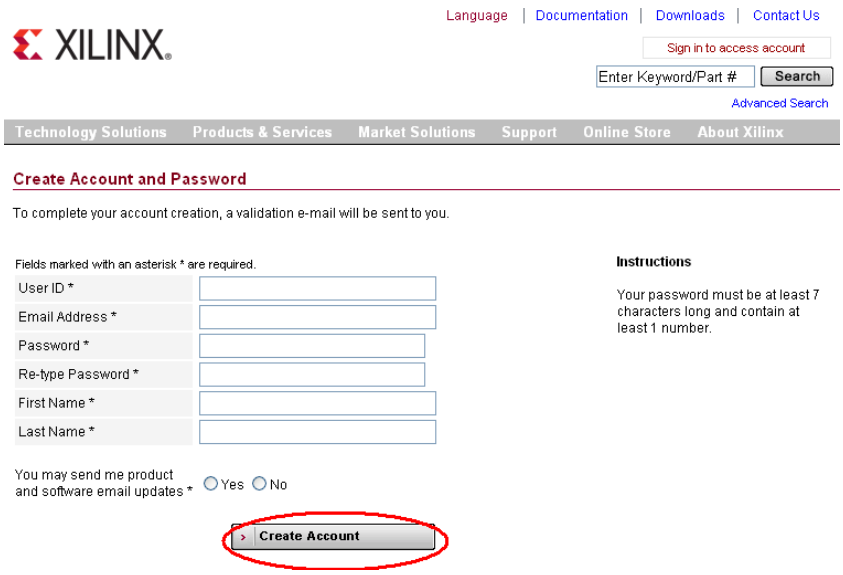

Figura 5. Formulario de registro.

Cuando llegue el correo al e-mail, hacer clic en el link de confirmación que empieza por *https://secure.xilinx.com/webreg*, luego se ingresa el nombre de usuario y la contraseña que se introdujo en el formulario anterior. Dando clic en *Sign In* aparecerá un nuevo formulario, el cual debe ser llenado en su totalidad (Figura 6).

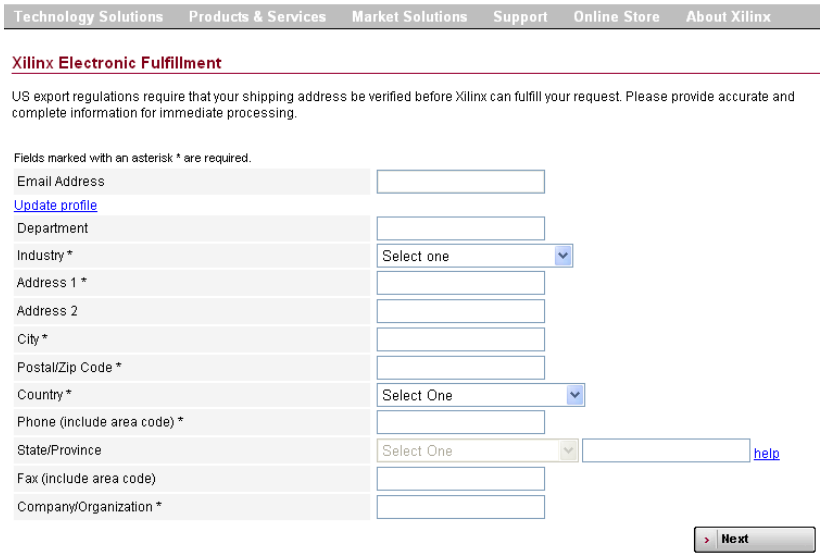

Figura 6. Formulario de ubicación y registro.

Se da clic en *Next*. El software ISE WebPACK se puede adquirir bajando el instalador de cliente que se encuentra en la parte izquierda de la figura 7 (*DOWLOAD web Install Client*); es un archivo comprimido de 88.8MB que debe ser instalado primero para poder adquirir el ISE WebPACK, este método permite pausar y continuar la descarga en otro momento y recuperar lo descargado hasta el momento si se presenta una interrupción en el internet o algún otro problema al obtener el software, si se requiere guardar el instalador para posterior instalación no es recomendable esta opción.

Un segundo método de adquirir el programa se visualiza en una lista de software (Download Individual Files, recomendable si se quiere guardar el instalador). Desde esta opción se descarga el instalador, en este caso se selecciona ISE WebPACK (Figura 7).

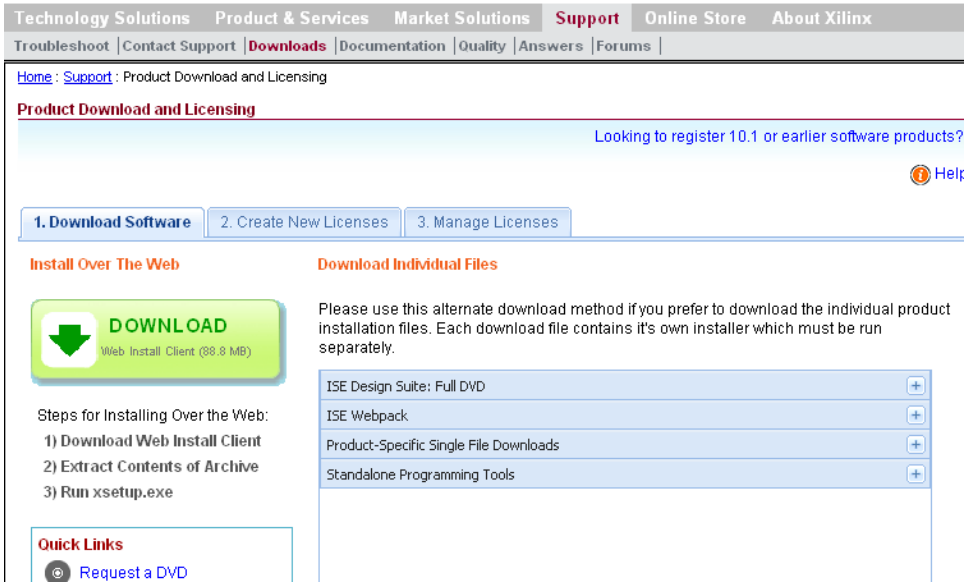

Figura 7. Lista de software y Download Web Install Client.

Al bajar el archivo comprimido se ejecuta *setup.exe*, seguido emerge una pantalla de bienvenida a la instalación y se pulsara *Next*. A continuación se aceptara los dos acuerdos de licencia correspondientes y se selecciona el directorio de destino, y los componentes del producto a instalar. Los dos pasos siguientes se dejan por defecto.

Seguidamente aparece una ventana donde se deselecciona las opción Launch XilinxUpdate (estas actualizaciones se podrán instalar posteriormente. Figura 8) y Enable WEbTalk (opción para enviar datos en la Web).

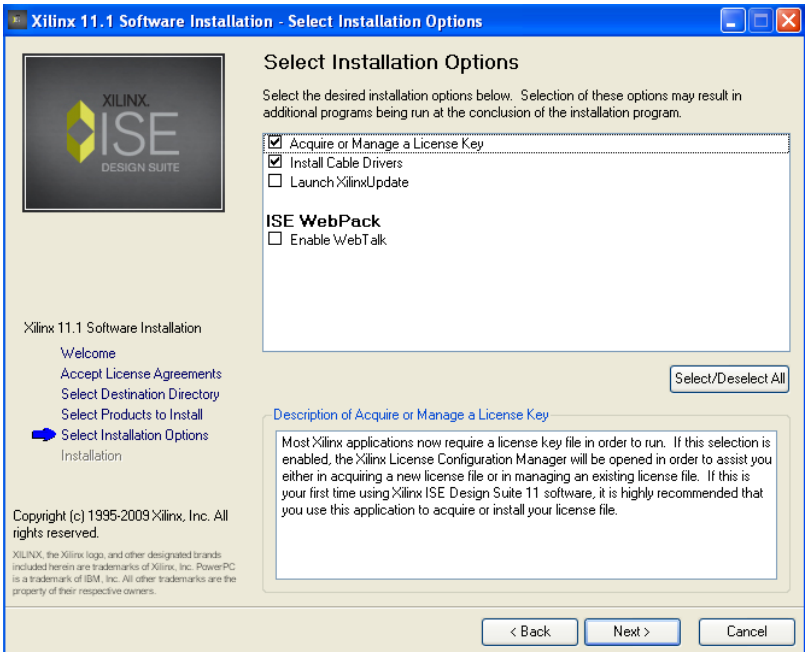

Figura 8. Opciones de instalación

Posteriormente emerge una ventana donde se resume los componentes de la instalación, clic en *Install* y se espera la conclusión de la misma.

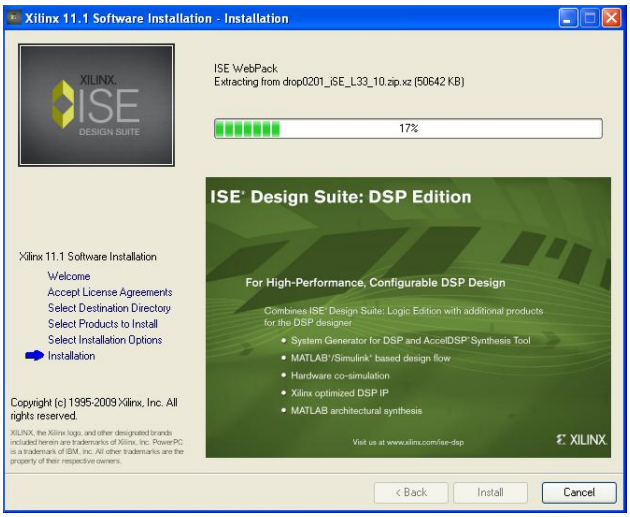

Figura 9. Instalación del ISE

Antes de terminar la instalación aparece una ventana afirmando que la instalación ha sido exitosa (Post instalación), seguidamente se pregunta si se acepta los términos del acuerdo de licencia de Microsoft Visual C++, clic en *Yes* para continuar (Figura 10).

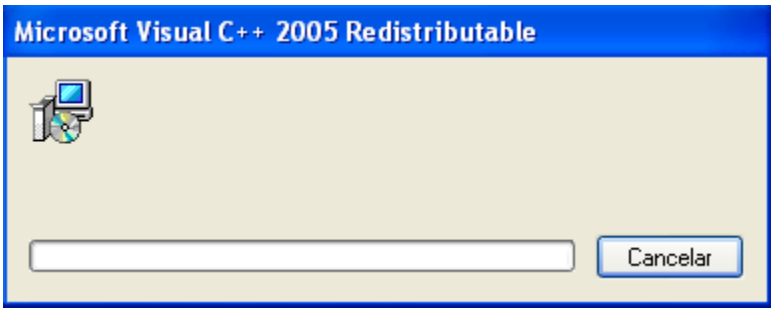

Figura 10. Instalación de Microsoft Visual C++

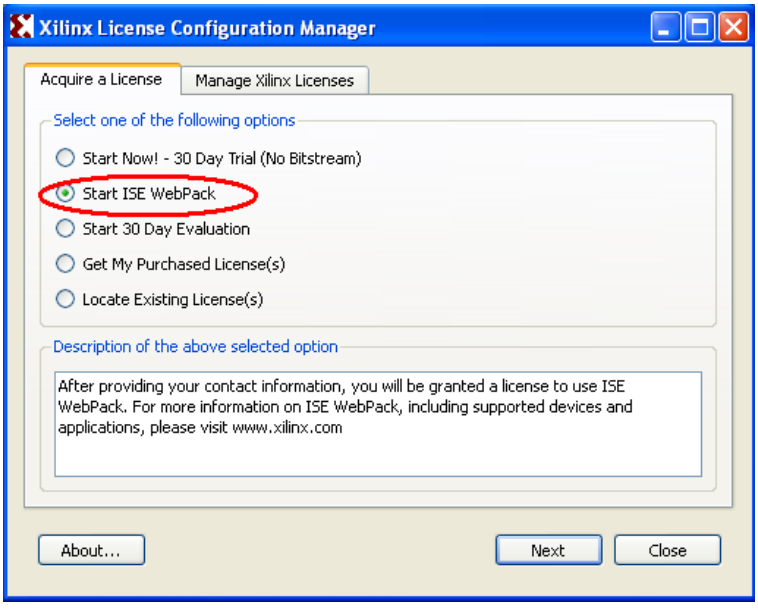

Figura 11. Admistrador de configurador de licencia

En la ventana de administrador de configuración de licencia (Figura 11) se selecciona *Start ISE WebPack,* ya que se da una licencia permanente (es indispensable en estos pasos estar conectado a internet), luego se pide un registro para poderse activar el producto instalado, seguidamente aparece una ventana dando las gracias por registrar el ISE Design Suite (Figura 12) las dos ventanas emergentes se da clic en *OK,* y listo ya se tiene instalado el ISE WebPack.

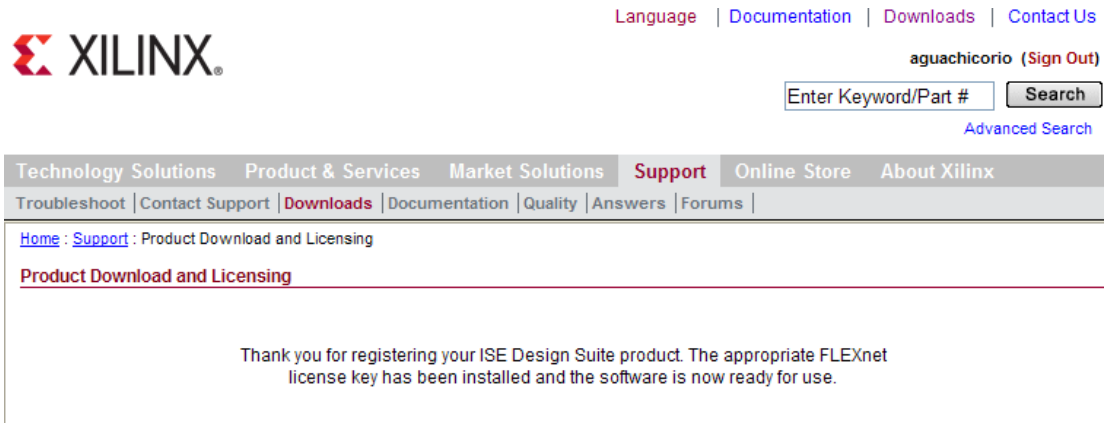

Figura 12. Finalización de registro de instalación

Procedimiento para la instalación del EDK

Para que funcione el EDK, es necesario haber instalado previamente el ISE. Al igual que el ISE, si no se tiene el DVD de instalación se puede bajar en la página de Xilinx siguiendo los pasos anteriores (ISE WebPACK) *www.xilinx.com, Design Tools, Embedded Design Tools, Platform Studio and Embedded Development kit, Free Evaluation, Download,* Registro (Clave y contraseña),*Product-Specific Single File Downloads, Sw Embedded Developers Kit (EDK) 11.1 Single File Download Image* tenga en cuenta que solo es gratis por 30 días , *Download*.

Cuando se complete la descarga, se procede con la instalación dando clic en el archivo *setup.exe*. A continuación aparece una ventana de bienvenida, seguidamente se aceptan los dos acuerdos de licencia, se designa el directorio de destino que por defecto coincide con el de la instalación del ISE.

Se seleccionar el producto a instalar.

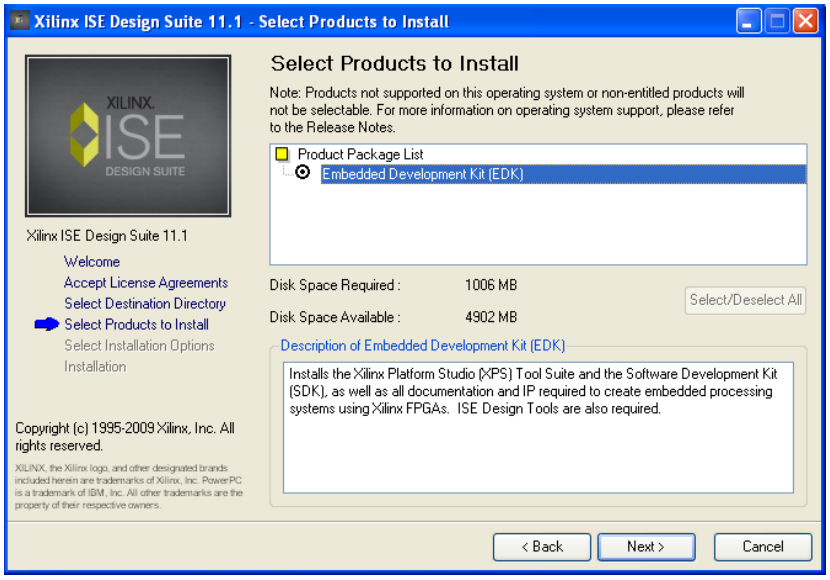

Figura 13. Selección de producto a instalar

El paso siguiente se deja por defecto, en la posterior ventana se deselecciona Launch XilinxUpdate, figura 14 (las actualizaciones se pueden bajar después).

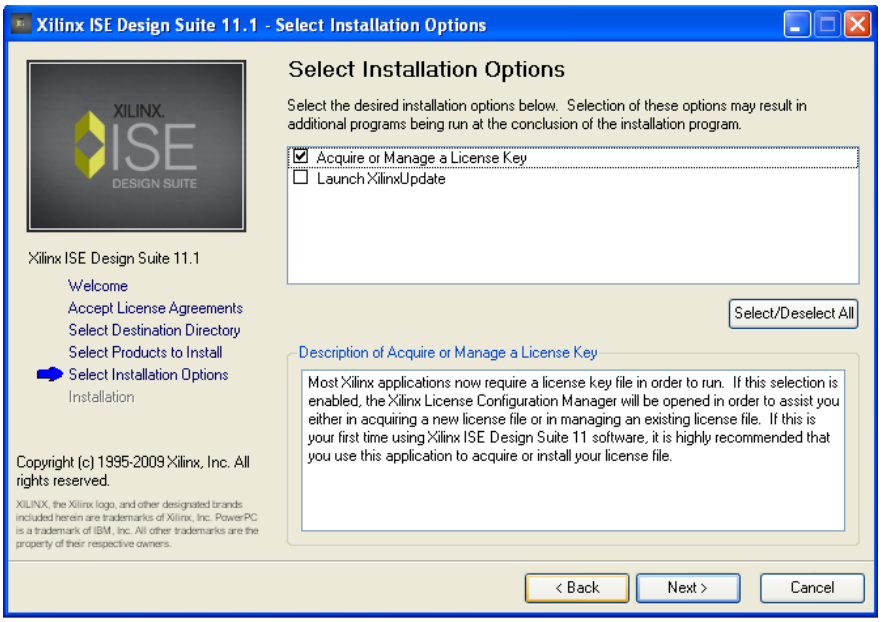

Figura 14. Opciones de instalación

Posteriormente aparece la ventana del resumen de la instalación, clic en *Install* para iniciar la instalación. Al terminar la instalación se presenta el acuerdo de licencia de instalación de Microsoft Visual C++, aceptando este acuerdo se instala el componente.

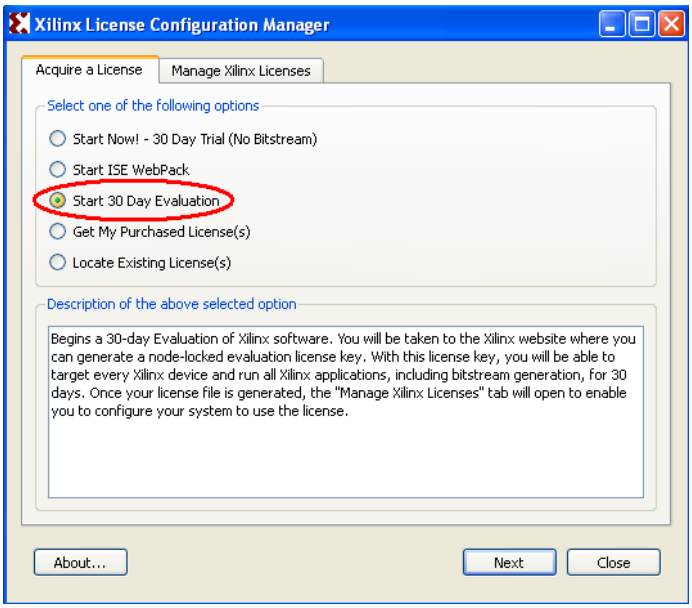

Figura 15. Administrador de configuración de licencia

Al emerger la ventana del administrador de licencia se selecciona la opción *Start 30 Day Evaluation* (figura 15).

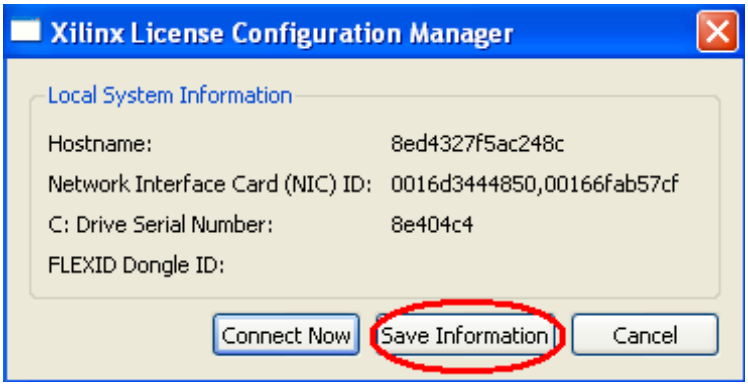

Figura 16. Informacion del sistema local, guardar informacion

Seguidamente se muestra una ventana con información del sistema local, si no se cuenta con internet por el momento guarde esta información (figura 16).

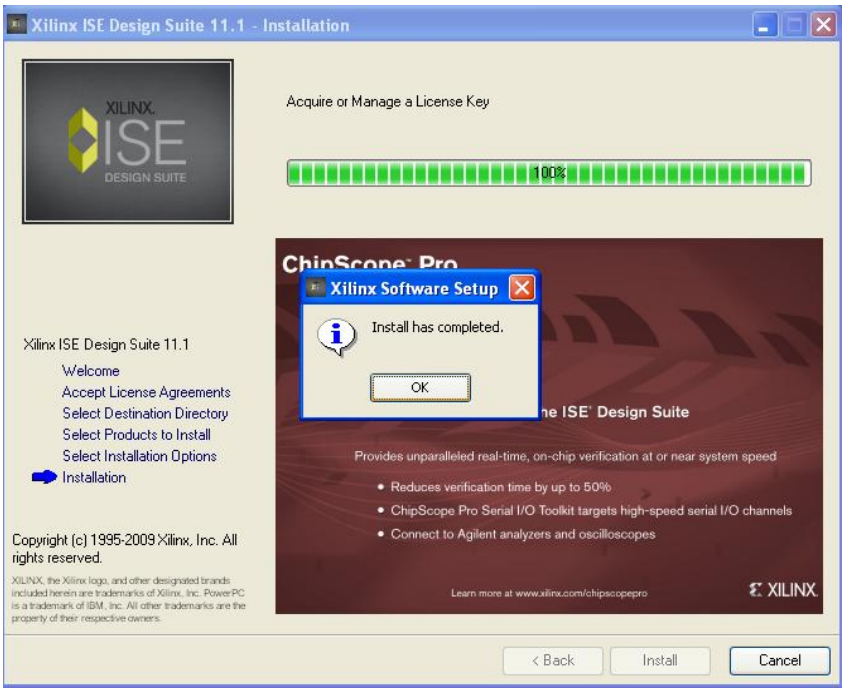

Figura 17. Instalación completada.

La instalación termina cuando aparezca la ventana emergente de instalación completada (figura 17).

Figura 18. Abriendo EDK, Xilinx platform Studio

Cuando se complete la instalación y se reinicie el equipo, al abrir el *EDK , Xilinx platform Studio* aparece un mensaje notando un error de licencia (figura 19).

Figura 19. Error de licencia

Dando clic en *OK* aparece nuevamente la ventana de la figura 15, se seleccionara la opción *Start 30 Day Evaluation*.

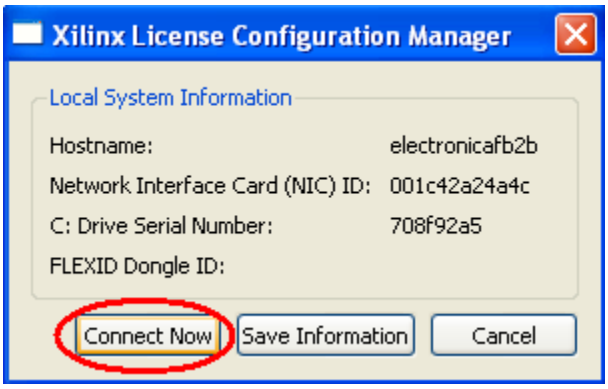

Figura 20. Información del sistema local, acceder a Xilinx

Consecutivamente emerge la ventana de información del sistema local, en este caso a diferencia de la figura 16 se debe estar conectado a una red de internet, dando clic en *Connect Now* (figura 20). Se debe estar registrado para poder continuar y poder adquirir la licencia de evaluación (no importa usar la misma cuenta para diferentes licencias).

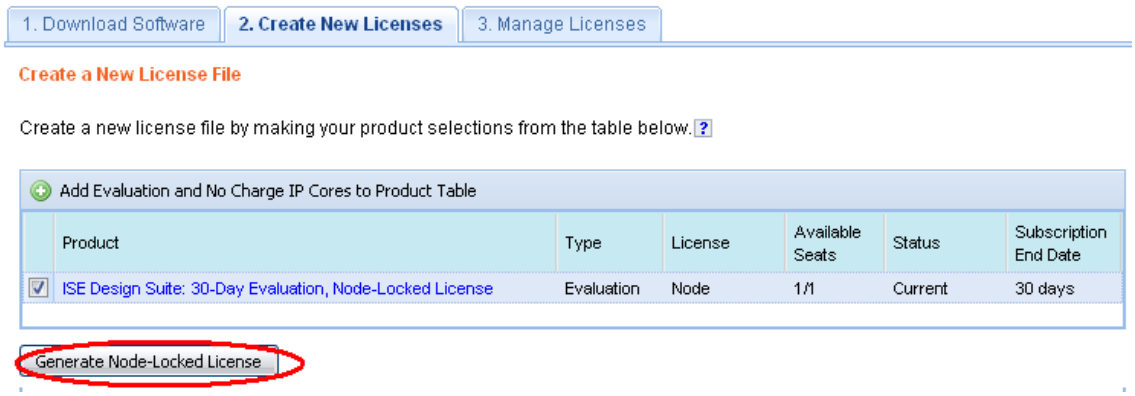

Figura 21. Generación de licencia

Luego de ingresar el usuario y contraseña y continuar la pagina de datos personales, se muestra la pestaña *Create a New License File* donde se da clic en

Generate node-Locked License (figura 21) para poder adquirir el archivo de licencia.

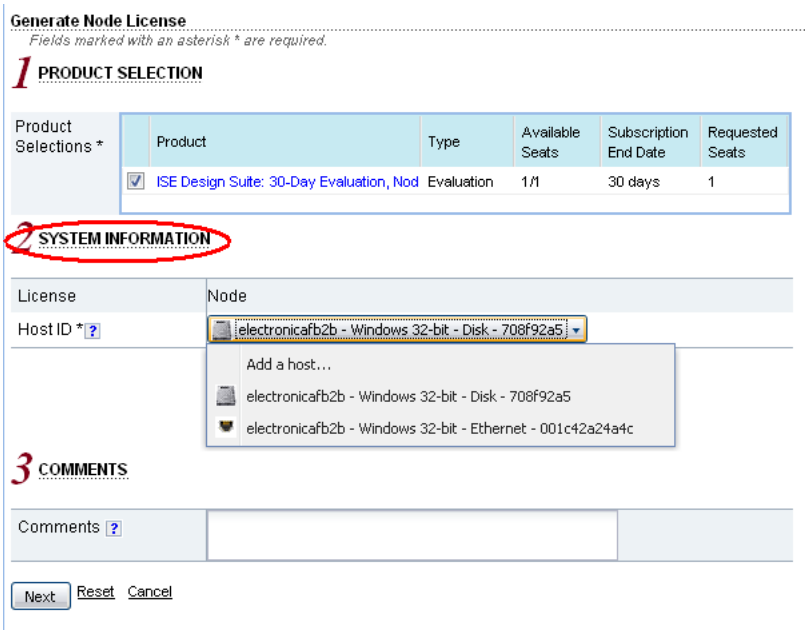

Figura 22. Información del sistema

En el segundo paso para generar la licencia, se debe seleccionar un Host ID para el archivo de licencia, este Host ID es un registro único del equipo donde se va a licenciar el producto y este depende del sistema operativo instalado (este puede ser la dirección física del equipo, el serial del disco duro o el identificador de la red, etc). El tercer paso es opcional si se quiere escribir algún comentario, se da clic en *Next* (figura 22). En el cuarto paso aparece la información del sistema se da clic en *Next* y aparece un mensaje indicando que el archivo de licencia fue enviado al correo registrado en la cuenta de Xilinx.

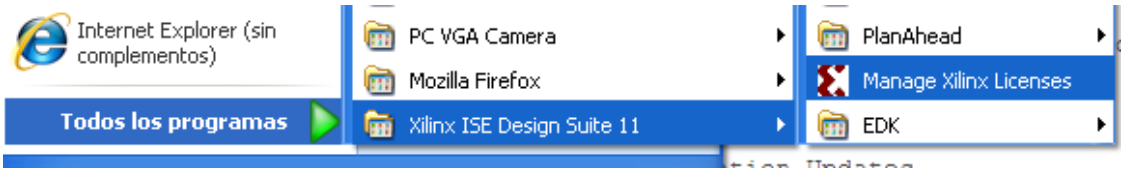

Figura 23. Administrador de licencias

Cuando llegue al correo el archivo de la licencia se guarda en una ubicación segura del equipo. Para poder cargar el archivo Xilinx.lic se va a *Todos los programas, Xilinx ISE Design Suite 11, Manage Xilinx Licenses* (figura 23).

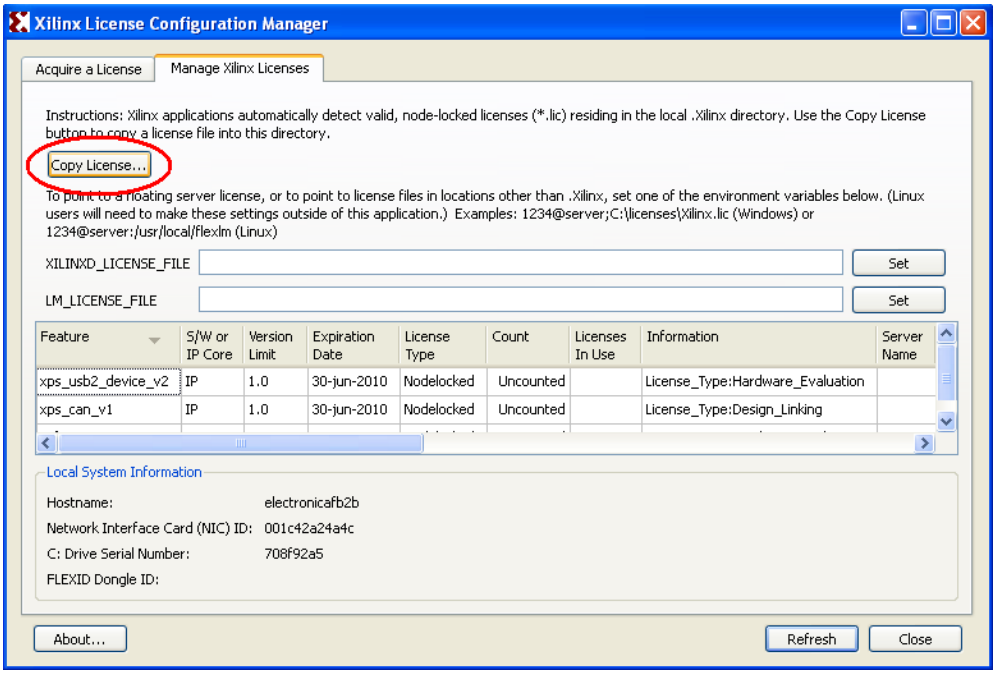

Figura 24. Cargar archivo de licencia

Seguidamente emerge la ventana de la figura 24, en *Copy License* se busca el lugar donde se guardo el archivo, al cargarlo se tiene activado el EDK 11.1 por 30 días.

Al utilizar Xilinx Plataforma Studio es posible tener algunos problemas, se recomienda actualizar el ISE WebPACK y el EDK 11.1. Estas se pueden encontrar en *www.xilinx.com,* en la columna Support aparece *Download Center,* y en *Product Updates* aparecen la lista de actualizaciones. Instale *Common Utilities Update* que es requerido para que produzca efecto las demás actualizaciones, después instale ISE™ Update y el EDK Update, estas actualizaciones son necesarias para poder trabajar con el Xilinx Plataforma Studio (Figura 25), las demás actualizaciones dependen de las necesidades del usuario con los demás componentes.

Figura 25. Actualizaciones requeridas

Referencias

- www.xilinx.com
- ISE Design Suite: Installation, Licensing and Release Notes. UG631 (v 11.1.0) April 27, 2009. Xilinx.

Anexo G

INTRODUCCIÓN AL SOFTWARE XILINX PLATFORM STUDIO (XPS)

En este apartado se van a desarrollar los principales pasos para el manejo del software Xilinx Platform Studio (XPS), dedicado especialmente para el desarrollo de aplicaciones de Embedded Systems. Para lograr este fin se propone a continuación la creación de un proyecto sencillo que explique de manera detallada el flujo de diseño en XPS.

El diseño consiste en el uso de uno de los periféricos internos que posee la plataforma Spartan-3A DSP 1800A de Xilinx, el objetivo entonces es encender y apagar el conjunto de LED's disponibles con un periodo de tiempo fijo.

Sistema de Bloques del Diseño

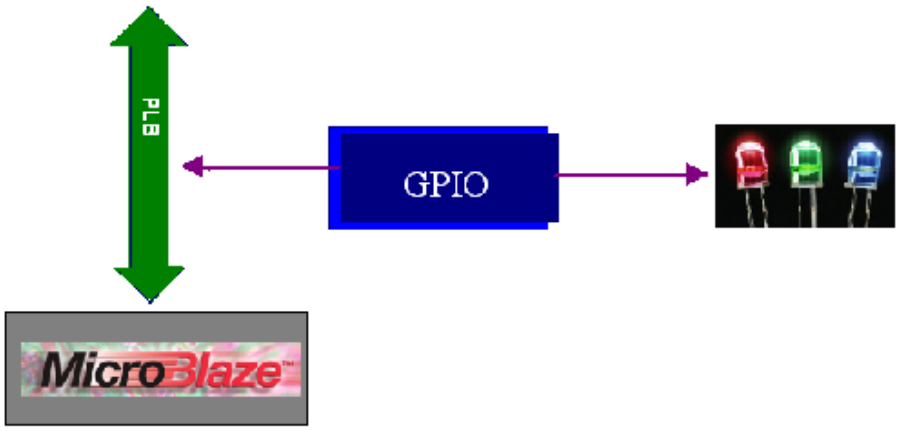

Figura 1. Diagrama de diseño

Creación del Diseño Básico

Para acceder al software XPS es necesario buscar el acceso directo creado durante la instalación del EDK o en el menú *Inicio*, *Todos los programas*, *Xilinx ISE Design Suite 11*, *EDK*, *Xilinx Platform Studio*; tal y como se muestra en la siguiente figura.

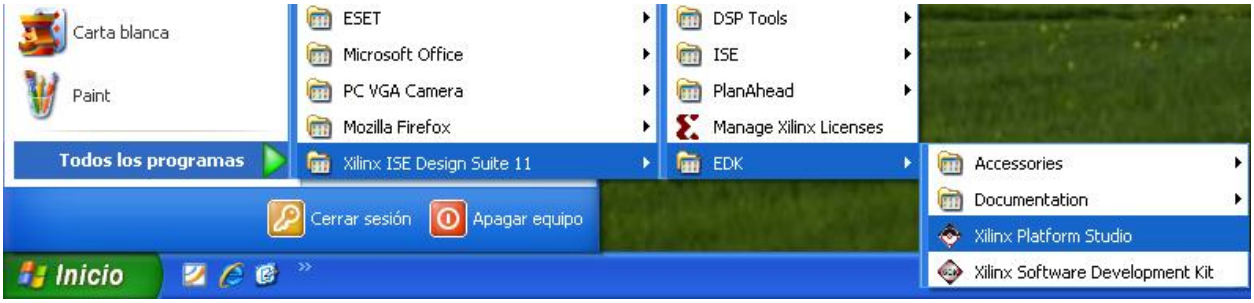

Figura 2. Acceso al Software XPS

Al iniciar XPS muestra una ventana preguntando si se quiere crear un proyecto nuevo o empezar con una configuración ya existente. Para crear un proyecto nuevo existen dos opciones la primera es *Base System Builder Wizard* el cual es un asistente para la creación de proyectos para determinados modelos de plataformas del fabricante y existe la segunda opción *Blank XPS Project* que no utiliza ningún asistente, por tal motivo se debe empezar desde cero especificando las características de la arquitectura dimensiones, paquete y velocidad de procesamiento.

Para abrir o modificar un proyecto ya existente se debe seleccionar *Open a Recent Project* y buscar el proyecto que se quiere en *Browse for More Projects*. En este caso se selecciona *Base System Builder Wizard*.

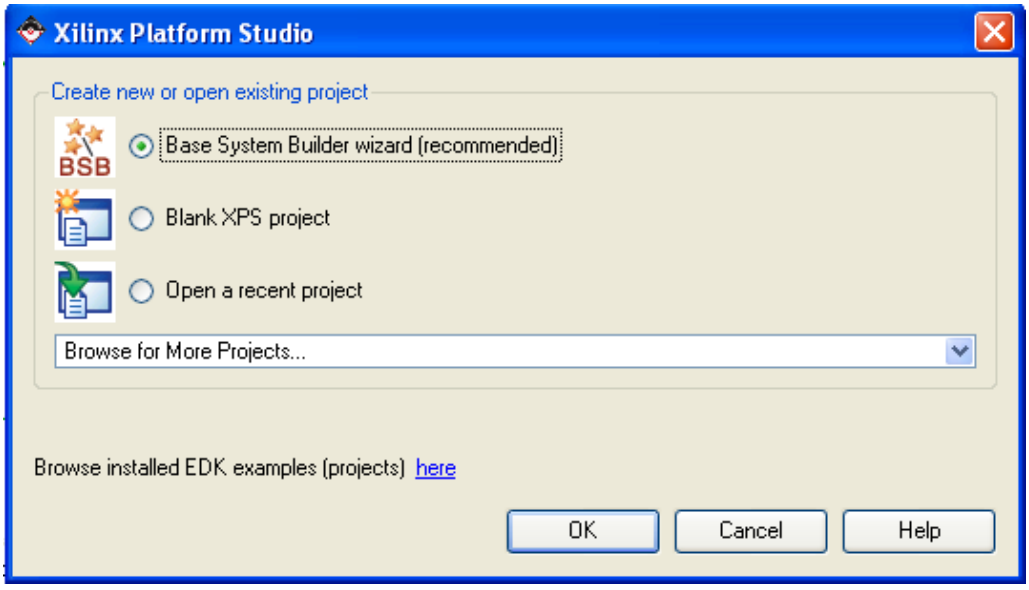

Figura 3. Ventana de Opción de Proyecto

Seguido aparece una nueva ventana que permite guardar el diseño en un directorio especificado dando clic en *Browse*. Es aconsejable que cualquier proyecto se guarde dentro de la carpeta de instalación de Xilinx (C:/Xilinx/), se puede crear una nueva carpeta denominada como “Proyectos” donde van a quedar guardados todos los proyectos que se diseñen y se recomienda que dentro de esta se cree una carpeta para cada proyecto con un nombre referente al mismo (este nombre no puede contener el carácter “ñ”). Al situarse en el directorio destino, XPS asigna por defecto el nombre “system”, pero este puede ser modificado a gusto del usuario guiándose por los lineamientos generales para el nombrado de archivos (que inicie con una letra, restringido el uso de algunos caracteres especiales, etc), sin dejar espacios y este debe terminar con la extensión .xmp.

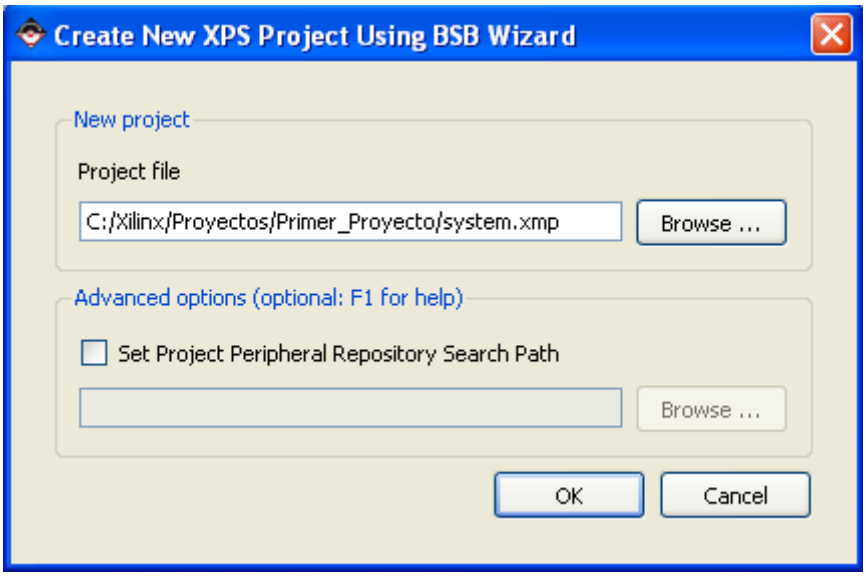

Figura 4. Selección del Directorio del Proyecto

Ya con el directorio y nombre especificado, clic en *OK* para continuar. A continuación una ventana pregunta si se quiere crear un nuevo diseño desde cero

(*I Would like to create a new design*), o quisiera cargar uno existente (*I would like to load an existing*) el cual utiliza algún archivo .bsb (Base System Builder) creado previamente. Este archivo consta de las configuraciones de plataforma, periféricos, etc.

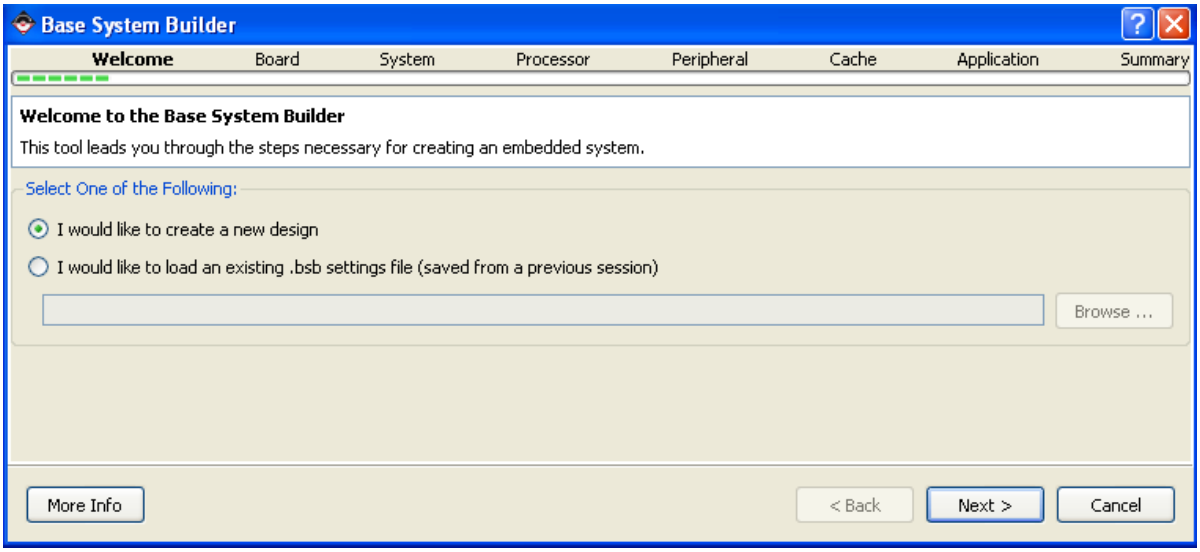

Figura 5. Bienvenida del Base System Builder

Se selecciona la primera opción y se da click en *Next*. Seguido a esto se pregunta si se quiere crear un sistema para una plataforma ya soportada por la herramienta (*I would like to a system for the following development board*), o si se quiere usar una configuración no genérica de alguna plataforma (*I would like to create a system for a custom board*); en este caso se usa la primera opción, escogiendo la plataforma Spartan-3A DSP 1800A Starter Board de Xilinx, como se muestra en la figura 6.

En la parte inferior de esta ventana en Related Information, aparece una descripción básica de los componentes primordiales de memoria, periféricos, relojes, etc; para la plataforma seleccionada.

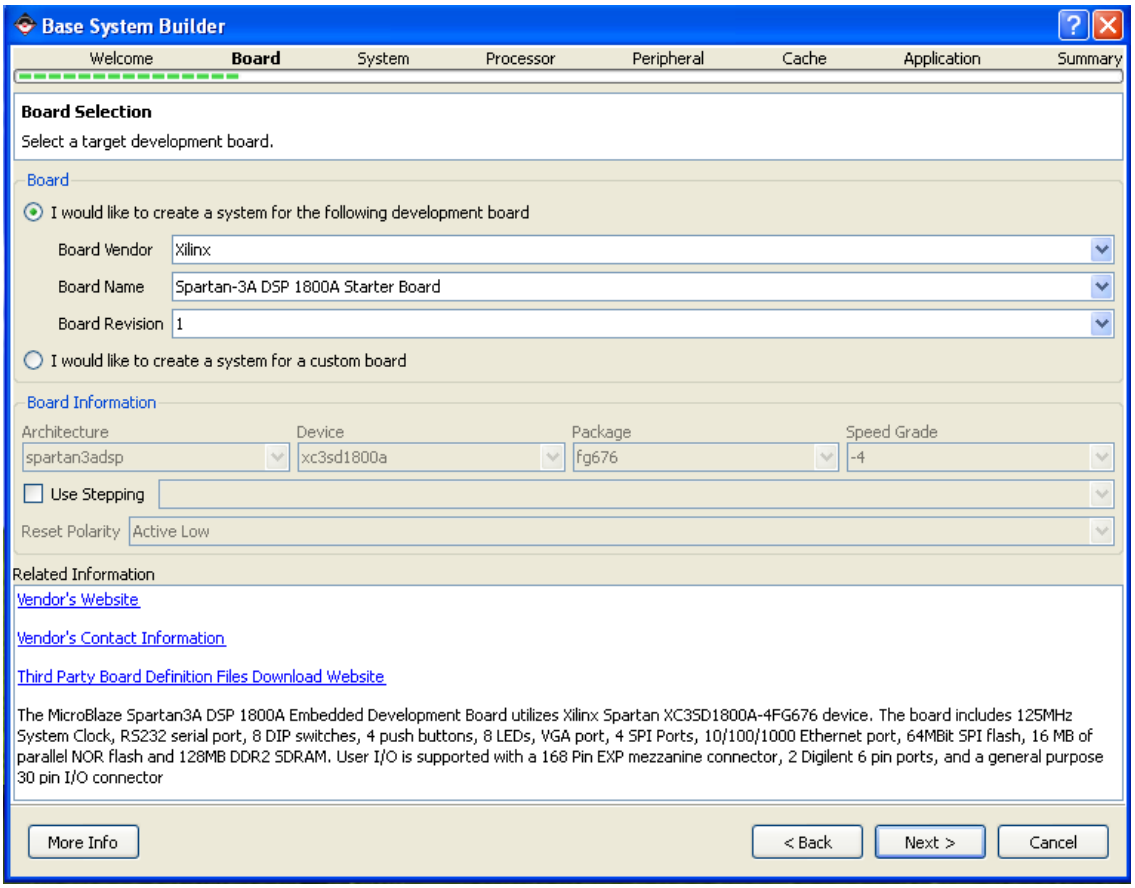

Figura 6. Selección de Plataforma

Al seleccionar *Next* se muestra una nueva ventana en la cual se pide que seleccione la configuración de procesador a utilizar. Un procesador que maneje todos los periféricos del sistema ò, un sistema dual de procesadores en el cual cada procesador controle sus propios periféricos y exista la posibilidad de compartir algún periférico entre estos.

Se escoge la opción de un procesador debido a la simplicidad del diseño, además de que el uso del sistema dual es recomendado para diseños de alto nivel de complejidad y computo.

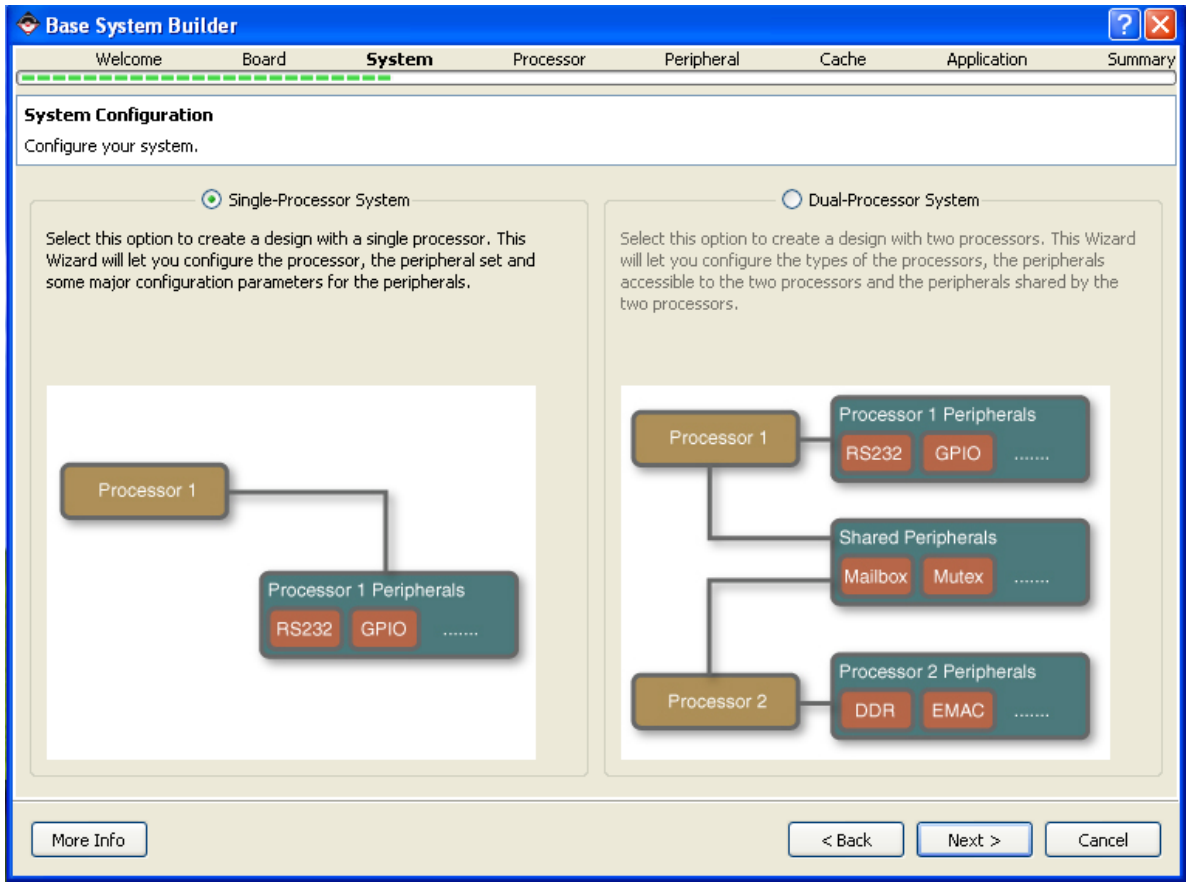

Figura 7. Configuración del Sistema de Procesado

Siguiendo con el proceso emerge una ventana para la configuración del procesador seleccionado. Para este caso se determina la configuración del MicroBlaze, ajustando la frecuencia del bus de datos del procesador (62.50 MHz entre los diferentes valores posibles), memoria local utilizada por el procesador, y habilitación de la unidad de punto flotante.

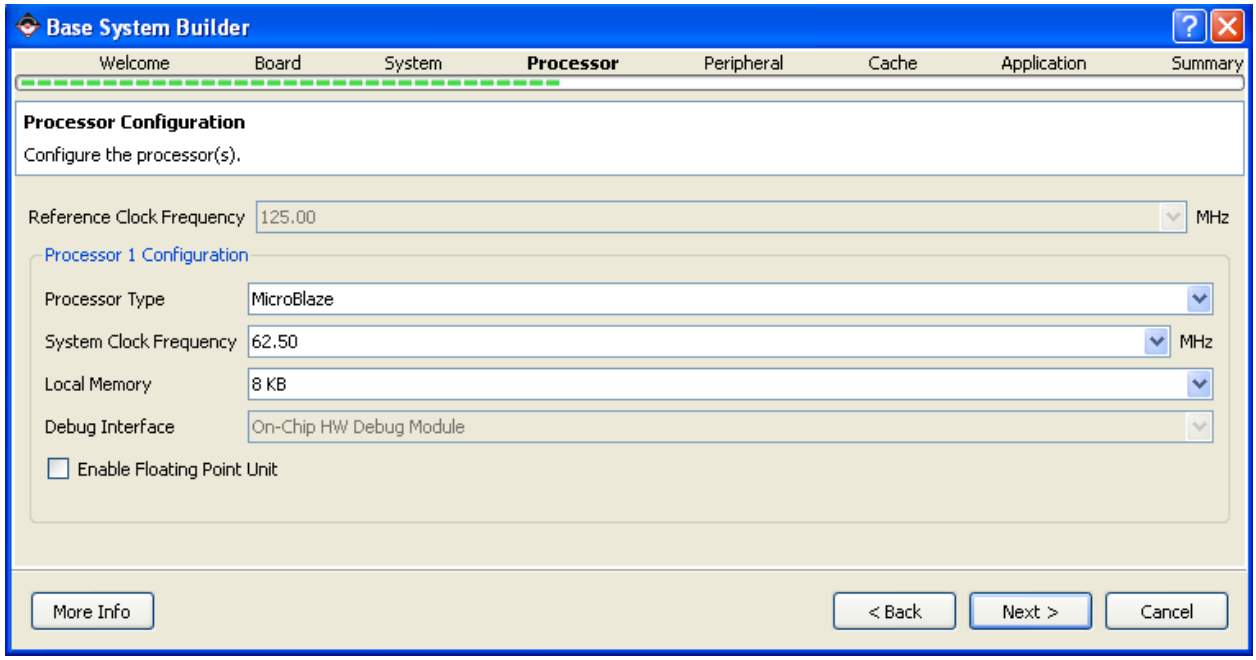

Figura 8. Configuración del Procesador

Los valores se deben ajustar a los mostrados en la figura anterior, no olvide que el valor de memoria local depende de la longitud y complejidad de la aplicación software asociado al proyecto; para este caso se usó un valor de 8 KB al tratarse de una aplicación simple, para aplicaciones que consuman mayor recurso debe usarse el valor máximo (64KB) de memoria para evitar problemas posteriores.

A continuación aparece la venta de selección de periféricos para el sistema divididos en dos grupos, los periféricos de entrada/salida (IO) y periféricos internos. Estos se pueden mover usando las pestañas *Add* y *Remove*. Para este ejemplo basta con usar los LED`s, los periféricos dlmb_cntlr y ilmb_cntlr son inmobles de cualquier diseño al ser los encargados del control de la memoria del procesador.

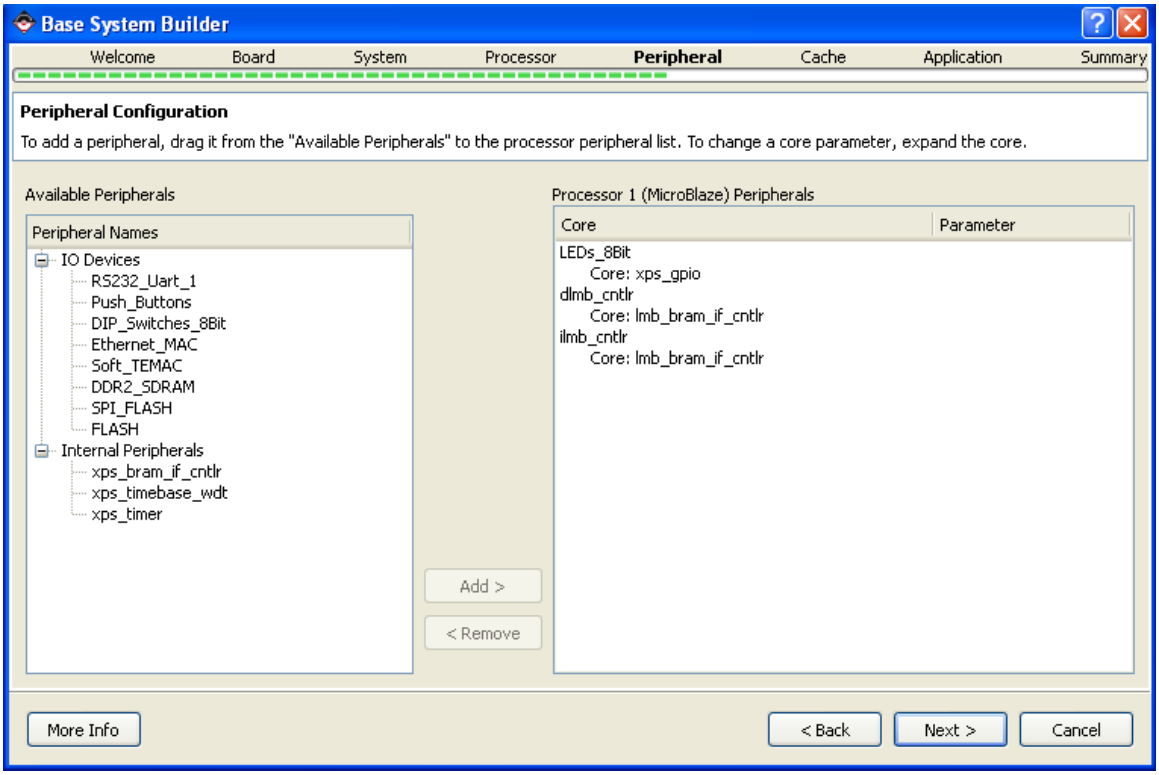

Figura 9. Periféricos del Sistema

Seguido se presenta la ventana de configuración de cache de memoria, aplicable únicamente en caso de agregar la memoria RAM disponible en la plataforma, clic en *Next* para continuar.

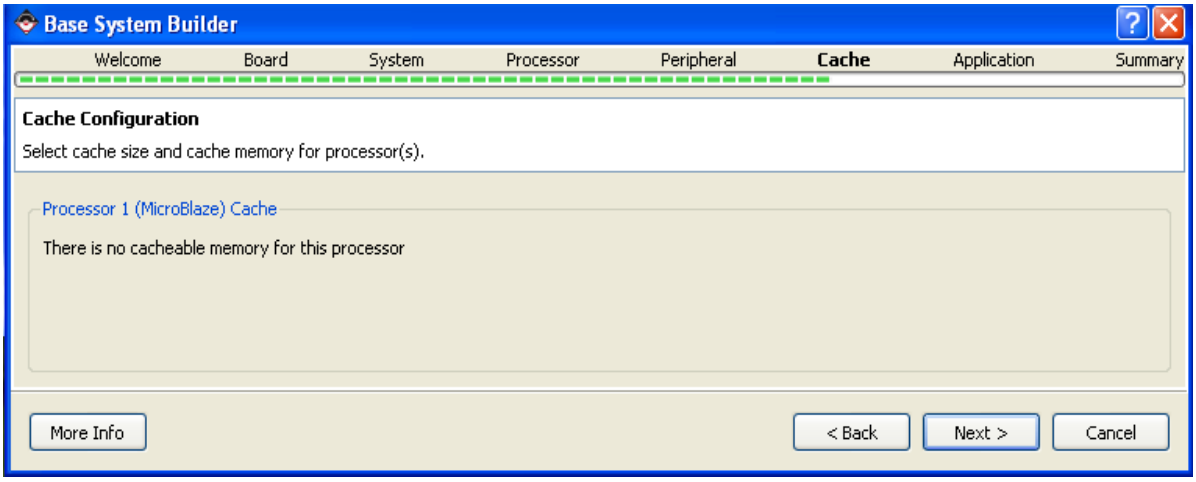

Figura 10. Configuración de Cache

En la ventana de configuración de aplicaciones se presentan las aplicaciones software aplicadas por defecto al sistema (test de memoria y test de periféricos).

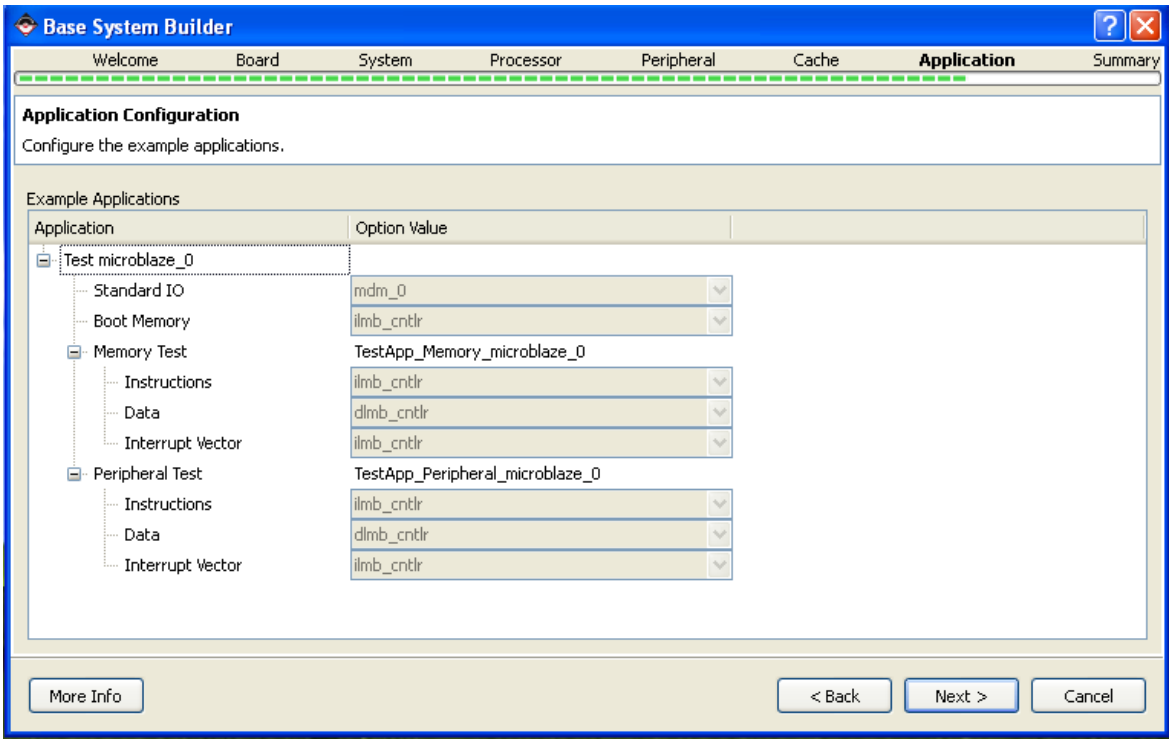

Figura 10. Configuración de Aplicaciones de Ejemplo

A continuación aparece la ventana de resumen en la cual se encuentra en detalle el procesador incluido, los periféricos anexados a este, así como la ubicación de los archivos creados tanto para el diseño como para las aplicaciones; además en la parte inferior se presenta la opción de guardado del archivo de configuración (.bsb) del proyecto para que pueda ser usado en una próxima ocasión. Clic en *Finish* para concluir el asistente y empezar a trabajar con la interfaz del XPS.

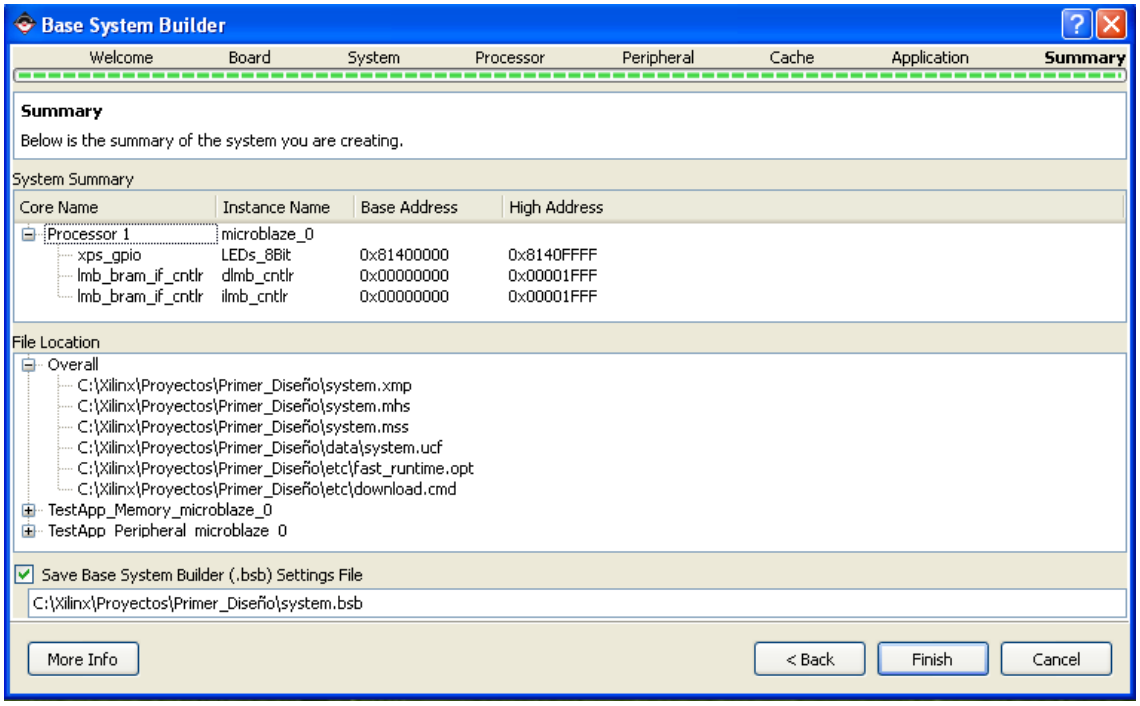

Figura 11. Resumen del Sistema Generado

Seguidamente surge una ventana preguntando qué acción se desea realizar, entre las opciones mostradas, seleccionar la última para comenzar a trabajar con proyectos de ES con la interfaz XPS (Xilinx Platform Studio).

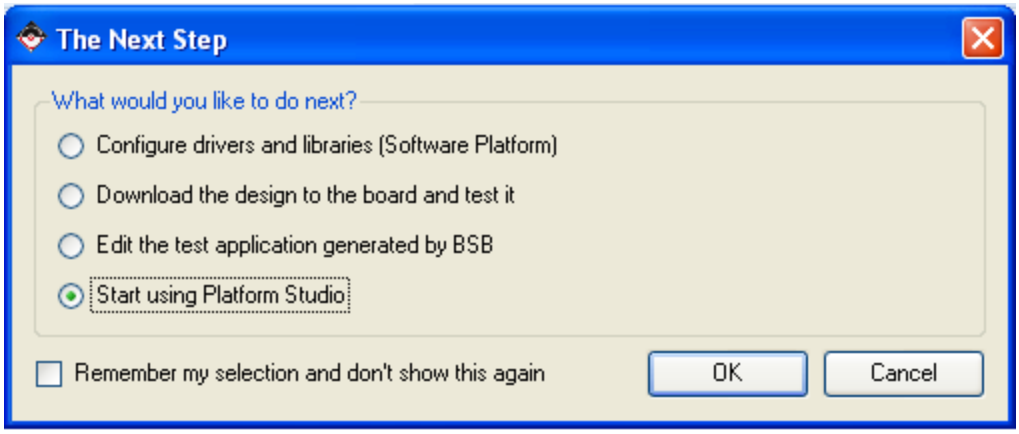

Figura 12. Comienzo del Trabajo con XPS

Interfaz XPS

Es aquí donde comienza la acción, es en esta interfaz donde se pueden añadir periféricos, controlar conexiones de los diferentes buses, añadir la aplicación software en lenguaje C, realizar la programación de la Plataforma, etc.

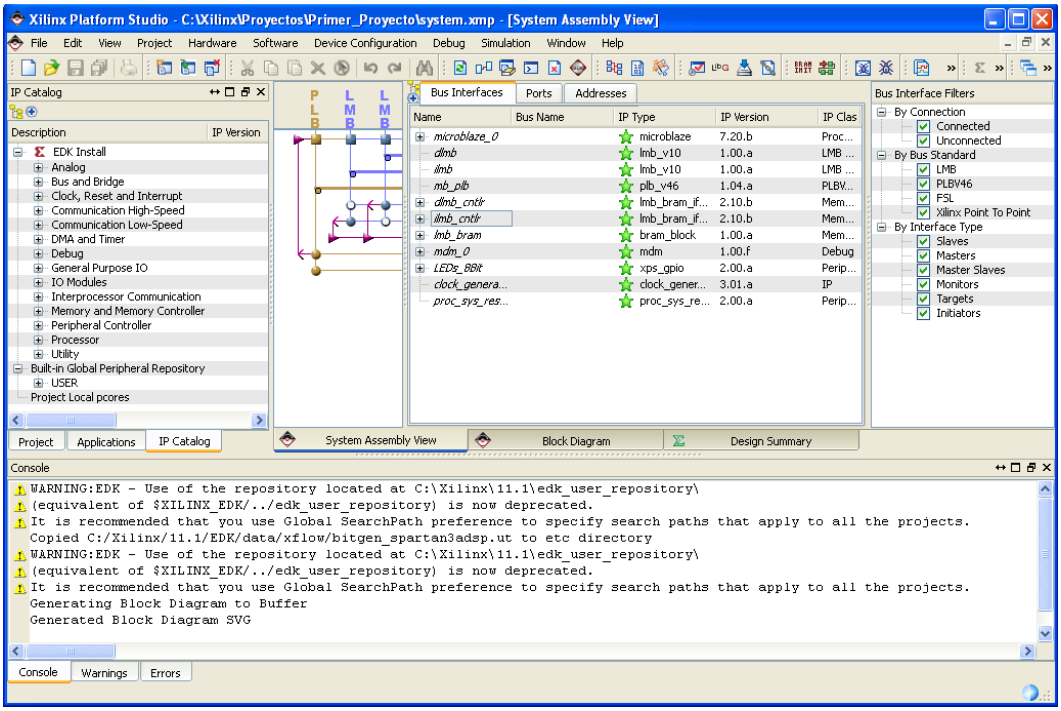

Figura 13. Interfaz XPS

La ventana principal de la Interfaz *XPS* está dividida en tres partes principales, a la izquierda se encuentra el área de información del proyecto, la cual está compuesta por tres pestañas:

- *Project*, en esta opción se encuentran diferentes archivos referentes al proyecto, desde donde el usuario los puede abrir y modificar (*Project. Files*), en *Project Options* se encuentra la descripción de la implementación del proyecto (plataforma, lenguaje de descripción de hardware, etc.), y *Design*

Summary en la cual accedemos en la parte derecha de la interfaz a un cuadro resumen de los elementos y de la configuración del proyecto.

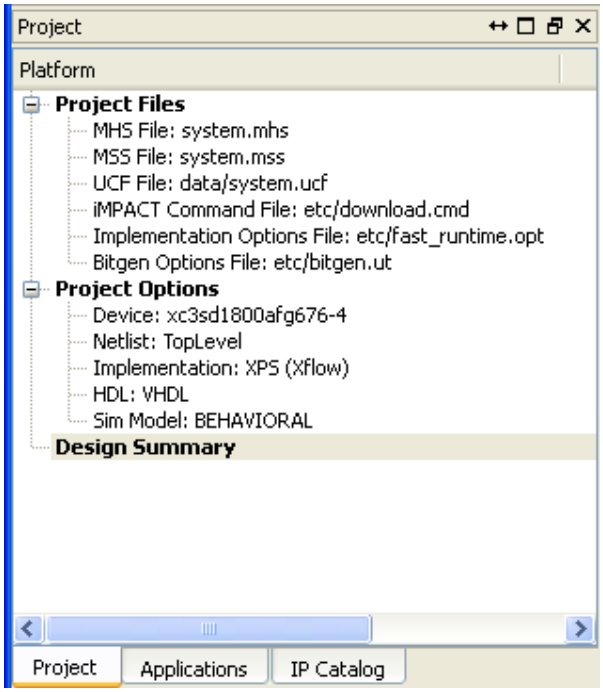

Figura 14. Pestaña Project

- *Applications*, aquí es donde se adhieren y administran las aplicaciones software del proyecto.

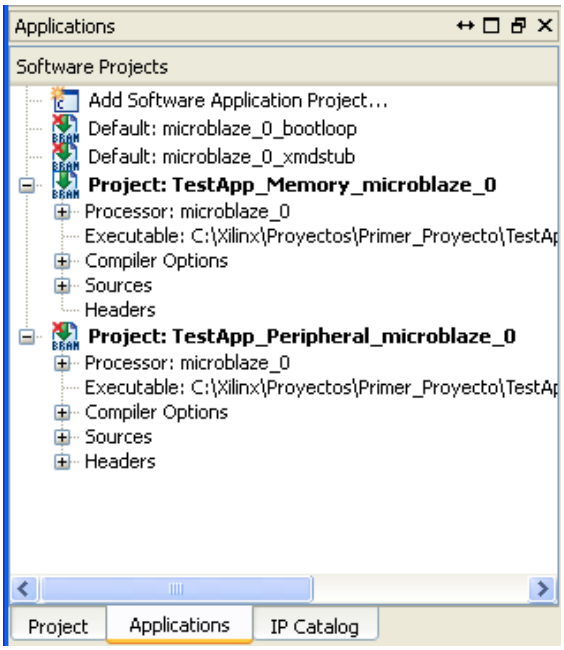

Figura 15. Pestaña Applications

- *IP Catalog*, es acá donde aparecen los IP Cores (drivers para uso de periféricos, memorias, buses, etc.) disponibles para ser usados en el proyecto.

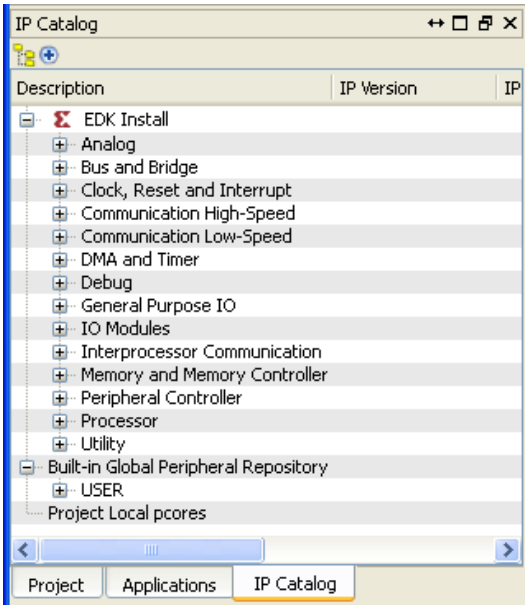

Figura 16. Pestaña IP Catalog

A la derecha de la interfaz se encuentra una ventana compuesta por tres pestañas nuevamente. La primera de estas es *System Assembly View*, en donde se encuentran diferentes configuraciones del proyecto. Esta pestaña está dividida en tres partes:

- *Bus Interfaces*, como se muestra en la siguiente figura, esta opción nos permite observar los componentes del sistema y sus respectivas interconexiones por medio de los buses disponibles en el proyecto. A la derecha de esta ventana se encuentra una aplicación de aplicar filtros a esta vista para tener una apreciación mas especifica de las conexiones.

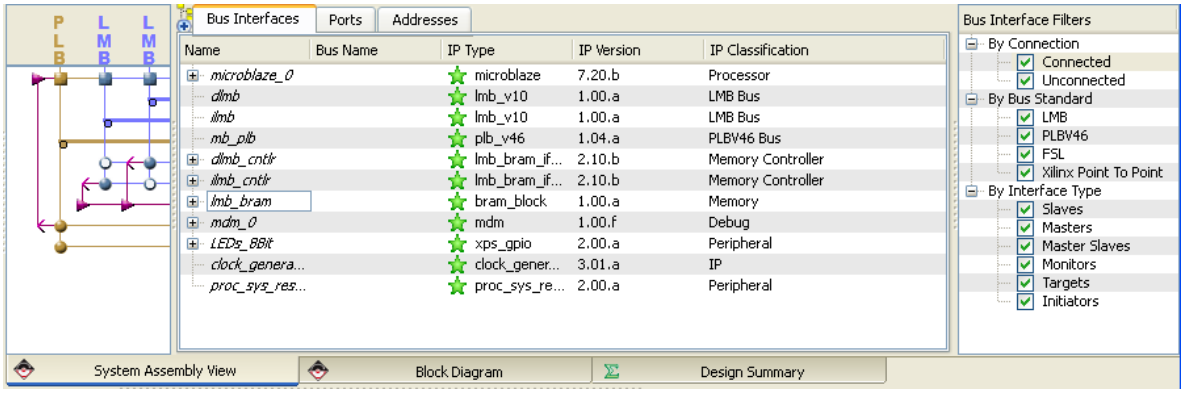

Figura 17. Ventana de Conexión de Buses

- *Ports*, enseña las conexiones de los puertos de cada uno de los elementos del proyecto, permitiendo su configuración y modificación, así como la descripción de los puertos externos (*External Ports*). De nuevo un filtro de puertos permite reducir la vista a necesidad del usuario.

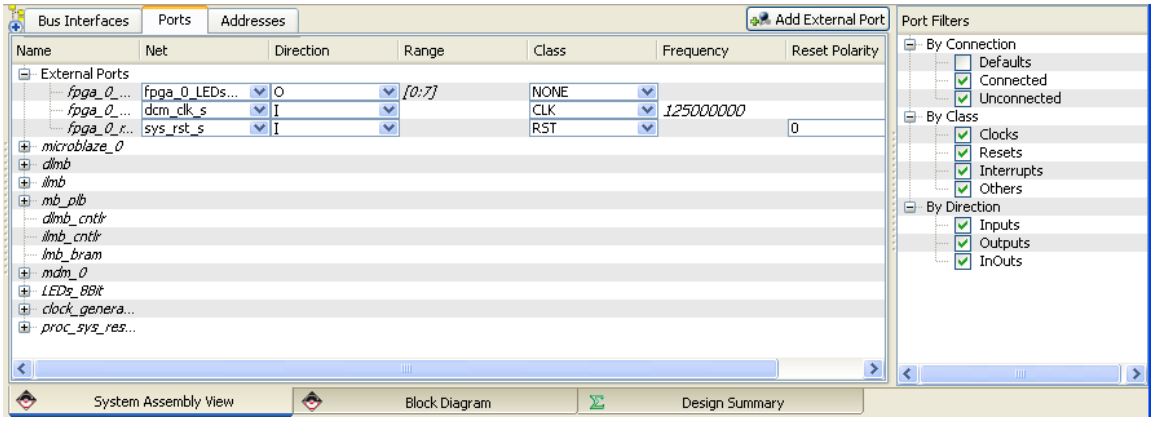

Figura 18. Puertos del Sistema

- *Addresses*, muestra el tamaño y las direcciones de las porciones de memoria usada por cada uno de los elementos del proyecto. Los rangos de memoria pueden ajustarse manualmente al modificar *Base Address* y *High Address*, o si se quiere una asignación automática (recomendada) basta con aplicar la opción *Generate Addresses*.

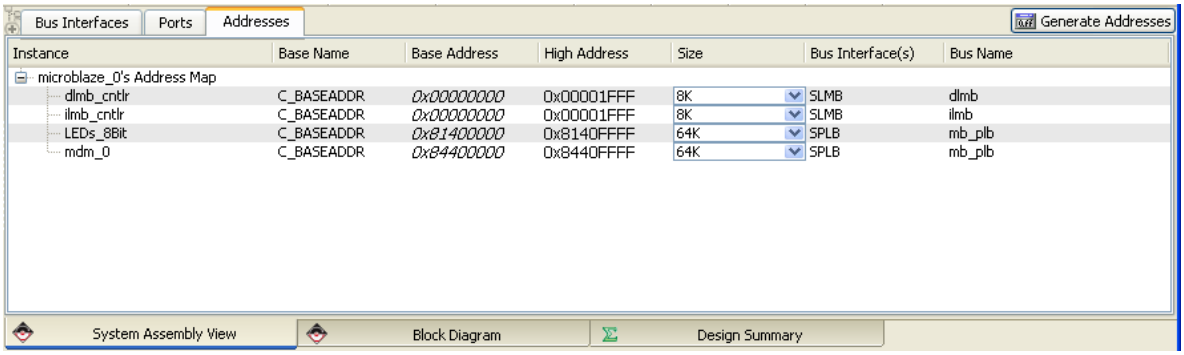

Figura 19. Mapeo de Memoria

La segunda pestaña de la parte derecha de la interfaz *Block Diagram* muestra la estructura del proyecto generado (Microblaze, periféricos, memorias, etc.)

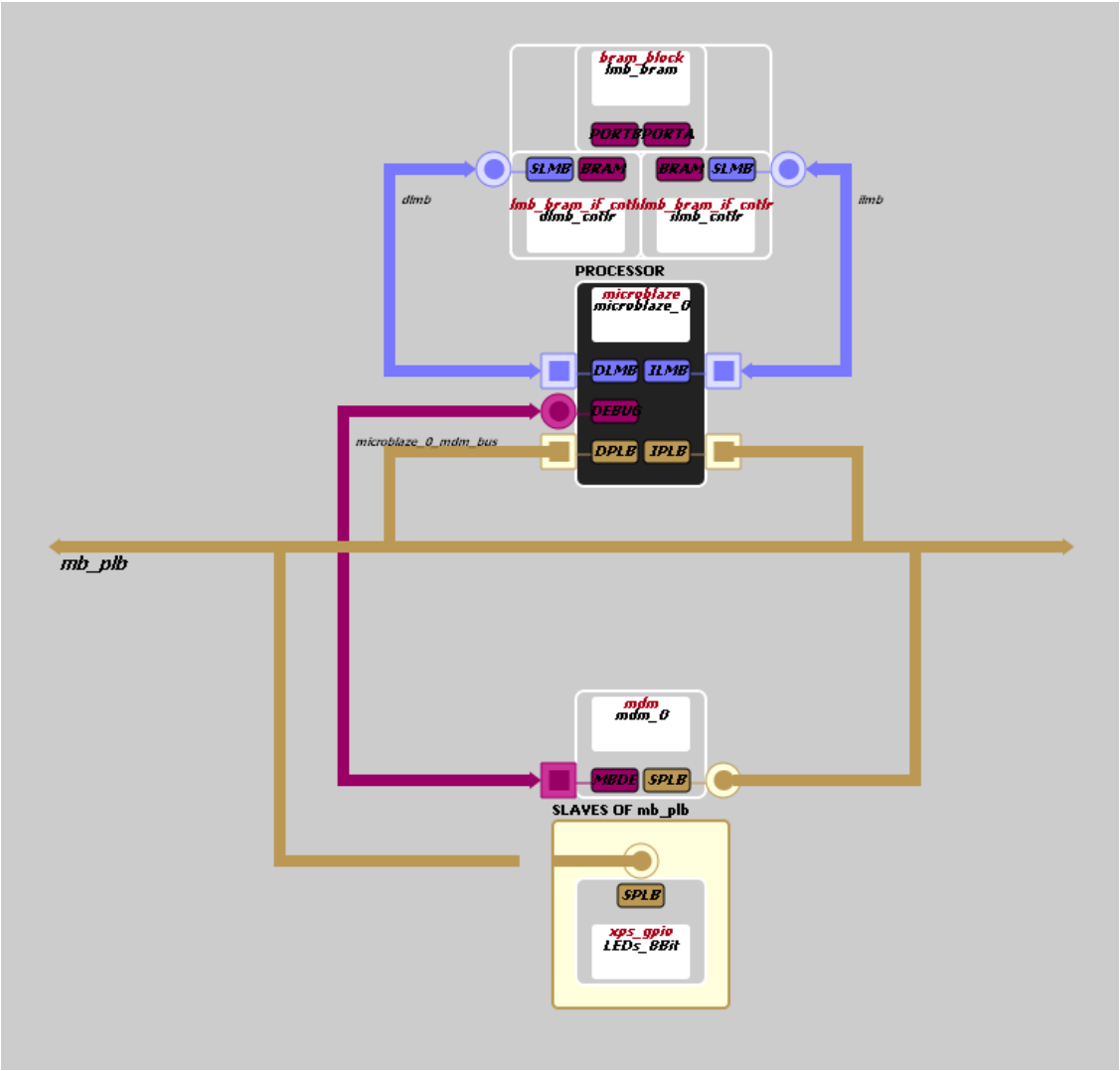

Figura 20. Diagrama de Bloques del Proyecto

La tercera y última pestaña *Design Summary*, muestra un resumen de los reportes de todo lo relacionado con el proyecto como errores, warnings y detalles de configuración.

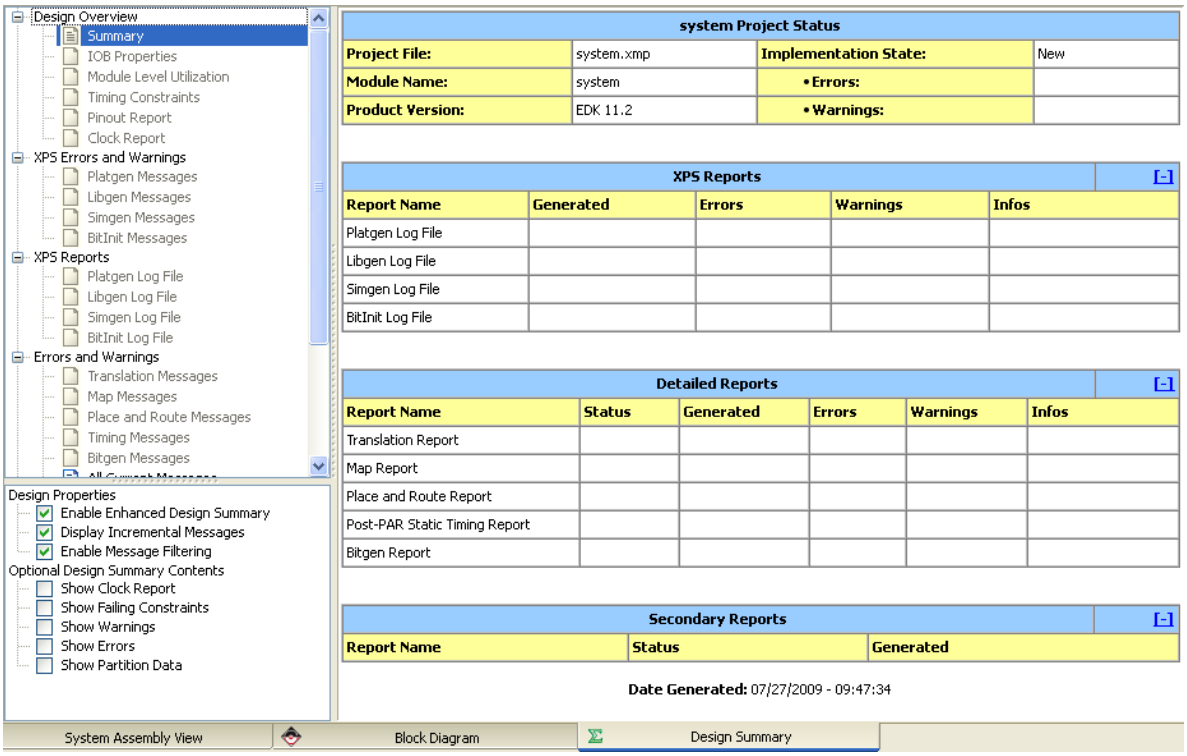

Figura 21. Resumen de Diseño

Para finalizar la interfaz XPS, una consola de mensajes aparece en la parte inferior de la misma, la cual consta de tres pestañas. La pestaña *Console* muestra toda la información referente al proyecto, *Warnings* muestra exclusivamente las advertencias y sugerencias generadas directamente por XPS o por algún proceso realizado; finalmente *Errors* muestra los errores cometidos en cualquier proceso realizado.

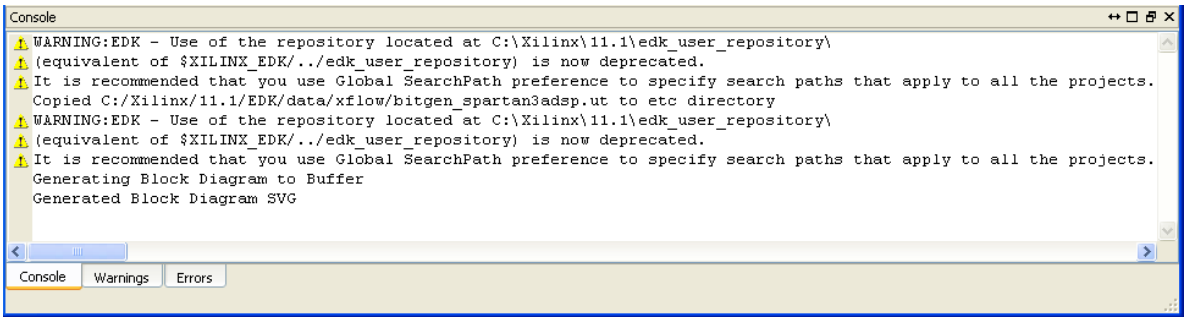

Figura 22. Consola de Mensajes

Para este diseño no es necesaria la inclusión de un nuevo IP Core o periférico, ya que el único necesario (LED`s) fue incluido desde el *Base System Builder.* Como ejemplo de entrenamiento se propone la inclusión de un periférico local (GPIO), que posteriormente se eliminará, se recomienda revisar cada uno de los pasos explicados a continuación para los LED´s ya incluidos (*LEDs_8bit*), ya que estos son necesarios para cualquier periférico.

Para incluir el periférico, basta con buscar en la pestaña *IP Catalog* y arrastrarlo hasta la ventana de la parte derecha de la interfaz. En este caso se incluye el periférico *xps_gpio_0* que encontramos en la opción *General Purpose IO* de la pestaña *IP Catalog*

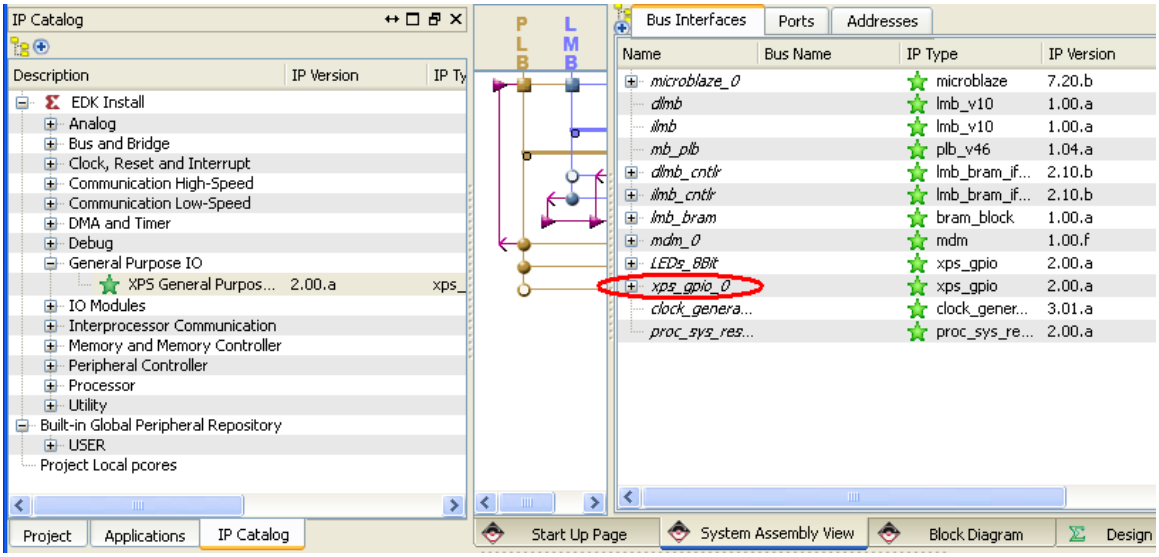

Figura 23. Inclusión de Periférico

El periférico añadido debe conectarse al bus del procesador (PLB, Processor Local Bus), para esto se expande el periférico desde la pestaña *Bus Interface* y en la opción emergente *SPLB* se selecciona mb_plb, la conexión correcta se evidencia marcada en la figura siguiente.

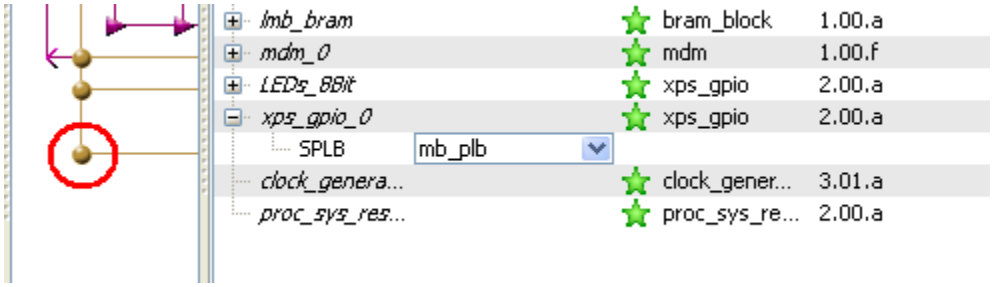

Figura 24. Conexión del Periférico al PLB

Cada periférico tiene su respectiva configuración, para observar o modificar estas opciones es necesario dar clic derecho sobre el periférico deseado y escoger la opción *Configure IP*. En la ventana emergente se puede configurar especificaciones como el ancho del canal de datos, tipo de datos (entrada/salida), etc. Es importante que se revise la configuración para cada periférico incluido en el proyecto desarrollado, esto para asegurarse de trabajar con los valores correctos y no llegar a cometer errores en etapas posteriores del diseño.

Además de la configuración del periférico con clic derecho se puede acceder a diferentes documentos como el datasheet del periférico, y demás archivos de configuración de puertos y drivers respectivos, siendo muy importante la opción *View API Documentation* dentro de *Driver*. Es acá donde se define el periférico, se listan y especifican las funciones y variables que precisan su funcionamiento.

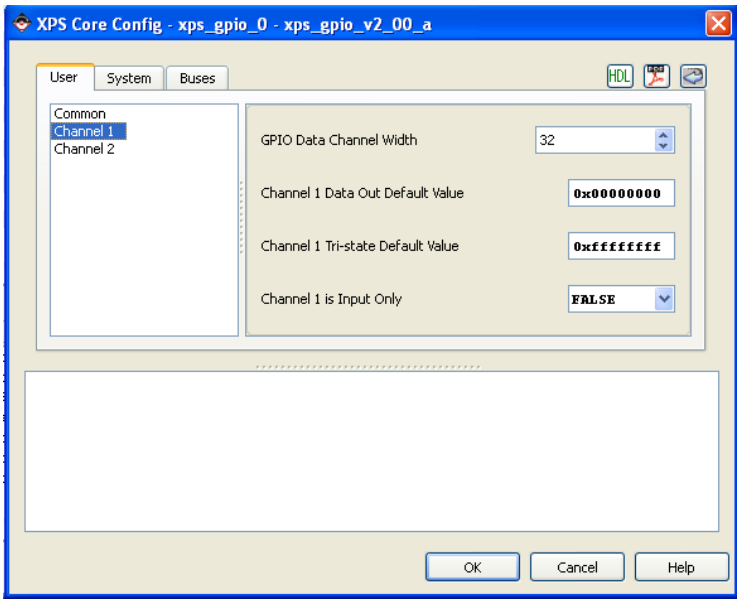

Figura 25. Configuración del GPIO

Luego de configurar el periférico añadido y conectarse a los diferentes buses, es necesario hacer externos los puertos del periférico, para esto se selecciona la pestaña *Ports* y se expande el periférico xps_gpio_0 y dependiendo del uso que se le va a dar al periférico (solo salida), entonces se escoge *GPIO_IO_O* y tomamos la opción *Make External*; esto significa que el puerto tendrá una conexión a un pin externo de la FPGA para ser conectado a cualquier dispositivo disponible.

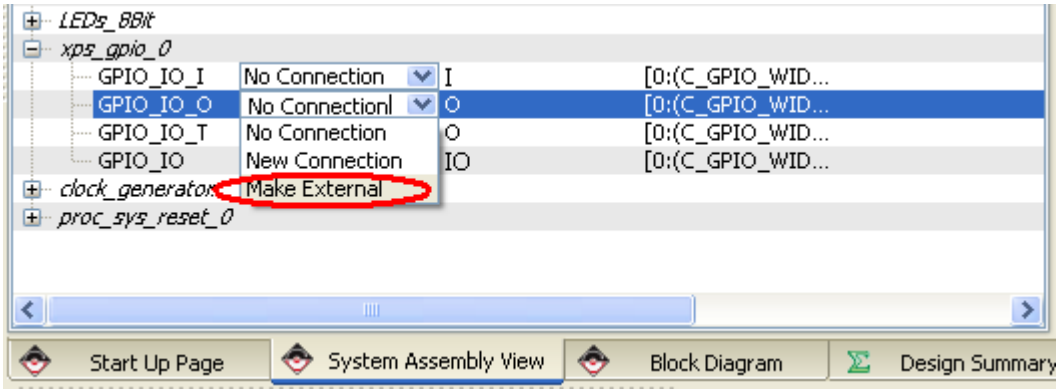

Figura 26. Conexión Externa del Periférico

Para asegurarse de la correcta conexión del periférico, dentro de la opción *External Ports* deberá aparecer de la siguiente forma, donde se aprecia el periférico como salida, tamaño en bits, etc; parámetros definidos en la previa configuración.

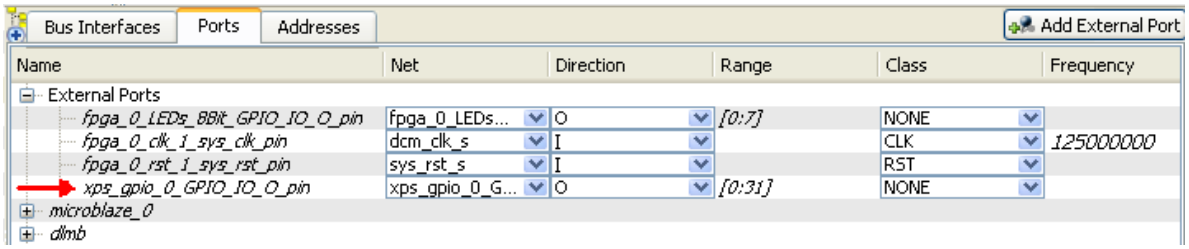

Figura 27. Confirmación de Conexión Externa del Periférico

Para culminar esta etapa falta asignar direcciones de memoria al nuevo periférico. Al situarse sobre la pestaña *Adresses* se observa que el mapeo de memoria para el periférico incluido no está definido.

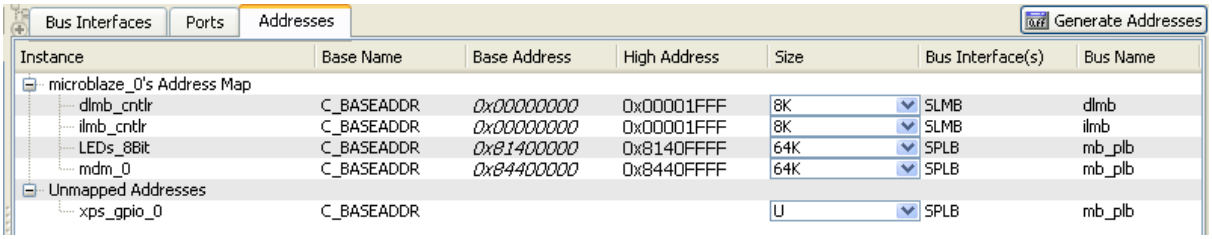

Figura 28. Mapeo incompleto de Memoria

Para asignar una porción de memoria al periférico basta con dar clic en la opción *Generate Adresses*. La siguiente figura muestra el mapeo completo.

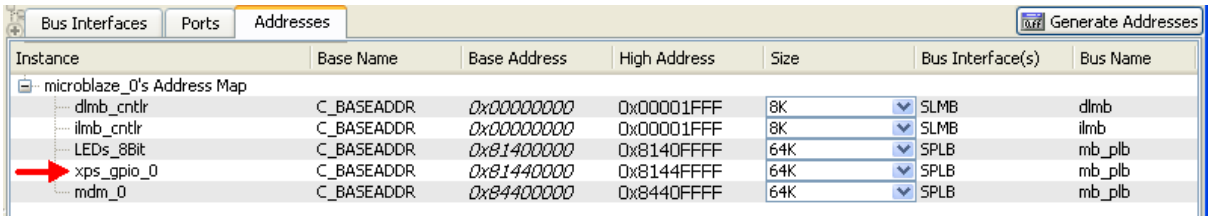

Figura 29. Mapeo finalizado

En este punto solo falta agregar en el archivo de restricciones (.ucf) las conexiones de los pines de la FPGA con los puertos externos (LED`s, Push Buttom`s, etc). A este archivo se accede desde la pestaña *Project*, en la opción *Project Files* y UCF file. Con doble clic sobre esta opción se muestra este archivo de texto en la parte derecha de la interfaz.

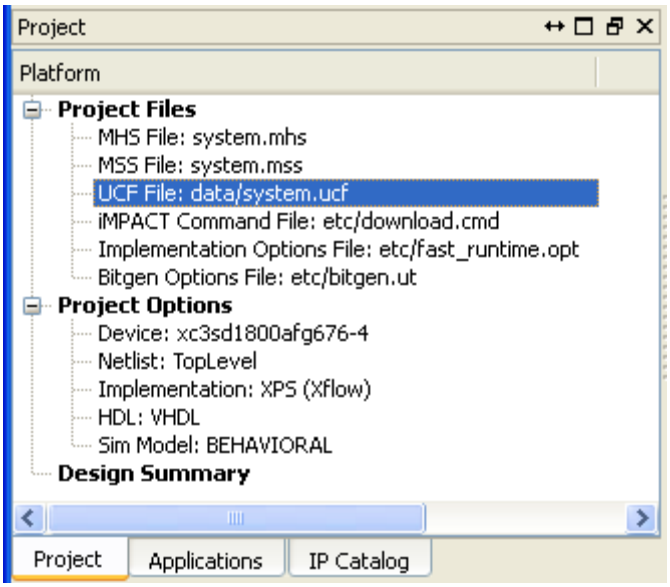

Figura 30. Acceso al Archivo UCF

Este archivo contiene las restricciones creadas por la herramienta (de defecto mas las necesarias para la utilización de los LED`s). Como parte del ejemplo de la

adicción de un periférico no se crearan las restricciones para este, se pone a consideración del usuario la estructura básica de una restricción:

Net “Nombre del Puerto”<”bit”> LOC = “Puerto externo” | IOSTANDARD=“Nivel de Voltaje”;

Ejemplo para el bit “0” de los LED’s:

Net fpga_0_LEDs_8Bit_GPIO_IO_O_pin<0> LOC = D25 | IOSTANDARD=LVTTL;

La sintaxis como se observa de una restricción debe comenzar con la palabra “Net”, seguido de un espacio y el nombre del puerto tal como aparece en la pestaña *Ports* opción *External Ports*, seguido del numero de bit encerrado entre Menor que (<) y Mayor que (>), recuerde que debe asignarse una restricción para cada uno de los bit’s de longitud del puerto (*Range [0:X]*), separada a un espacio la palabra “LOC” es igualada al puerto externo en la Plataforma (este tiene una notación letra-numero, y se encuentra disponible en la plataforma pero por comodidad se puede encontrar la lista completa en el datasheet de la Plataforma). Separado por el carácter (|) se especifica el nivel de voltaje establecido para el puerto, por defecto se utiliza el mostrado en el ejemplo.

En este punto el sistema está listo para generar el archivo binario para implementar en la plataforma (Bitstream), no se olvide eliminar el periférico añadido como ejemplo, para lograr esto basta con dirigirse a la pestaña *Bus Interface*s, seleccionar el periférico, dar clic derecho y seleccionar la opción *Delete Instance…* , y se escoge la opción *Delete instance and its ports (both internal and external)* que elimina en su totalidad todo lo referente o creado acerca del periférico.

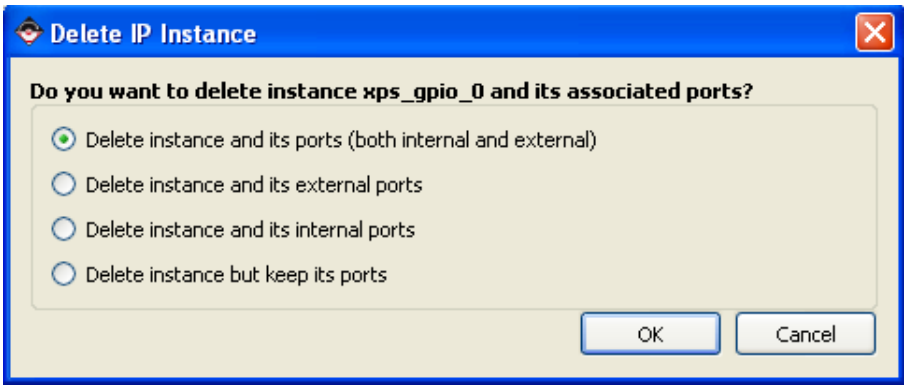

Figura 31. Ventana de Eliminación de Periférico

Siguiendo con el flujo de diseño se procede a sintetizar el diseño, para ello en el menú *Hardware* se selecciona la opción *Generate Bitstream*.

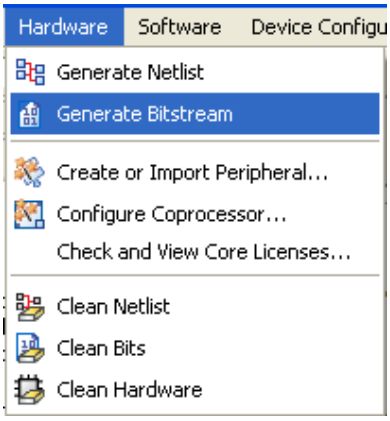

Figura 31. Generación del Bitstream

Este proceso puede durar varios minutos dependiendo del equipo utilizado y de la complejidad del diseño, el progreso de este puede observarse en la ventana inferior *Console Window*. Si no han ocurrido errores este proceso debe terminar en:

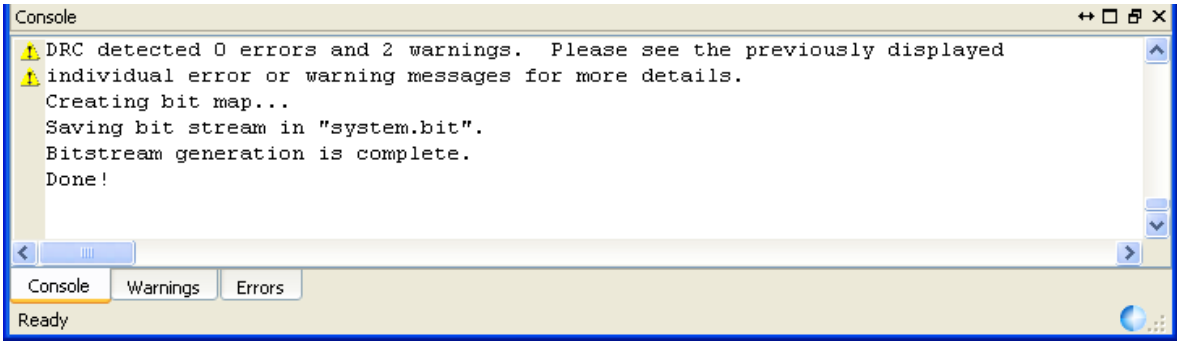

Figura 32. Bitstream Completado Satisfactoriamente

El archivo creado (*.bit), es el que contiene toda la configuración para la FPGA de la Plataforma.

Aplicación Software

Ya que el sistema está listo, se tiene que agregar la aplicación software que va a realizar la Plataforma. Se pretende probar el funcionamiento de la plataforma mediante el encendido y apagado de los LED`s disponibles, periódicamente. Para añadir un nuevo proyecto software se da doble clic en la opción *Add Software Application Project…* de la pestaña *Applications*.

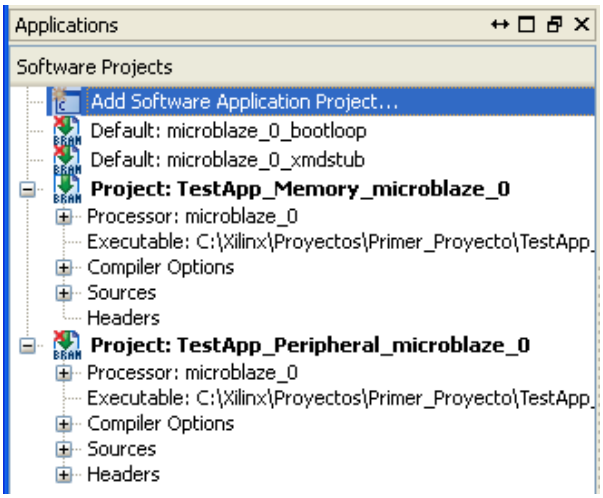

Figura 33. Adición de Aplicación Software

La ventana emergente permite nombrar la aplicación a crear; clic en *OK* para continuar.

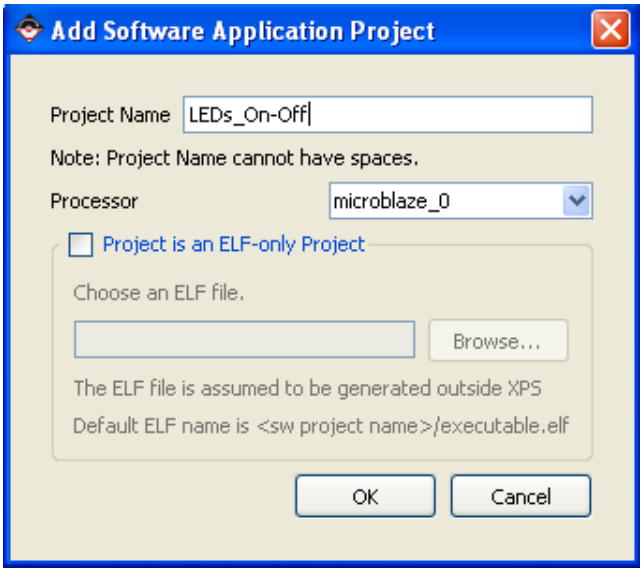

Figura 34. Creación de Aplicación Software

Completada esta etapa, en la pestaña *Applications* aparece el árbol del proyecto creado, en el cual se debe configurar el compilador asociado, Header`s e importar la aplicación en lenguaje C.

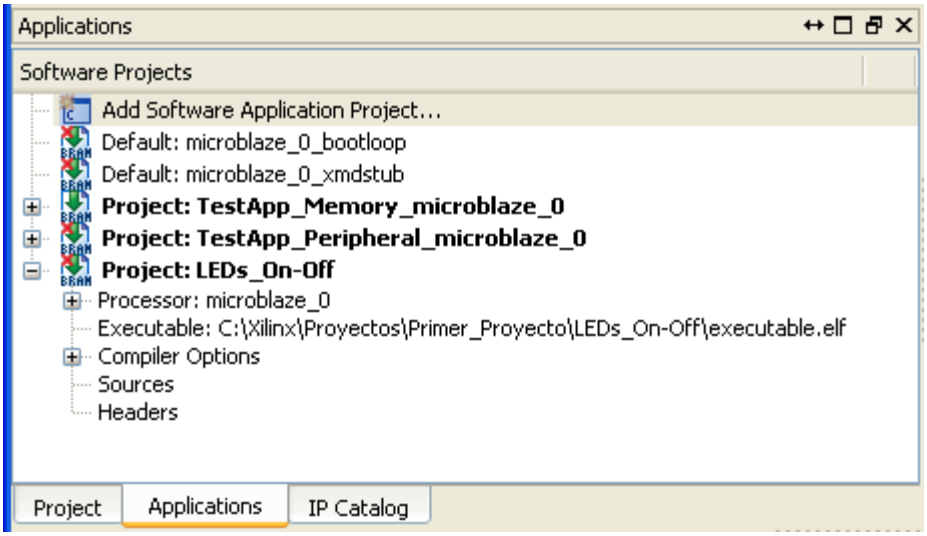

Figura 35. Árbol del Proyecto Software

La estructura de los códigos a implementar está determinada por los lineamientos del lenguaje C. El código de la aplicación es mostrado a continuación, nótese el uso de funciones especiales que dependen del periférico usado, estas se encuentran definidas en los driver`s de cada periférico. Los Header`s “xparameters.h”, “stdio.h” y “xutil.h” se usan por defecto y se recomienda su inclusión en cada código elaborado; finalmente “xgpio.h” es necesario para el correcto funcionamiento del periférico (funciones).

```
#include "xparameters.h"
#include "stdio.h"
#include "xutil.h"
#include "xgpio.h"
//====================================================
int main (void) {
XGpio led8bits;
```

```
int i;
XGpio_Initialize(&led8bits, XPAR_LEDS_8BIT_DEVICE_ID);
XGpio_SetDataDirection(&led8bits, 1, 0x00000000);
while (1)
      {
        XGpio_DiscreteWrite(&led8bits, 1, 0x00);
        for(i=0;i<999999;i++);
        XGpio_DiscreteWrite(&led8bits, 1, 0xFF);
        for(i=0;i<999999;i++);
        }
}
```

Es necesario que este código sea copiado a un archivo en un procesador de texto, y que este sea guardado en una ubicación dentro del mismo disco duro donde se instalaron los componentes de Xilinx (ISE y EDK), se recomienda crear una carpeta para almacenar este tipo de archivos (por ejemplo C:\Códigos). No olvidar al momento de guardar el archivo terminarlo en ".c" (Nombre.c).

Dando doble clic sobre la opción *Sources* del árbol del proyecto creado, aparece una ventana de búsqueda para cargar el archivo de código generado recientemente. Luego de este procedimiento el proyecto estará de la siguiente manera:

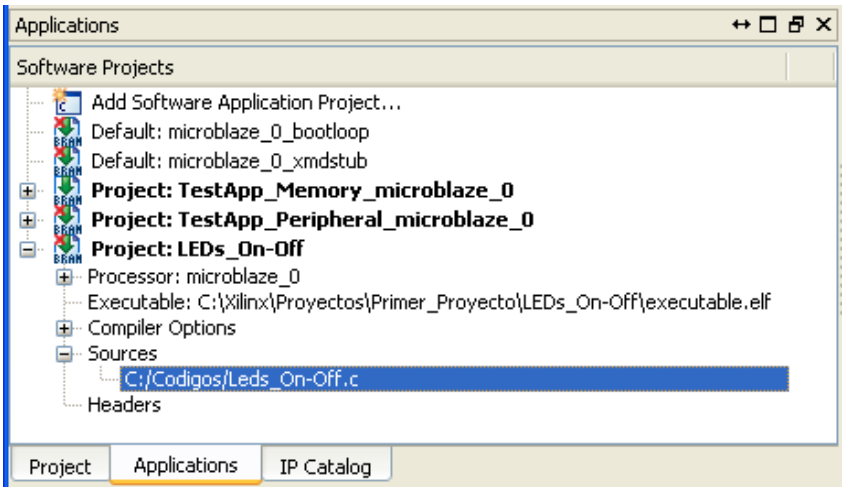

Figura 36. Código de la Aplicación Adherido

Recuerde que cualquier archivo de las pestañas *Project*, *Applications* e *IP Catalog* puede abrirse con doble clic sobre él, este aparece en la parte derecha de la interfaz para realizar cualquier modificación necesaria.

Otro paso importante es asegurarse que todas las inicializaciones de periféricos (XGpio_Initialize(&led8bits, XPAR_LEDS_8BIT_DEVICE_ID);) incluidas en el código fuente estén dentro del archivo “xparameters.h”, para esto generamos librerías desde el menú *Software* como se muestra en la figura 37.

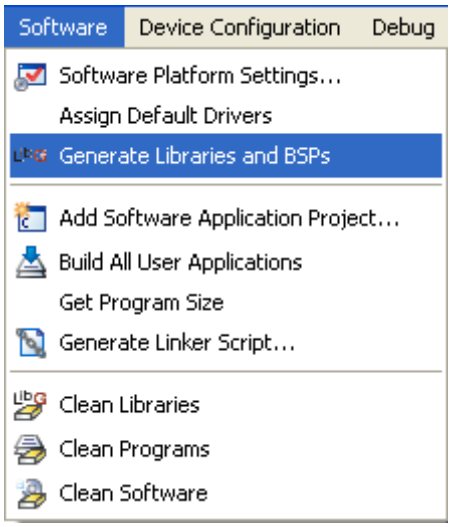

Figura 37. Generación de Librerías

Luego de terminado este proceso el árbol del proyecto queda de la siguiente manera:

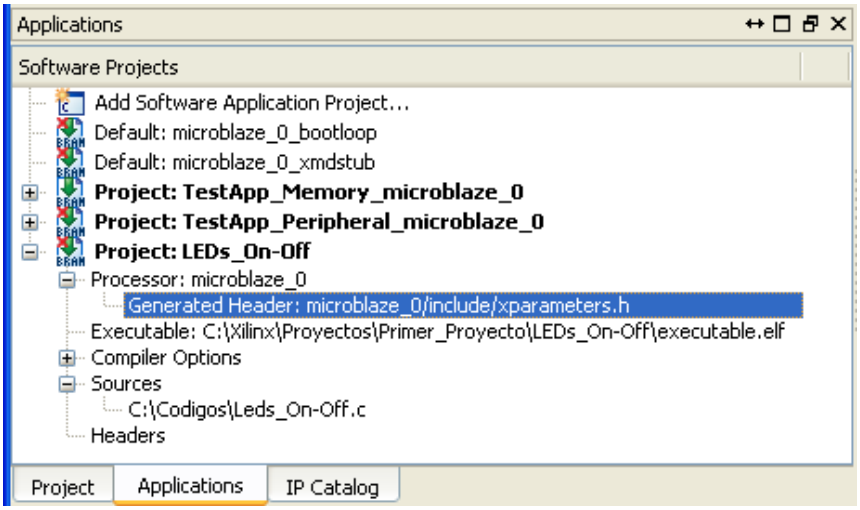

Figura 38. Archivo “xparameters.h” Generado

Al acceder al archivo “xparameters.h” deben aparecer las siguientes líneas para que el periférico funcione debidamente:

```
/* Definitions for peripheral LEDS_8BIT */
#define XPAR_LEDS_8BIT_BASEADDR 0x81400000
#define XPAR_LEDS_8BIT_HIGHADDR 0x8140FFFF
#define XPAR_LEDS_8BIT_DEVICE_ID 0
#define XPAR_LEDS_8BIT_INTERRUPT_PRESENT 0
#define XPAR_LEDS_8BIT_IS_DUAL 0
```

Ahora es necesario configurar las opciones del compilador, para esto se da clic derecho sobre *Compiler Options* dentro del árbol del proyecto y seleccionamos *Set Compiler Options…* . Únicamente es necesario cambiar el nivel de optimización en *Optimization Parameters* a *No Optimization* en la pestaña *Debug and Optimization*. Las demás opciones de configuración disponibles quedan a disposición y necesidad del usuario.

Seguido se compila la aplicación desarrollada, basta con oprimir clic derecho sobre la rama principal del proyecto software creado y seleccionar *Build Project* para compilar la aplicación.

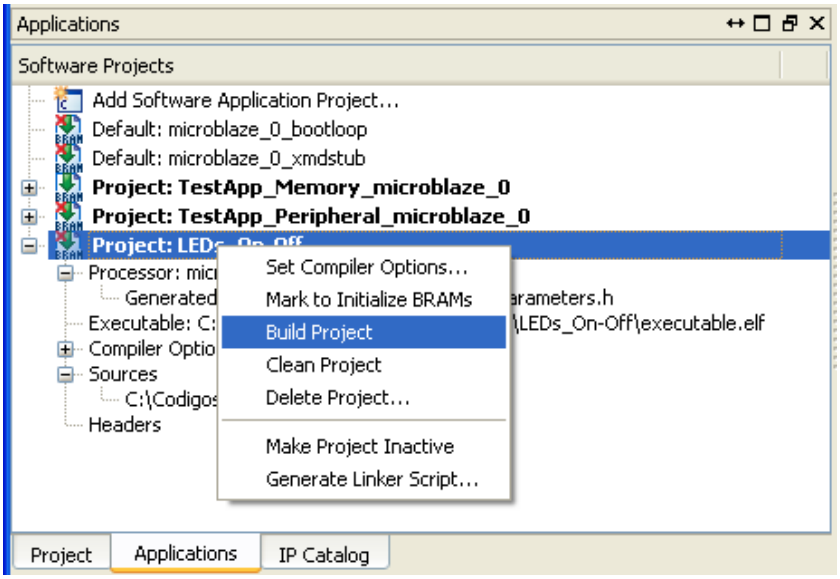

Figura 39. Compilación del Proyecto

Si la compilación termina correctamente, confirmando en la ventana de *Console Window*, se debe inicializar el bloque de memoria dando clic derecho sobre el proyecto nuevamente y escogiendo *Mark to Initialize BRAMs*. Es necesario eliminar de la memoria de bloque del sistema el *bootloop* que aparece incluido por defecto, esto se logra al hacer clic derecho sobre *Default: microblaze_0_bootloop* y seleccionando nuevamente *Mark to Initialize BRAMs*.

Concluido el manejo del proyecto software, la pestaña *Applications* tiene el siguiente aspecto:

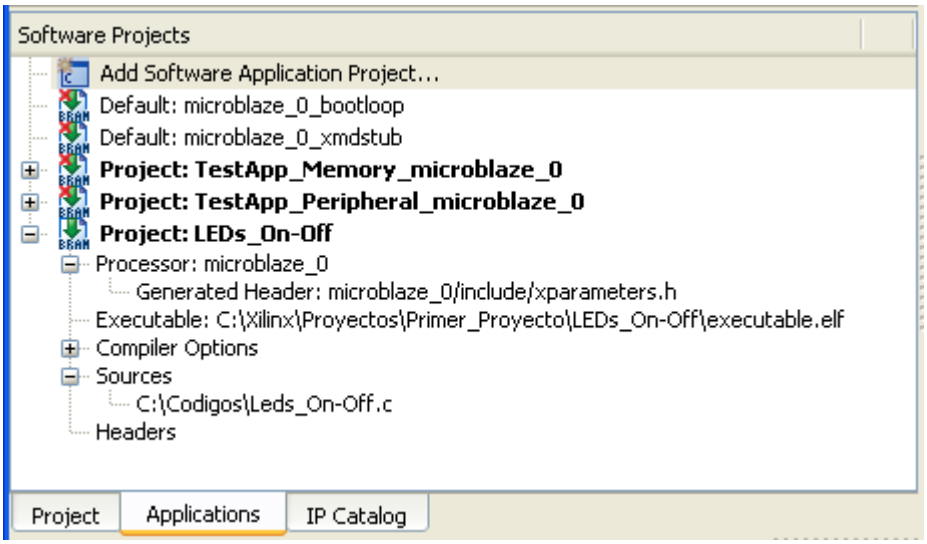

Figura 40. Proyecto Software Terminado

Seguido se actualiza el Bitstream ya generado con los cambios introducidos por la aplicación software. Para ello se debe escoger la opción *Update Bitstream* del menú *Device Configuration*.

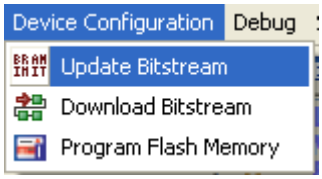

Figura 41. Actualizar Bitstream

Antes de realizar la programación de la Plataforma, es necesario que la misma esté preparada. Para esto se debe asegurar que la Plataforma está conectada a la alimentación de 5 Voltios y el switch de encendido (SW1) se encuentre en la posición ON, este se encuentra en la parte superior derecha de la Plataforma:

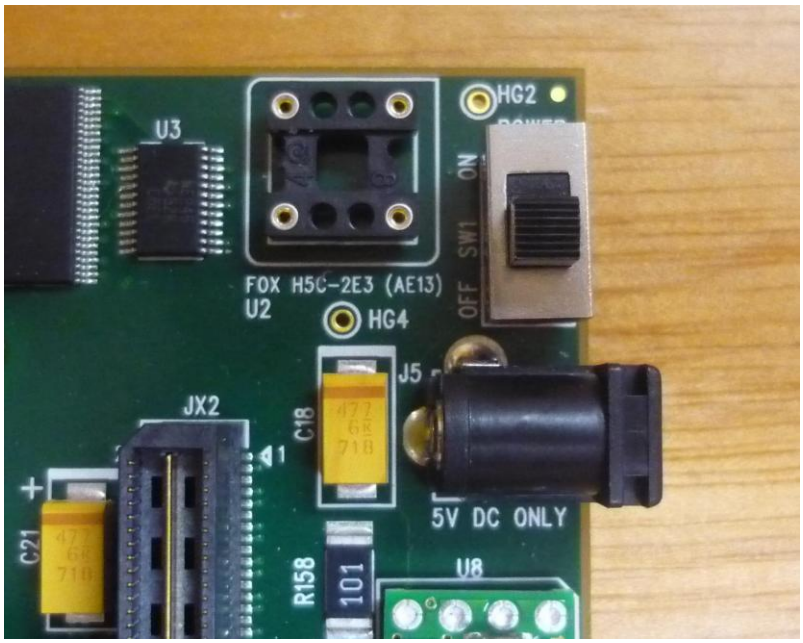

Figura 42. Alimentación y Switch de Encendido de la Plataforma

Así mismo la conexión Plataforma-Computador por medio del cable JTAG; no olvide que el modo de configuración para la programación de la Plataforma seleccionable mediante los jumper´s de Modo (JP9) debe estar en USB-JTAG (M0=Open, M1=Closed y M2=Open):

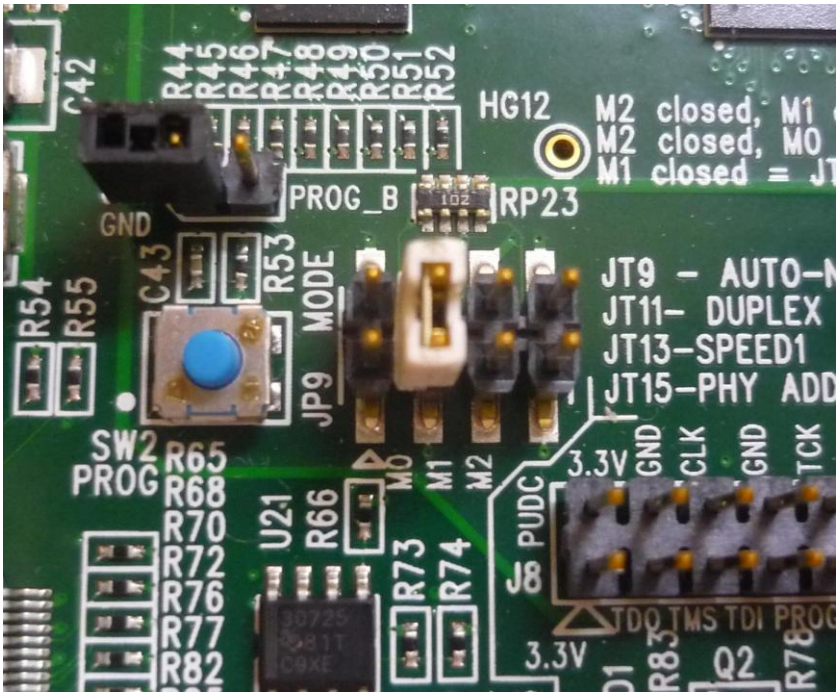

Figura 43. Jumper`s de Configuración

Luego de asegurarse de la correcta conexión de la Plataforma, esta se presenta de la siguiente manera estando lista para la programación:

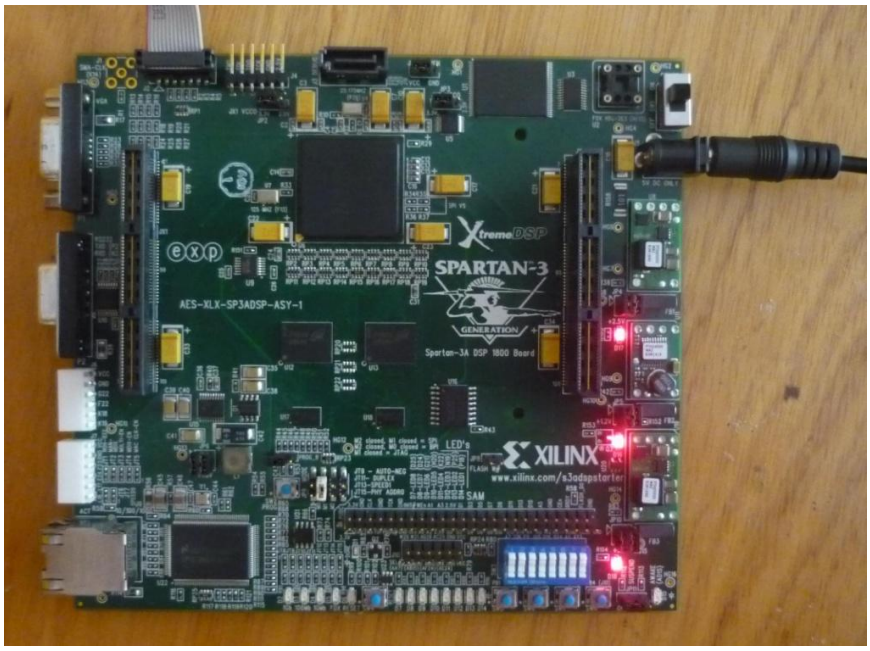

Figura 44. Plataforma Lista para Programación

Para programar la plataforma se da clic en *Download Bitstream*.

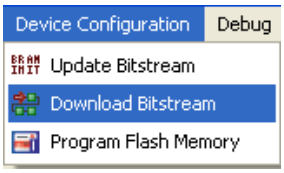

Figura 45. Programación de la Plataforma

Cuando se indique en el equipo que la programación ha sido satisfactoria, se enciende el Led de programación exitosa (DONE), nombrado D1, ubicado en la parte inferior central de la Plataforma; e inmediatamente la aplicación implementada comienza su funcionamiento.

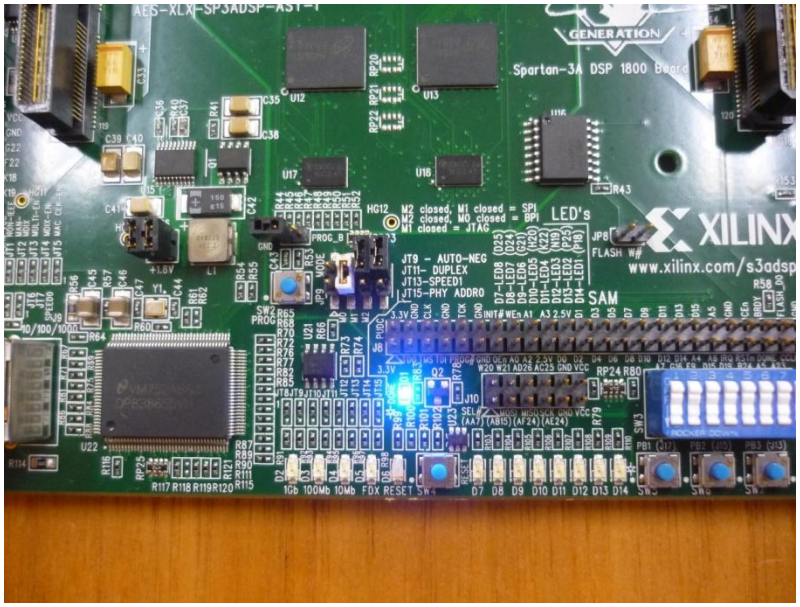

Figura 46. Programación Exitosa

El Switch (SW4) ubicado en la parte inferior central de la Plataforma es el RESET general. El diodo (D6) a la izquierda del pulsador de RESET es el indicador de

reset que se enciende momentáneamente como confirmación luego de pulsar RESET.

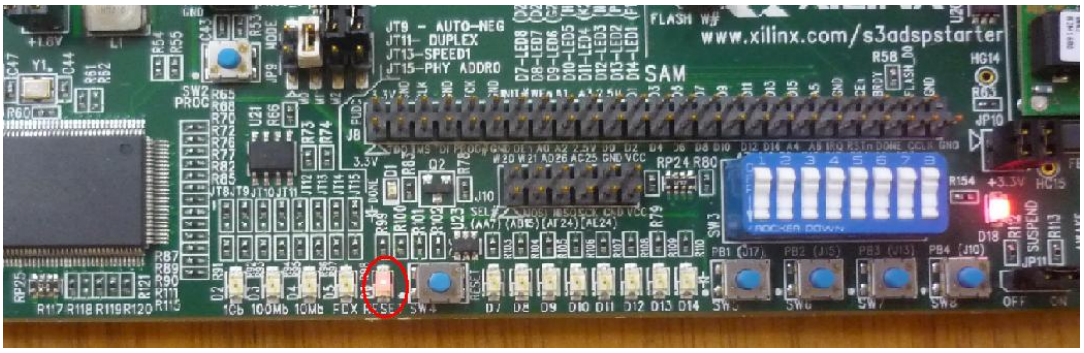

Figura 47. Reset de la Plataforma

Referencias

- www.xilinx.com

Anexo H

CREACIÓN DE UN IP CORE

En el presente documento se muestra una guía práctica para la generación de un IP CORE (periférico), para su utilización en los proyectos creados en la herramienta EDK. Estos son determinantes para el uso de periféricos externos, para los cuales el EDK no cuenta con los driver para su control y manejo.

El EDK proporciona la opción de crear un nuevo periférico o de importar uno ya creado. En el siguiente procedimiento se muestra la forma para la creación de un periférico externo genérico.

Para comenzar diríjase al menú ***Hardware*** y seleccione la opción ***Create or Import Peripheral***, como se muestra en la siguiente figura.

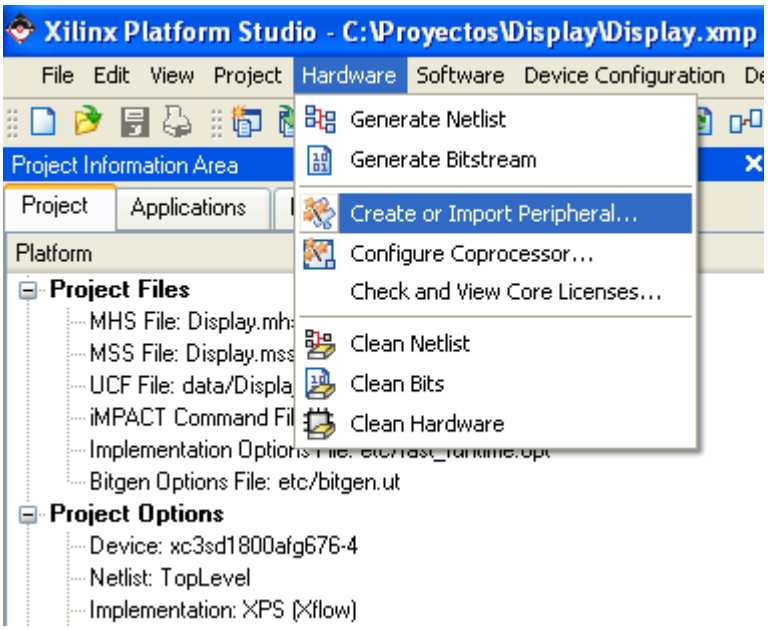

Figura 1. Ubicación del Wizard para la creación o importación de un periférico

A continuación aparece la ventana de bienvenida del Wizard, donde se debe dar clic en ***Next*** para continuar.

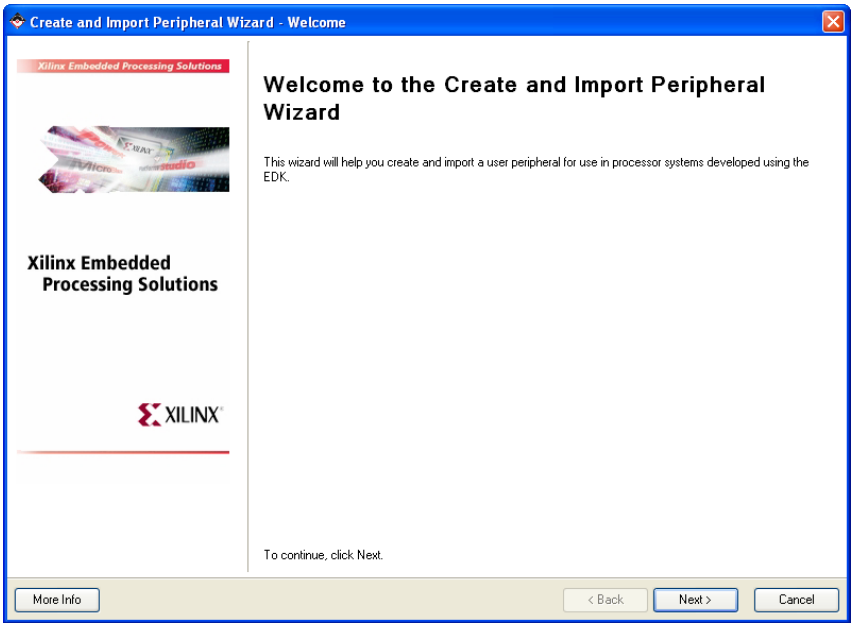

Figura 2. Ventana de bienvenida del Wizard

En la ventana emergente aparecen las opciones para seleccionar entre la creación de un nuevo periférico o la importación de uno ya existente (Select flow); además de la opción de la creación del periférico desde un lenguaje de descripción de hardware (HDL). Clic en ***Next*** para continuar.

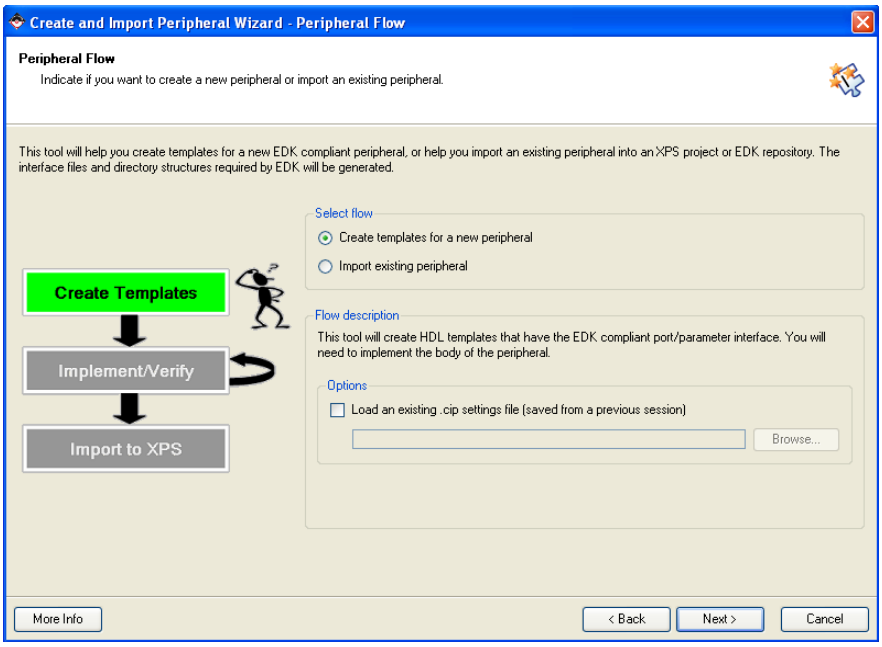

Figura 3. Ventana de Configuración del Wizard

Seguido se muestra otro ventana donde se pide donde se guardara el periférico creado, en una carpeta definida por el usuario (***To an EDK user repository***) con lo cual se podrá utilizar en cualquier otro proyecto; o en un proyecto activo (***To an XPS project***) con lo cual se adscribe a un proyecto ya generado.

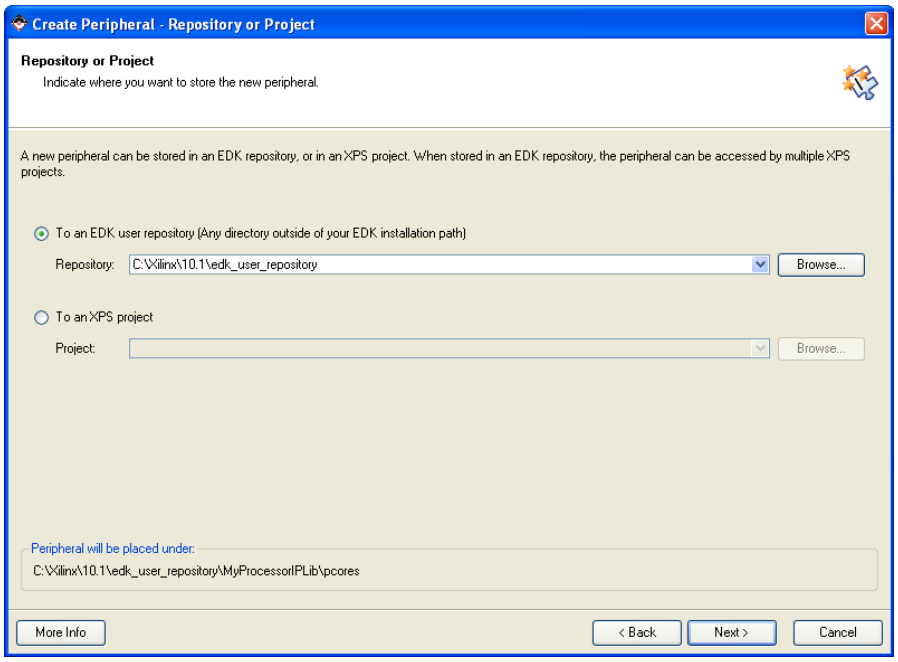

Figura 4. Ventana para el archivado del periférico

Para este caso se selecciona la primera opción en la cual guardara el periférico creado por defecto en la dirección C:\Xilinx\10.0\edk_user_repository. Clic en ***Next*** para continuar.

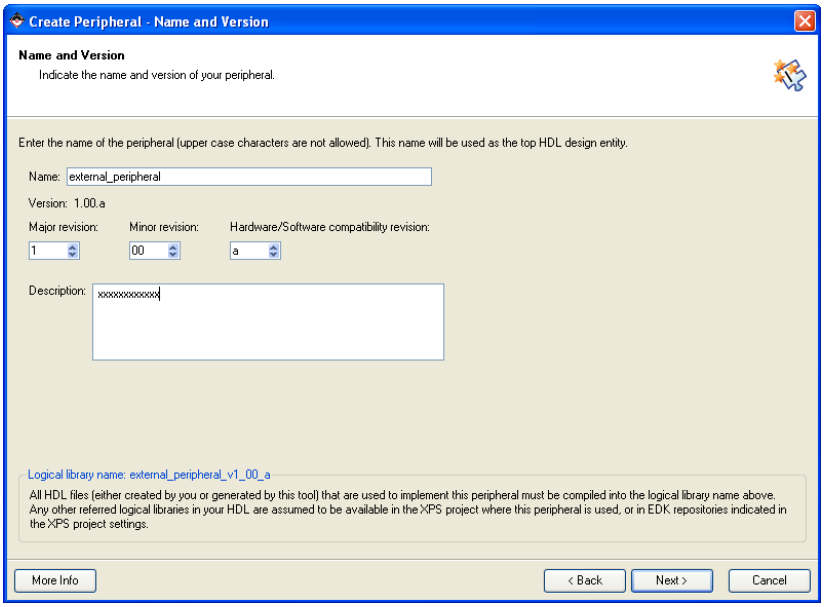

Figura 5. Ventana de nombrado de y descripción del periférico

En la ventana de la figura 5, se nombra al periférico a crear, el número de versión se deja por defecto y en descripción el usuario puede hacer una breve descripción de la funcionalidad del periférico. Clic en ***Next*** para continuar.

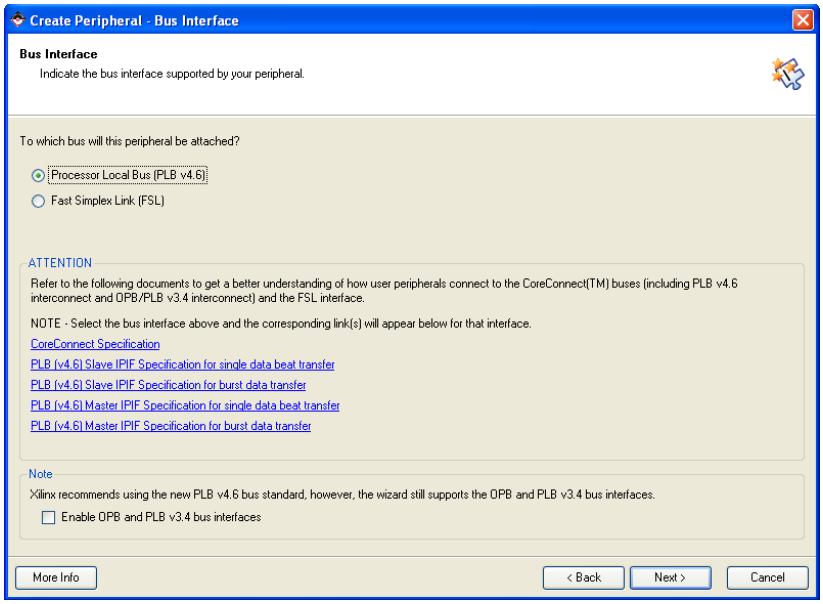

Figura 6. Ventana de interfaces de bus

En esta ventana aparecen las opciones para interconectar el periférico, las opciones disponibles son el Processor Local Bus (PLB) y el Fast Simplex Link (FSL). Con el PLB se conecta el periférico a cualquier otro periférico, procesador, etc; mientras que con el FSL se logra una conexión entre dos periféricos únicamente. Por defecto se selecciona el Processor Local Bus.

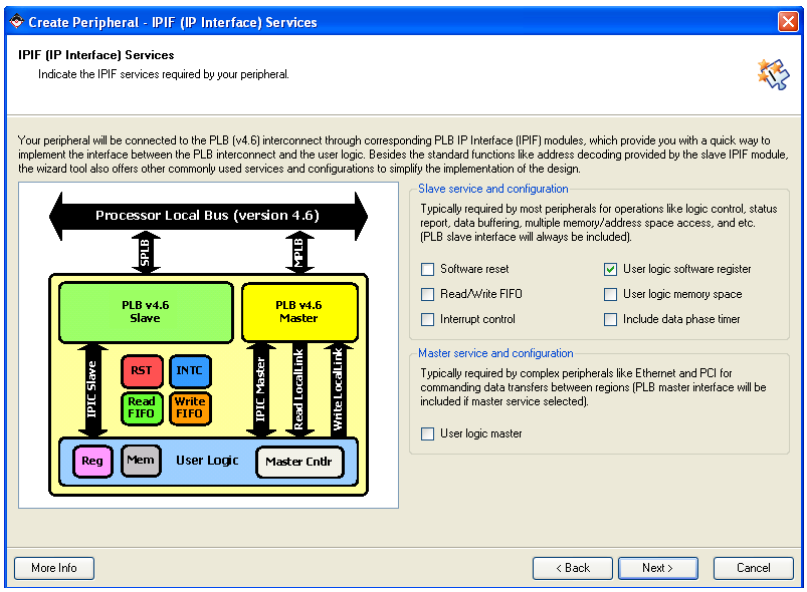

Figura 7. Conexión interna del periférico

En esta ventana se escogen las opciones de conectividad interna del periférico, como la utilización de reset, interrupciones, registros lógicos, espacio de memoria lógica, etc. Por defecto se deja esta ventana como aparece en la figura 7, únicamente seleccionando ***User logic software register***. Clic en ***Next***.

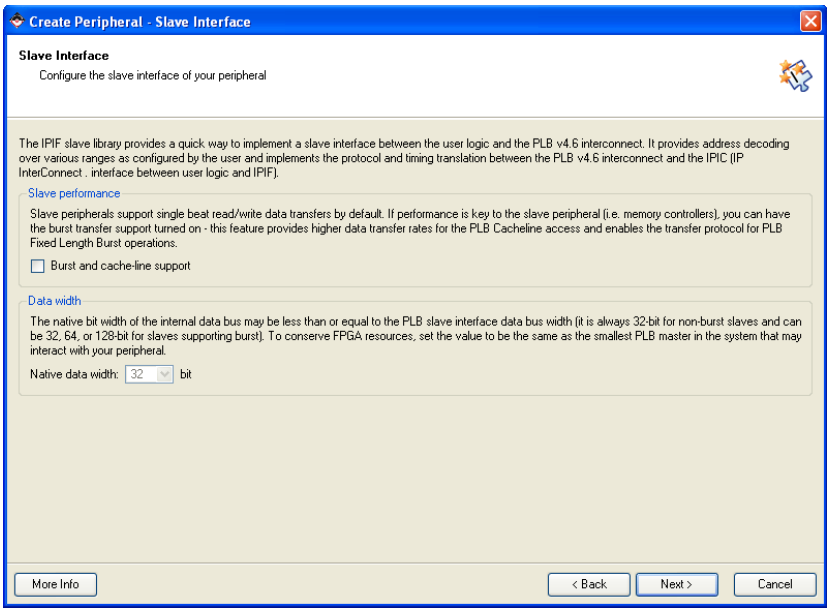

Figura 8. Interfaz esclava del periférico

El la ventana de al interfaz esclava del periférico, figura 8, permite la configuración entre el bloque esclavo del periférico y el PLB. Por defecto no se modifica esta configuración.

A continuación se requiere especificar la cantidad de registros S/H de control a usar. Dejando el número de registros en uno (1), a menos que se tenga certeza del número a utilizar en el proyecto, se continúa con clic en ***Next***. Estos registros son los encargados de la comunicación entre la aplicación software y el hardware del IP Core a generar.

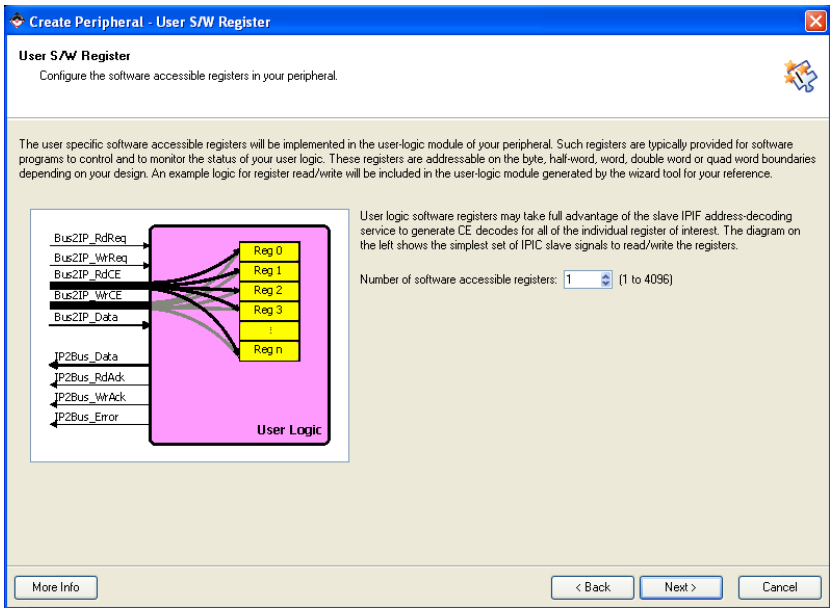

Figura 9. Registros del usuario

En la ventana de interconexiones del IP Core se muestran las conexiones disponibles entre el IP Core y el bus al cual este está conectado. Clic en ***Next*** para continuar dejando las conexiones por defecto.

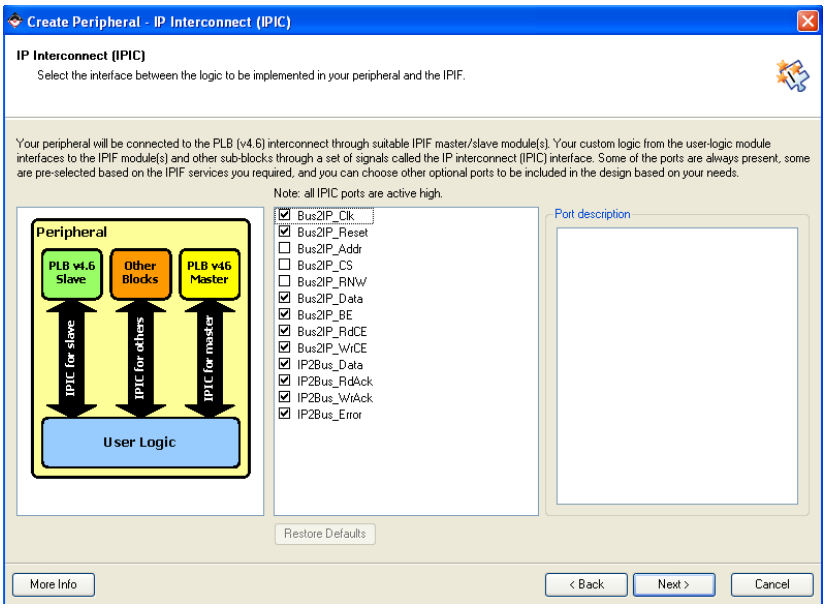

Figura 10. Interconexiones del IP Core

En la ventana de soporte de simulación del periférico, dejar la opción ***Generate BFM simulation Platform*** sin seleccionarla, clic en ***Next***.

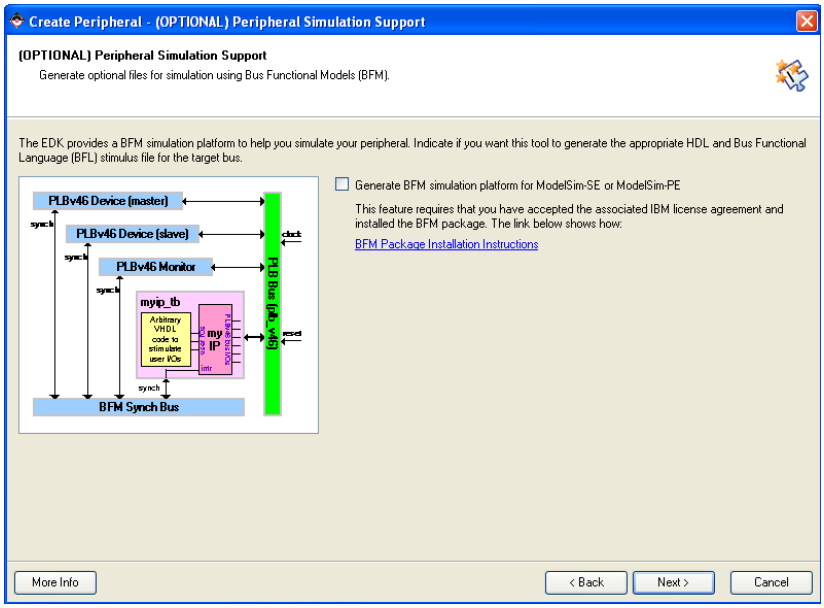

Figura 11. Soporte de simulación

Se presenta otra configuración opcional, el soporte de implementación, con la cual se crean diferentes tipos de archivos como ayuda al usuario; por defecto se selecciona únicamente la opción ***Generate template driver files to help you implement software interface***, clic en ***Next*** para continuar.

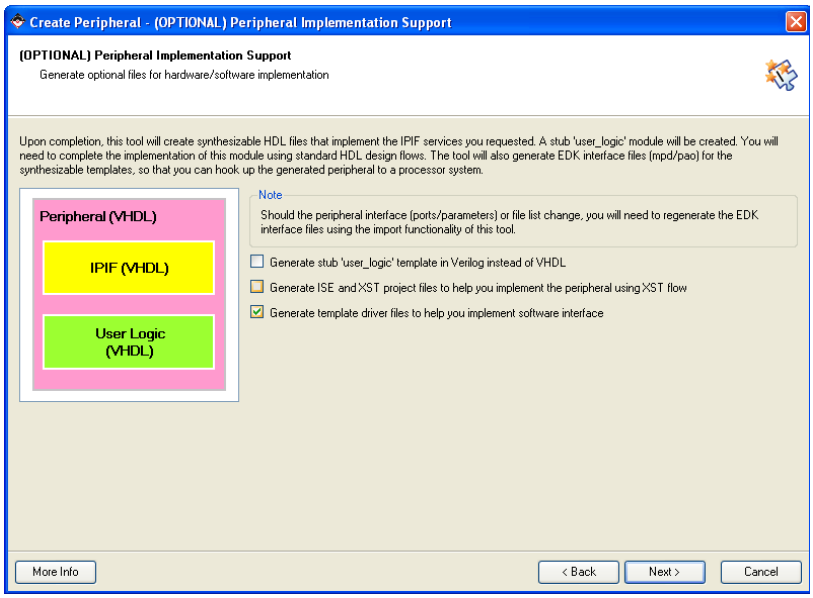

Figura 12. Soporte de implementación

Seguido aparece la ventana de resumen con la información del periférico generado, en ella aparecen sus principales características así como los diferentes archivos a generar y su respectiva dirección. Clic en ***Finish*** para finalizar el Wizard y comenzar la generación.

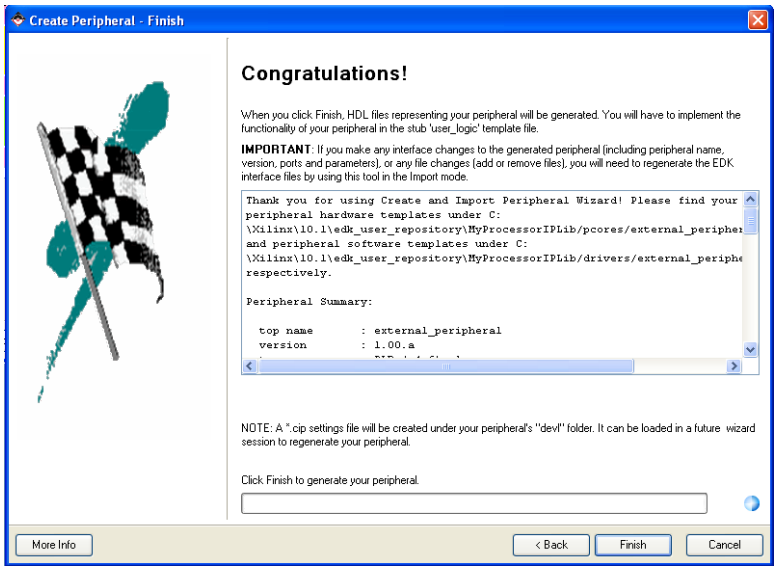

Figura 13. Información de resumen

Para asegurarse que la creación del periférico es correcta, diríjase a la pestaña ***IP Catalog*** del ***Project information Area*** de XPS, en la cual debe aparecer el nuevo periférico creado como se muestra:

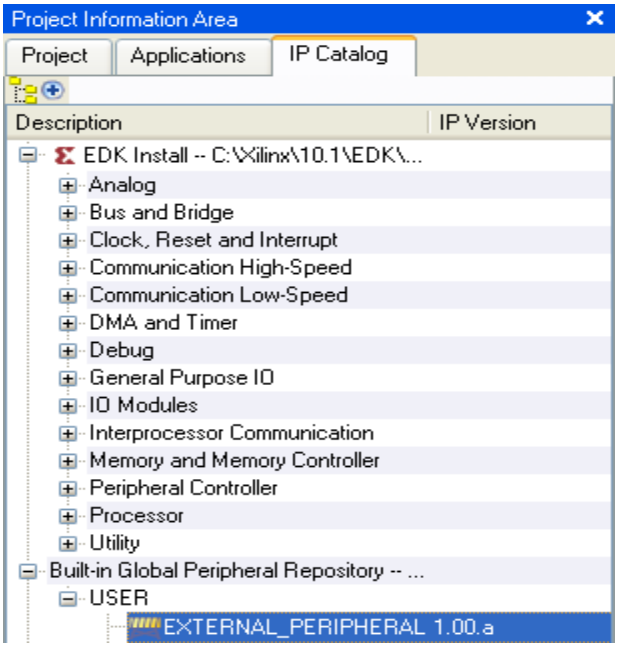

Figura 14. Pestaña ***IP Catalog*** actualizada

A continuación es necesaria la modificación de algunos de los archivos generados, esto con el fin de declarar las respectivas entradas-salidas del periférico, así como el código HDL que describe su funcionamiento.

Estos archivos están guardados en dos carpetas (drivers y pcores), por defecto estas carpetas se encuentra en la dirección: "**C:\Xilinx\11.1\edk_user_repository\MyProcessorIPLib**", la carpeta drivers consta de los archivos fuente para las aplicaciones software (archivos *.C y *.H) y la carpeta pcores consta de la programación hardware del periférico (archivos *.MPD y *.VHD); dentro de estas carpetas aparece una nueva carpeta por cada periférico creado, identificándose con el nombre del periférico usado en el Wizard. Estos archivos pueden ser abiertos con cualquier programa de edición de texto o pueden ser llamados directamente desde el XPS para realizar cualquier modificación necesaria. A continuación se muestra el árbol de carpetas completo para el periférico creado.

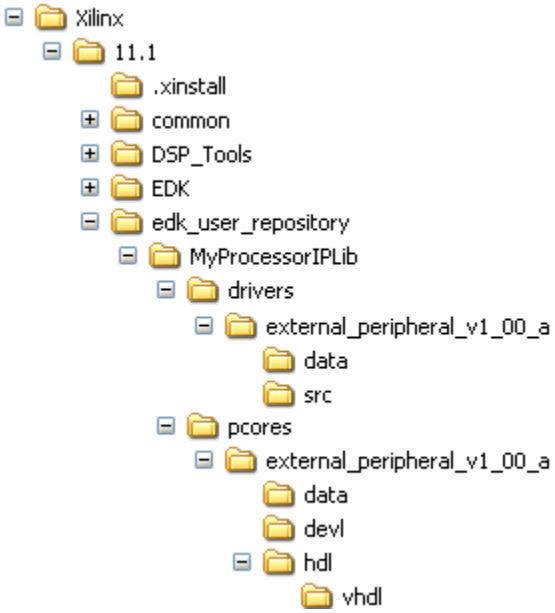

Figura 15. Carpetas Generadas del Periférico

La inclusión del periférico es necesaria para la generación del Bitstream, por esto es indispensable aplicar las modificaciones de hardware necesarias. Como primera medida incluimos los puertos del periférico para que este pueda ser conectado desde XPS. Para lograr esto se debe modificar el archivo MPD localizado en la subcarpeta data (**…pcores\external_peripheral_v1_00_a\data**). Con el fin de generalizar se agregaran una entrada y una salida para el periférico, para esto dentro del archivo **external_peripheral_v2_1_0.mpd** añadimos las siguientes líneas dentro de la sección **Ports**:

```
PORT entrada = "", DIR = I, VEC = [0:X]
PORT salida = "", DIR = O, VEC = [0:X]
```

El tamaño de los puertos ([0:X]) se define dentro de los corchetes cuadrados dependiendo de la necesidad del proyecto, luego de definir esto se procede a guardar y cerrar el archivo.

Ahora se debe crear los controladores del periférico así como el código en HDL que describe su funcionamiento. Para empezar se abre el archivo **external_peripheral.vhd** de la carpeta **…pcores\external_peripheral_v1_00_a\hdl\vhdl**, y se añaden los puertos deseados debajo de la línea de comentario --**USER ports added here** de la siguiente forma:

```
entrada          : in    std_logic_vector (0 to X);
salida           : out std_logic_vector (0 to X);
```

Al realizar esto el archivo debe verse de la forma:

```
-- DO NOT EDIT ABOVE THIS LINE ----------------------
  );
  port
  (
    -- ADD USER PORTS BELOW THIS LINE ------------------
    --USER ports added here
    entrada                    : in  std_logic_vector(0 to X);
    salida                     : out std_logic_vector(0 to X);
    -- ADD USER PORTS ABOVE THIS LINE ------------------

    -- DO NOT EDIT BELOW THIS LINE ----------------------
    -- Bus protocol ports, do not add to or delete
    SPLB_Clk                   : in  std_logic;
    SPLB_Rst                   : in  std_logic;
```

Dentro del mismo archivo es necesario crear el mapeo de los puertos, para esto buscamos la línea de comentario **--USER ports mapped here** y debajo de esta añadimos las siguientes líneas:

```
entrada   => entrada,
salida      => salida,
```

El archivo debe verse de la siguiente manera:

```
USER_LOGIC_I : entity external_peripheral_v1_00_a.user_logic
    generic map
    (
      -- MAP USER GENERICS BELOW THIS LINE ----------------
```

```
  --USER generics mapped here
  -- MAP USER GENERICS ABOVE THIS LINE ----------------

  C_SLV_DWIDTH                => USER_SLV_DWIDTH,
  C_NUM_REG                   => USER_NUM_REG
)
port map
(
  -- MAP USER PORTS BELOW THIS LINE ------------------
  --USER ports mapped here
  entrada                     => entrada,
  salida                      => salida,
  -- MAP USER PORTS ABOVE THIS LINE ------------------

  Bus2IP_Clk                  => ipif_Bus2IP_Clk,
  Bus2IP_Reset                => ipif_Bus2IP_Reset,
```

Luego de realizar estos cambios se guarda y cierra el archivo **external_peripheral.vhd**.

Dentro de la misma ubicación (**…pcores\external_peripheral_v1_00_a\hdl\vhdl**) se encuentra el archivo **user_logic.vhd** en el cual definimos nuevamente los puertos añadidos. En la sección USER PORTS y debajo de la línea de comentario --**USER ports added here** incluimos nuevamente la declaración de los puertos como en al anterior archivo.

```
entrada          : in    std_logic_vector (0 to X);
salida           : out std_logic_vector (0 to X);
```

La sección del archivo queda de la siguiente manera:

```
  -- DO NOT EDIT BELOW THIS LINE ----------------------
  -- Bus protocol parameters, do not add to or delete
  C_SLV_DWIDTH                 : integer              := 32;
  C_NUM_REG                    : integer              := 1
  -- DO NOT EDIT ABOVE THIS LINE ----------------------
);
port
(
  -- ADD USER PORTS BELOW THIS LINE ------------------
  --USER ports added here
  entrada                      : in  std_logic_vector(0 to 3);
  salida                       : out std_logic_vector(0 to 7);
  -- ADD USER PORTS ABOVE THIS LINE ------------------

  -- DO NOT EDIT BELOW THIS LINE ----------------------
  -- Bus protocol ports, do not add to or delete
  Bus2IP_Clk                   : in  std_logic;
  Bus2IP_Reset                 : in  std_logic;
```

Cabe la anotación que cualquier señal que el usuario quiera incluir al periférico debe añadirla en este archivo debajo de la línea de comentario **--USER signal declarations added here, as needed for user logic**, siguiendo los lineamientos del lenguaje utilizado.

Para finalizar falta agregar a este archivo el código a implementar que controla el periférico. Este código se copia debajo de la línea de comentario **--USER logic implementation added here**. Luego de añadido el código se debe guardar y cerrar el archivo **user_logic.vhd**.

De vuelta en XPS, damos clic en el menú ***Project*** y se selecciona ***Rescan User Repositories*** para actualizar los contenidos anteriormente cambiados. Para asegurarse de que los cambios se tomen por completo se recomienda cerrar XPS y reabrirlo nuevamente. En este punto el periférico puede ser usado como cualquier otro anteriormente (GPIO, etc.).

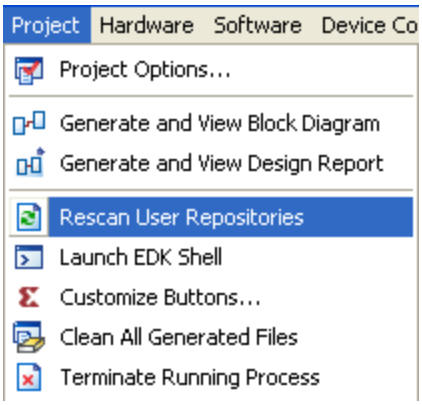

Figura 16. Actualización de contenidos

Siguiendo el flujo de diseño del XPS, luego de agregar los periféricos a usar para el proyecto, realizar conexiones, hacer externos los puertos necesarios, mapeo de memoria, añadir restricciones (.ucf) y realizar el Bitstream; se continúa con la creación de la aplicación software del proyecto. En este punto es ineludible que el usuario tenga conocimiento de los archivos **external_peripheral.H**(Header file) y **external_peripheral.C**(Source file) de la carpeta **(…drivers\external_peripheral_v1_00_a\src**), en el archivo .H se encuentran las definiciones de las funciones genéricas desarrolladas por el Wizard:

Escritura de dato al registro del periférico:

```
#define EXTERNAL_PERIPHERAL_mWriteReg(BaseAddress, RegOffset, Data) \
        XIo_Out32((BaseAddress) + (RegOffset), (Xuint32)(Data))
```

Lectura de dato del registro del periférico:

```
#define EXTERNAL_PERIPHERAL_mReadReg(BaseAddress, RegOffset) \
        XIo_In32((BaseAddress) + (RegOffset))
```

Escritura de dato al registro esclavo (0) del periférico:

```
#define EXTERNAL_PERIPHERAL_mWriteSlaveReg0(BaseAddress, RegOffset,
Value) \
        XIo_Out32((BaseAddress) +
(EXTERNAL_PERIPHERAL_SLV_REG0_OFFSET) + (RegOffset),
(Xuint32)(Value))
```

Lectura de dato al registro esclavo (0) del periférico:

```
#define EXTERNAL_PERIPHERAL_mReadSlaveReg0(BaseAddress, RegOffset) \
        XIo_In32((BaseAddress)                                    +
(EXTERNAL_PERIPHERAL_SLV_REG0_OFFSET) + (RegOffset))
```

Un ejemplo del uso de una de estas funciones en código C es:

```
EXTERNAL_PERIPHERAL_mWriteReg(0xC4600000, 0, 0xFF);
```

En esta línea se escribe en el registro (0) del periférico el dato FF hexadecimal, la dirección de memoria base está determinada por el mapeo de memoria realizado para el periférico antes del Bitstream.

Los archivos anteriormente mencionados son modificables desde un procesador de texto o desde el XPS según necesidad del usuario. Se recuerda que las funciones del archivo fuente de la aplicación software para operar el periférico

deben ser llamadas agregando el header del periférico, usando la línea de código: #include " external_peripheral.h ", al comienzo del archivo fuente.

Referencia

- http://www.xilinx.com/support/documentation/application_notes/xapp967.pdf

Anexo I

XILKERNEL [1]

El propósito de este documento es dar a conocer las principales características y funciones del microkernel provisto poro Xilinx para el desarrollo de aplicaciones sobre sus plataformas de desarrollo usando los procesadores soft-core soportados.

Además se muestra los requerimientos especiales (timer e interrupción) y la configuración software para el funcionamiento del SO.

Descripción

Xilkernel son un conjunto de librerías software gratuitas que se distribuyen con el Embedded Development Kit (EDK) y está diseñado para el trabajo en el Xilinx Plattform Studio (XPS); estructuralmente es un kernel modular, robusto y pequeño, diseñado para el trabajo desde las plataformas de Xilinx, apoyado por un procesador soft-core como MicroBlaze o PowerPC.

A continuación se muestran algunas razones para usar un kernel:

- En los sistemas de control o sistemas complejos se necesita operar muchas tareas o procesos de acuerdo a un orden establecido. A medida que el numero de tareas se incrementa, la dificultad para organizar e identificar manualmente aumenta, perjudicando el desarrollo de la aplicación.
- Con la ayuda de un kernel es posible escribir las aplicaciones en lenguajes de alto nivel.
- La configuración y despliegue del Xilkernel es fácil y completamente hecha desde XPS.
- La POSIX[29] API[30] permite una serie de funciones como: Thread's (hilos de ejecución), Sistemas de sincronización (Semáforos, Bloqueos, etc), Configuración de memoria dinámica, Temporizadores por software, Interrupciones a nivel de usuario, etc.
- Creación de thread's estáticos al iniciar Xilkernel.
- Protección de memoria por medio del gestor de memoria provisto por el MicroBlaze.

Xilkernel está compuesto por distintos módulos, estos pueden ser modificados de acuerdo a las necesidades del usuario. En la siguiente figura se muestra un diagrama de los módulos que componen el kernel.

[29] Portable Operating System Interface, uniX
[30] Application Programming Interface

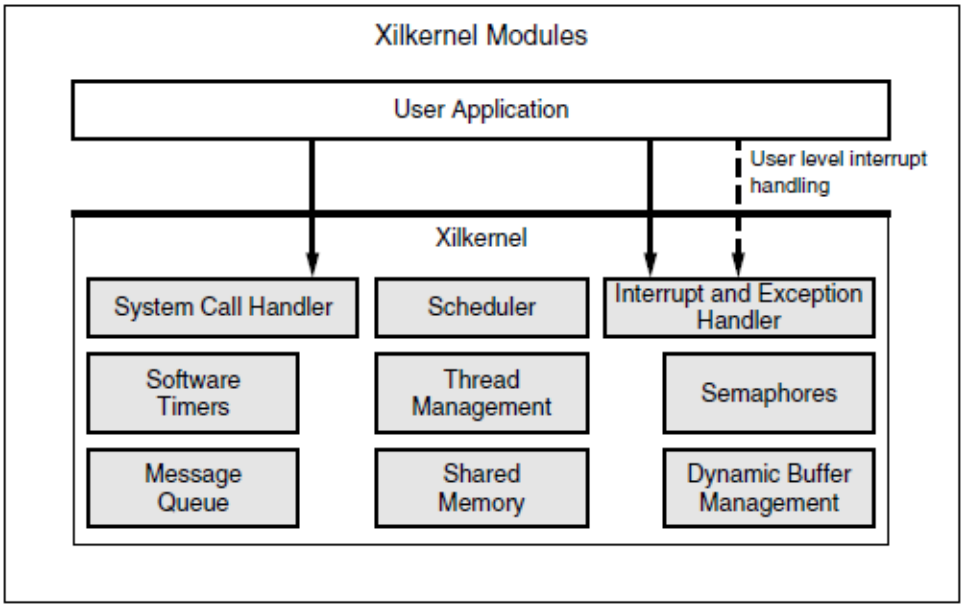

Figura 1. Módulos del Xilkernel [1]

Funciones

La POSIX es un conjunto de estándares definidos por la IEEE[31], que reúnen distintas conjunto de interfaces de aplicación adaptables a una gran variedad de implementaciones de sistemas operativos. Estos surgieron como el resultado de la normalización de las API's, con el fin de que fueran usados en diferentes plataformas [2].

La ejecución del Xilkernel es casi equivalente a cualquier SO comercial, lo que hace más fácil el desarrollo de aplicaciones, portabilidad, etc. A continuación se da una breve descripción de las funciones con las que cuenta el Xilkernel, acotando que no todos los conceptos e interfaces contenidos en los estándares POSIX están disponibles.

[31] Institute of Electrical and Electronics Engineers

- Manejo de Hilos (Thread Management); la creación y manipulación de thread's se basa en la notación estándar de la POSIX. Los thread's son identificados por el comando *pthread_t*, este identificador diferencia un thread de un proceso.

  Los thread's creados son almacenados en una memoria especifica definida por la configuración del número máximo de thread's, además es posible asignar memoria manualmente para la creación de thread's dinámicos. El modulo de thread's como todos es opcional y es configurado por el usuario, en el apartado de configuración se muestra la forma de realizar esta operación.

- Semáforos (Semaphores); Xilkernel soporta los semáforos declarados dentro de POSIX para sincronización, además soporta algunas interfaces para trabajar con semáforos con nombres. El numero de semáforos y su longitud se determina desde la configuración inicial.

- Mensajes en Espera (Message Queues); estos mensajes pueden ser usados dentro de un mecanismo IPC (Inter-process communication), que sirve para intercambiar datos entre thread's en uno o más procesos.

  Los mensajes pueden tomas cualquier tamaño pero este depende del buffer de memoria establecido en la inicialización de la memoria. El número de mensajes y el tamaño de los mismos puede ser configurado desde la configuración inicial. Este modulo requiere la adición de los módulos de semáforos y asignación dinámica de memoria.

- Memoria Compartida (Shared Memory); otro mecanismo IPC, de baja latencia y muy común. Los bloques de memoria compartida requeridos durante la ejecución del programa deben ser identificados en la configuración del sistema ya que esta no se asigna dinámicamente durante

la ejecución. Este modulo como los anteriores es opcional y se configura inicialmente.

- Bloqueos de Exclusión Mutua (Mutex Locks); sistema que se usa básicamente para evitar que diferentes thread's usen los mismos recursos al mismo tiempo, el numero de bloqueos y su longitud es configurable desde el inicio.

- Manejo de memoria dinámica (Dynamic Buffer Memory Management); Xilkernel provee un buffer de memoria el cual puede ser usado por las aplicaciones que lo requieran. El buffer de memoria maneja el bloque total de memoria, además el usuario puede crear nuevos bloques de memoria dinámicamente o definir estáticamente el número de bloques y sus tamaños inicialmente.

- Temporizadores (Software Timers); para procesos relacionados con funciones de tiempo el Xilkernel provee temporizadores. Este modulo puede ajustarse en la configuración inicial.

- Interrupciones a Nivel de Usuario (User-Level Interrup Handling API).

La lista detallada de las funciones para cada uno de los módulos descritos anteriormente se encuentra en el documento de la referencia [1], en el se muestra la nomenclatura en código C y los parámetros necesarios para su uso.

Requerimientos (Hardware - XPS)

Luego de crear el proyecto en el XPS, y agregar los periféricos necesarios es indispensable, antes de generar el bitstream (*.bit), agregar otros IP Cores inevitables para el funcionamiento del Xilkernel. En primer lugar se necesita un

timer que interrumpa periódicamente el MicroBlaze, para esto agregamos el periférico *XPS_Timer* disponible en la pestaña *IP Catalog* del XPS, en la sección *DMA and Timer.*

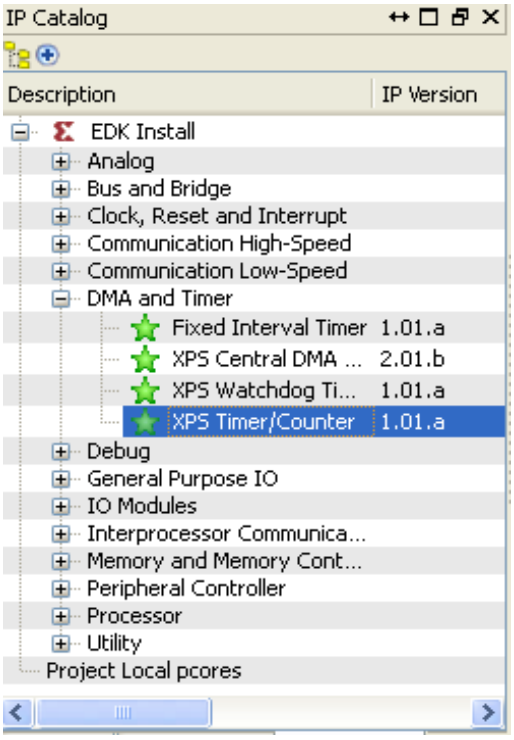

Figura 2. Periférico Timer

Como segunda medida es necesario agregar un periférico para que controle estas interrupciones, esto lo hace el periférico *XPS Interrutp Controller*, que se encuentra en la misma pestaña dentro de la sección *Clock, Reset and Interrupt*.

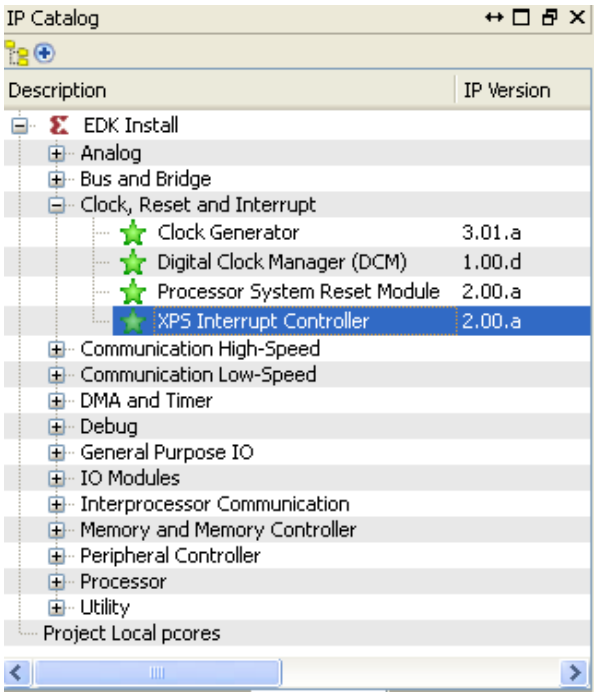

Figura 3. Periférico Controlador de Interrupciones

Luego de agregar los periféricos mencionados se agregan al PLB en la pestaña Bus Interface y se generan las posiciones de memoria en Addresses, a continuación es necesario configurar los puertos de los periféricos añadidos, para esto expandimos el controlador de interrupciones (*xps_Intc_0*) y en el puerto *Intr* añadimos la interrupción del Timer.

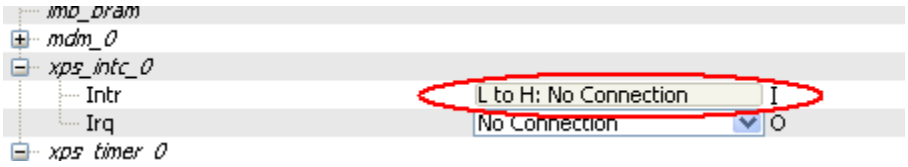

Figura 4. Configuración del Controlador de Interrupciones

Al dar clic sobre la región mostrada en la figura 4 emerge una ventana como la siguiente en donde se escoge el Timer.

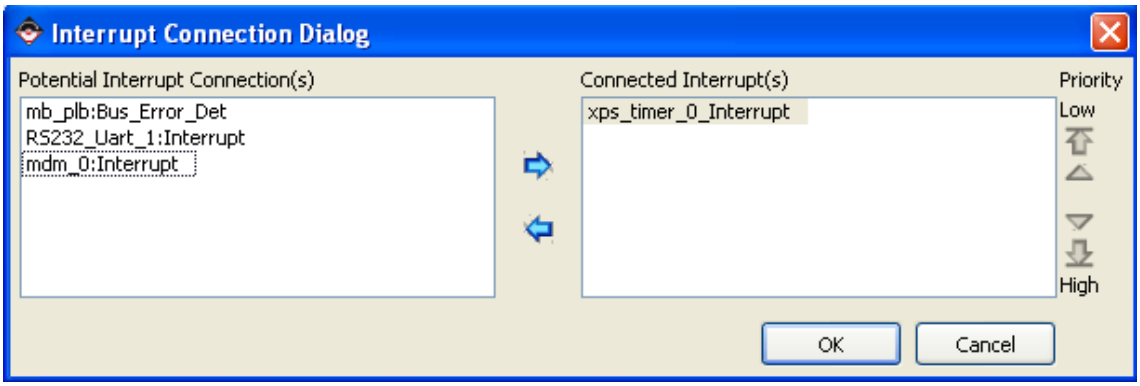

Figura 5. Adición del Timer como Interrupción

Seguido se configura el periférico *xps_timer_0*, agregando al puerto *Interrupt* la opción *xps_timer_0_Interrupt*.

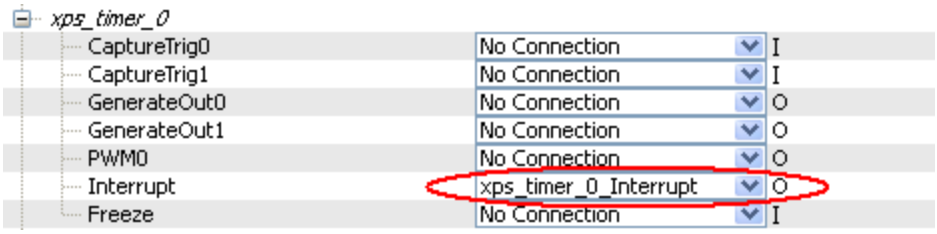

Figura 6. Conexión del Puerto Interrupt

A continuación se realiza la misma acción para el procesador, agregando *xps_timer_0_Interrupt* en el puerto INTERRUPT del MicroBlaze.

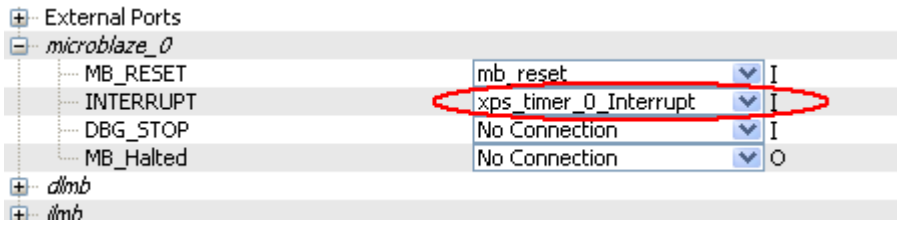

Figura 7. Conexión de la Interrupción del Procesador

Al terminar estas configuraciones la pestaña de *Ports* debe verse de la siguiente forma.

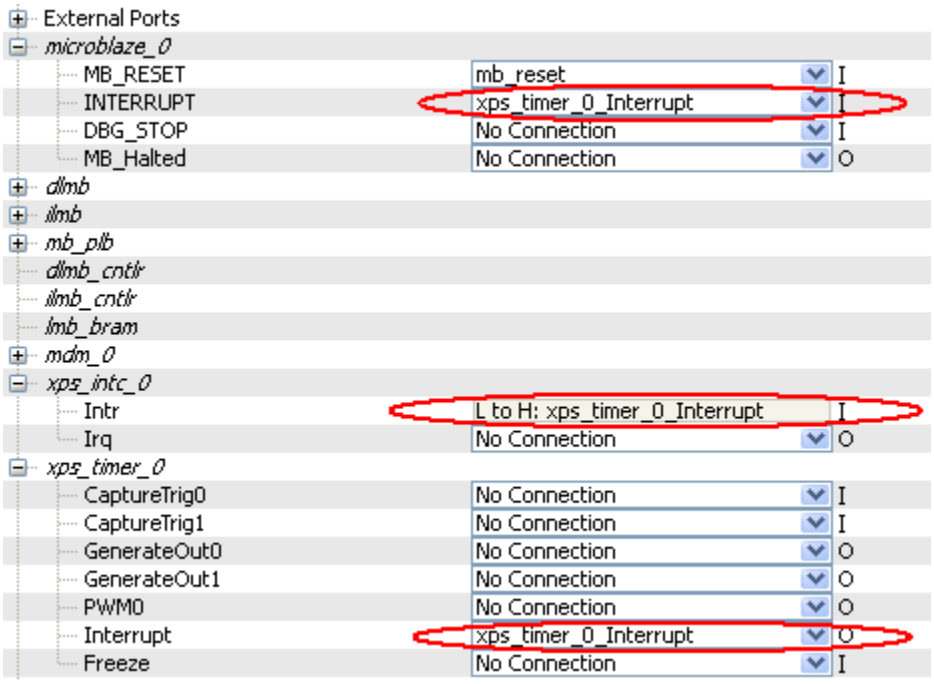

Figura 8. Configuración Final de la Interrupción

Luego de estos complementos se procede a generar el bitstream del proyecto y seguido a la configuración software.

Configuración Software

Es aquí donde se incluye la utilización del Xilkernel, así como donde se configura el uso de los módulos anteriormente descritos.

Iniciamos siguiendo el flujo de diseño utilizado en el anexo “Introducción al XSP” para una aplicación software, la pestaña *Applicactions* se vera de la forma:

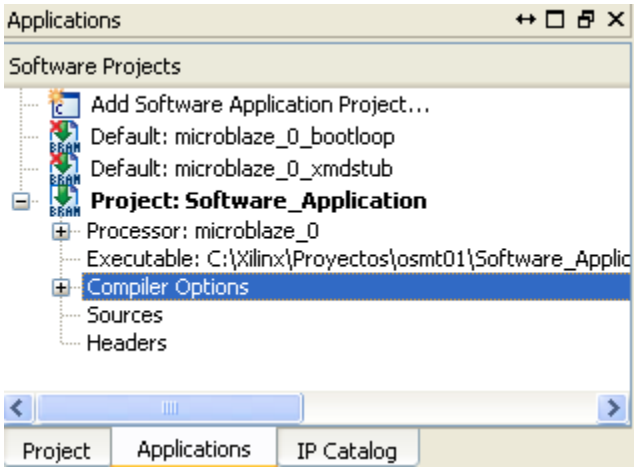

Figura 9. Opciones del Compilador de Aplicación

Al dar doble clic sobre la opción Compiler Options de la aplicación creada, es necesario agregar la librería -lxilkernel en la casilla *Libraries to Link against (-l)* de la pestaña *Paths and Options* de la ventana emergente.

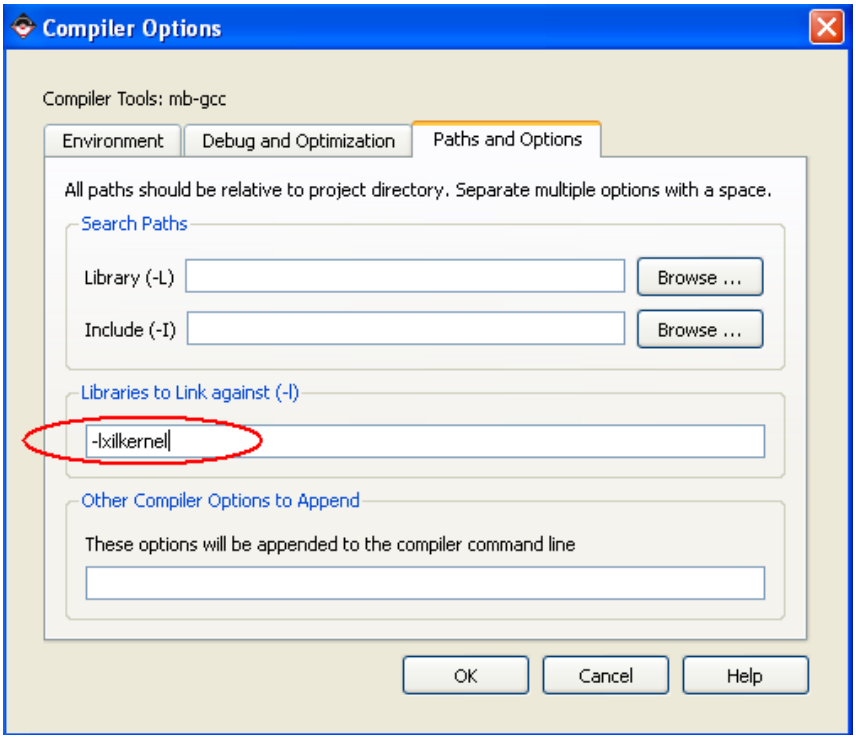

Figura 10. Inclusión de Librerías del Xilkernel

Con clic en OK aceptamos los cambios y se continúa a realizar la configuración inicial y de los módulos del Xilkernel. Para esto accedemos a la opción *Software Platform Settings* del menú *Software*.

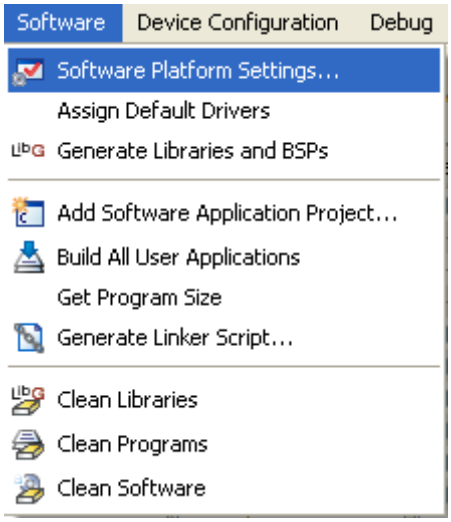

Figura 11. Acceso a la Configuración Software

Luego de aceptar la recomendación que muestra el software acerca de usar el SDK (Software Development Kit) para estas configuraciones, aparece una ventana como se muestra en la siguiente figura. En la parte superior de la ventana de configuración software aparece el procesador con el que se está trabajando, en la parte derecha aparecen las tres opciones de configuración: *Software Platform*, *OS and Lib Configuration* y *Drivers*.

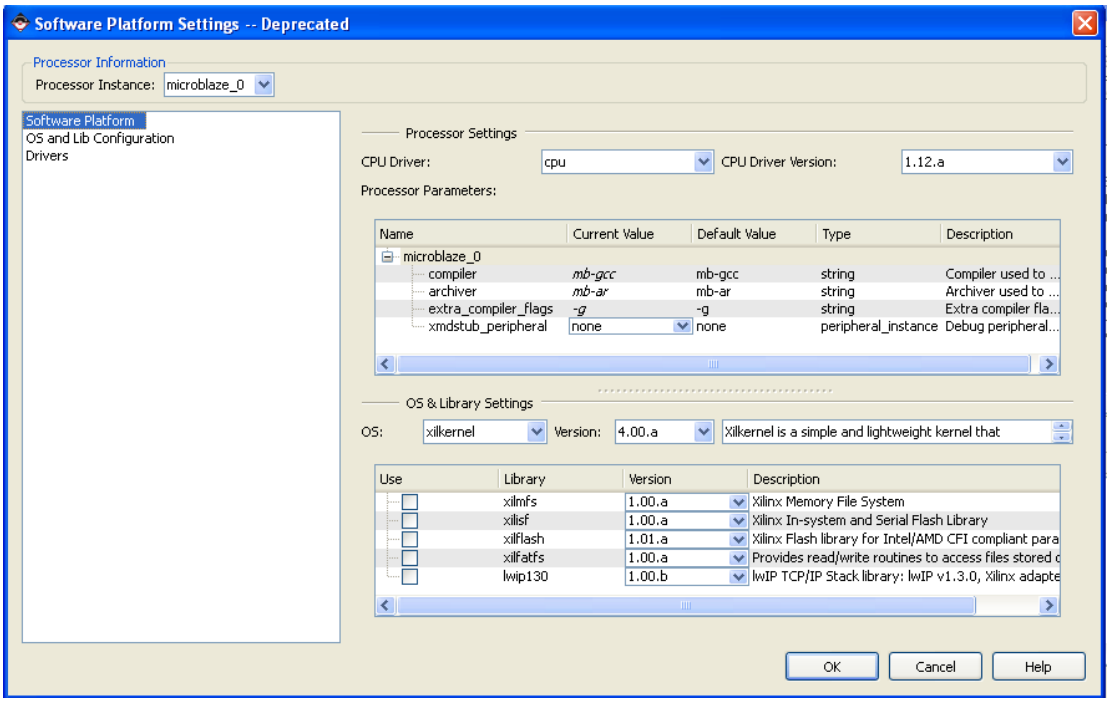

Figura 12. Ventana de Configuración software

En la opción *Software Platform* está la configuración de CPU (Processor Settings) que se deja por defecto o el usuario puede modificarla según su necesidad; y la configuración del SO y librerías (OS & Library Settings), es acá, en la opción *OS:* donde seleccionamos el sistema operativo a utilizar, en este caso Xilkernel, además se encuentra la posibilidad de utilizar diferentes librerías especificas para agregar funciones al kernel.

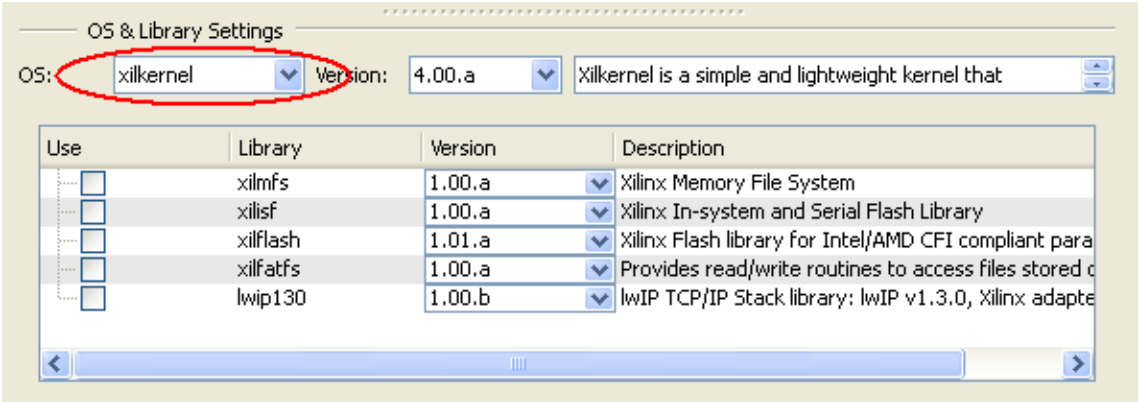

Figura 13. Configuración del SO y Librerías

La opción *OS and Lib Configuration* es la encargada de la configuración de los módulos del kernel y de la configuración inicial del mismo, la ventana al expandir *xilkernel* se muestra así:

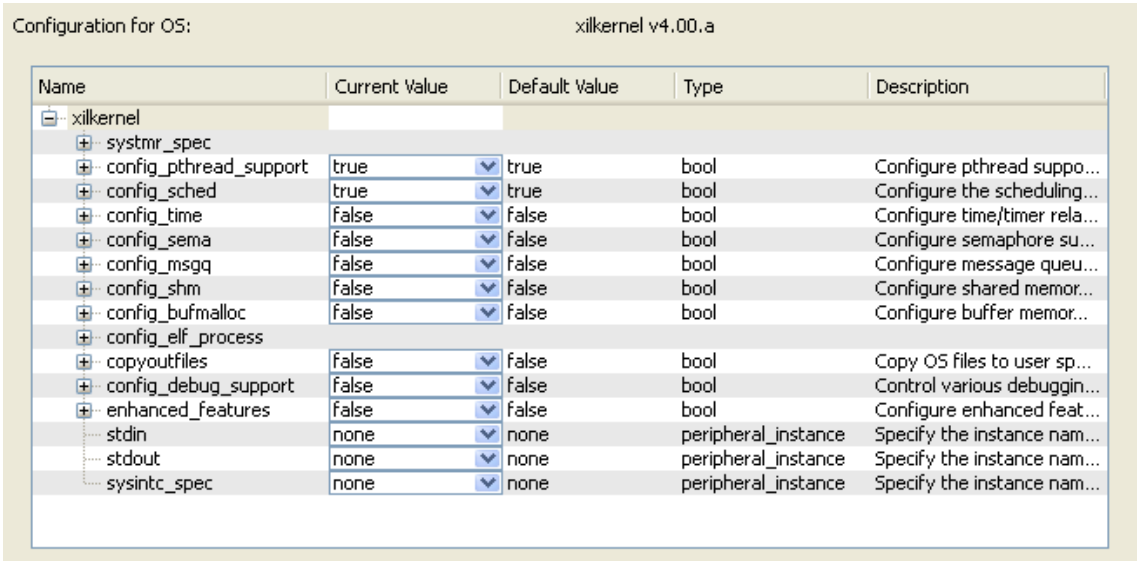

Figura 14. Configuración del SO

Esta es la configuración que ofrece el software por defecto, utilizando el mudulo de thread's estaticos (*config_pthread_support*) y el de programación del kernel (*config_sched*), los módulos activos se marcan con la palabra *true* como se observa en la figura 14. Para una configuración básica es necesario agregar el timer, la interrupción y la entrada y salida estándar.

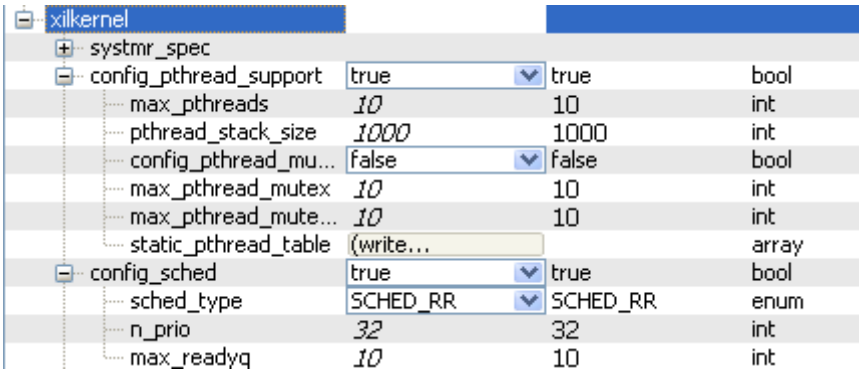

Figura 15. Configuración de Thread's y Programación

Es importante mencionar que la declaración de thread's estáticos se realiza desde la configuración *static_phtread_table*, dando clic sobre *(write…* se puede definir el numero de thread's, que no debe superara al parámetro *max_pthreads*, a usar y el nombre de las funciones que cada uno llama al ejecutarse dentro del código en C de la aplicación.

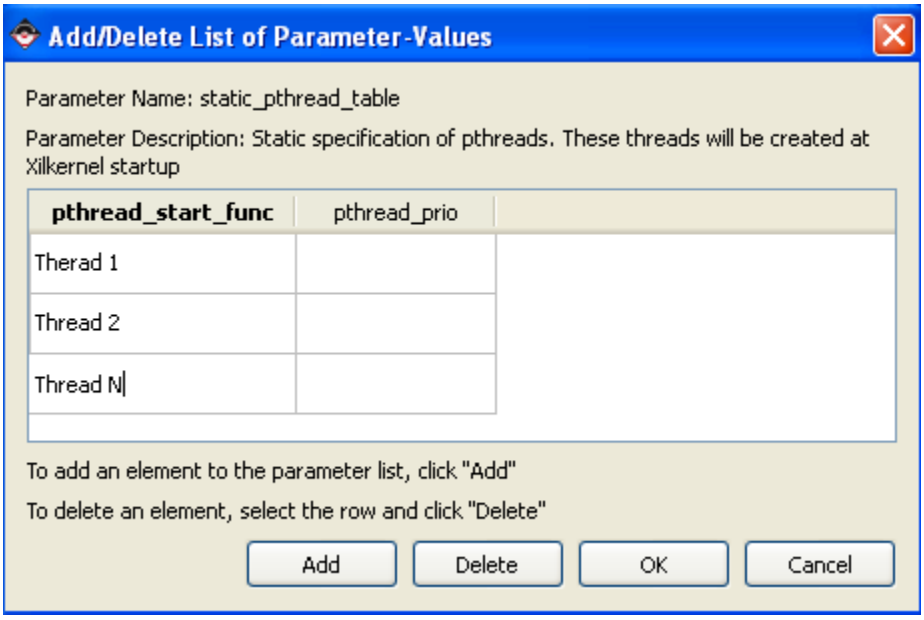

Figura 16. Tabla de Thread's Estáticos

En la configuración systmr_spec se escoge, en el parámetro *systmr_dev,* el periférico de timer añadido anteriormente (xps_timer_0) y se define la frecuencia de trabajo del sistema, para este caso se uso 62.5 MHz en el parámetro *systmr_freq*. El parámetro *systmr_interval* se refiere al tiempo de intervalo entre ejecuciones del kernel, este es de suma importancia para funciones como sincronización y el manejo de thread's (Multithreading).

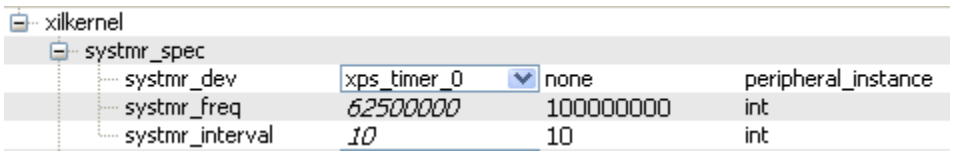

Figura 17. Configuración de Timer y Frecuencia

La configuración de la interrupción y los puertos estándares, se realiza en la parte inferior de las configuraciones del *xilkernel*. Los puertos (*stdin* y *stdout*) son los usados por el Xilkernel como estándar y deben ser definidos según la necesidad del usuario dentro de los periféricos añadidos al sistema. En el parámetro sysintc_spec debe seleccionarse el controlador de interrupciones que se añadió (*xps_intc_0*), un ejemplo para estas configuraciones se muestra a continuación:

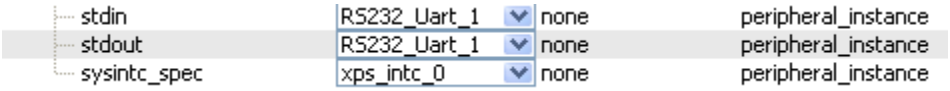

Figura 18. Configuración de interrupción y Puertos

Para el uso de cualquier otro modulo del Xilkernel debe activarse desde estas configuración y especificar los parámetros internos necesarios como se mostro para los thread's. Para finalizar en la opción Drivers se muestra un resumen sencillo de los periféricos utilizados en el sistema.

Programación

Luego de realizar las configuraciones necesarias para el Xilkernel, solo falta realizar el código en lenguaje C para la aplicación. Tenga en cuenta estas recomendaciones para su elaboración:

- Toda aplicación que use el Xilkernel debe incluir el encabezado xmk.h (#include "xmk.h")
- El uso de algunos módulos y funciones de estos requieren de encabezados adicionales, refiérase al documento proporcionado en la referencia [1]

- El sistema puede configurarse de dos formas de operación, que inicie el SO y permanezca operando según sus módulos y procesos o que se usen los módulos y funciones para completar un flujo de procesos con un fin.
- Para que el sistema permanezca operando debe incluir el comando xilkernel_main () dentro de la función main() del código. Cualquier inicialización necesaria debe realizarse antes de invocar al kernel.

Como ejemplo se muestra un código que imprime desde el SO un mensaje, esto usando las configuraciones anteriormente realizadas:

```
#include "xmk.h"
#include "xparameters.h"
#include "xutil.h"
#include "xuartlite.h"
#include <stdio.h>
#include <pthread.h>

extern "C" {
void Thread_1() {
print("Mensaje desde Thread 1 \r\n");
        }
}

int main () {
XUartLite instance;
XUartLite_Initialize(&instance, XPAR_RS232_UART_1_DEVICE_ID);
```

```
print("\033[H\033[J");
print("Entering xilkernel_main()\r\n");
xilkernel_main();
print("No se llega aca \r\n");
}
```

Referencias

[1] http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

[2] http://es.wikipedia.org/wiki/POSIX