Implementación de los periféricos en VHDL: Controlador VGA, Teclado Ps2 y Lector de Código de Barras PS2; para ser incorporados al microprocesador empotrado MicroBlaze, usando el sistema de desarrollo Spartan 3A DSP de Xilinx.

Juan Guillermo Ortiz Cala
Jorge Andrés Pachón Cañas

UNIVERSIDAD INDUSTRIAL DE SANTANDER FACULTAD DE INGENIERÍAS FISICOMECÁNICAS ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y TELECOMUNICACIONES BUCARAMANGA
2010

Implementación de los periféricos en VHDL: Controlador VGA, Teclado Ps2 y Lector de Código de Barras PS2; para ser incorporados al microprocesador empotrado MicroBlaze, usando el sistema de desarrollo Spartan 3A DSP de Xilinx.

Juan Guillermo Ortiz Cala
Jorge Andrés Pachón Cañas

TITULO A OBTENER:
INGENIERO ELECTRÓNICO

DIRECTOR:
MIE (c) Carlos Augusto Fajardo Ariza

UNIVERSIDAD INDUSTRIAL DE SANTANDER FACULTAD DE INGENIERÍAS FISICOMECÁNICAS ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y TELECOMUNICACIONES BUCARAMANGA
2010

Tabla de contenido

Lista de Figuras

Lista de Tablas

RESUMEN

TÍTULO: IMPLEMENTACIÓN DE LOS PERIFÉRICOS EN VHDL: CONTROLADOR VGA, TECLADO PS2 Y LECTOR DE CÓDIGO DE BARRAS PS2; PARA SER INCORPORADOS AL MICROPROCESADOR EMPOTRADO MICROBLAZE, USANDO EL SISTEMA DE DESARROLLO SPARTAN 3A DSP DE XILINX.[*]

AUTORES: ORTIZ CALA Juan Guillermo, PACHÓN CAÑAS Jorge Andrés.[**]

PALABRAS CLAVE: Embedded Systems (ES), MICROBLAZE, FPGA, VHDL, EDK.

RESUMEN:

En este artículo se describe el proceso de diseño de un ES "Verificador de Precios", el cual se ha implementado en el sistema de desarrollo Spartan 3A DSP de la empresa Xilinx. El enfoque de diseño seleccionado es el de un procesador IP implementado sobre una FPGA, al cual se le añaden módulos descritos en VHDL que controlan diferentes periféricos utilizando la metodología de codiseño.

La herramienta empleada para el desarrollo del ES es el EDK, donde se encuentra la Plataforma de Estudio de Xilinx (XPS), que da la posibilidad de diseñar un sistema de procesamiento empotrado
La arquitectura general del ES contiene: el procesador *soft-core MicroBlaze* proporcionado también por la empresa Xilinx. Donde se implementa la parte de software del diseño en el cual se define el proceso de visualización de los códigos en el VGA y adicionalmente contiene tres módulos de Hardware descritos en VHDL donde se describe el manejo de los periféricos (teclado, VGA, lector de código barras).

Para cada uno de los módulos implementados se describe: El diseño del HW, el proceso de adición al procesador Softcore MicroBlaze y test de funcionalidad o SW que aprueba el funcionamiento del módulo incorporado.
Por último se realiza la integración de los módulos IP para crear el ES "Verificador de precios".

Se concluye con la implementación del ES *Verificador de Precios* las posibilidades que ofrece el codiseño pues esta metodología permite aprovechar las ventajas tanto del Hardware como del Software en un solo diseño.

[*] Modalidad: Trabajo de grado.

[**]Facultad de Ingenierías Fisicomecánicas. Ingeniería Electrónica. Director MIE(c). Carlos Augusto Fajardo Ariza.

ABSTRACT

TITLE: IMPLEMENTATION OF THE PERIPHERAL IN VHDL: DRIVER VGA, PS2 KEYBOARD AND PS2 BARCODE SCANNER, TO BE BUILT IN THE MICROBLAZE™ EMBEDDED PROCESSOR, USING THE SYSTEM DEVELOPMENT SPARTAN 3A DSP OF XILINX.[*]

AUTHORS: ORTIZ CALA Juan Guillermo, PACHÓN CAÑAS Jorge Andrés.[**]

KEYWORDS: Embedded Systems (ES), MICROBLAZE, FPGA, VHDL, EDK.

ABSTRACT:

This article describes the process of designing an ES "Price Checker", which has been implemented in the development system Spartan 3A DSP Xilinx Company. The design approach selected is the IP processor implemented on an FPGA, which are added as described in VHDL modules that control various peripherals using co-design methodology.

The tool used for the development of ES is the EDK, where the Xilinx Platform Studio (XPS), which gives the possibility of designing an embedded processing system.
The overall architecture of the ES has: the MicroBlaze soft-core processor also provided by the company Xilinx. Where to deploy the software portion of the design which defines the process of displaying the codes on the VGA and additionally contains three modules described in VHDL hardware which describes the management of peripherals (keyboard, VGA, barcode scanner).

For each of the described modules implemented: HW design, the process of adding the MicroBlaze softcore processor and SW test of functionality or approving the operation of the built-in module.

Finally, we performed the integration of IP modules to create the ES "Price Checker."It concludes with the implementation of ES Price Checker possibilities offered by the co-design methodology allows for this to take advantage of hardware and software in a single design.

[*]Modality: Degree Work.

[**]Faculty of Physical-Mechanical Engineering. Electronic Engineering. Director MIE(c). Carlos Augusto Fajardo Ariza.

Implementación de los periféricos en VHDL: Controlador VGA, Teclado Ps2 y Lector de Código de Barras PS2; para ser incorporados al microprocesador empotrado MicroBlaze, usando el sistema de desarrollo Spartan 3A DSP de Xilinx. **(Agosto 2010)**

Ortiz Cala Juan Guillermo [1], Pachón Cañas Jorge Andrés [2], Fajardo Ariza Carlos Augusto [3]

[1]Estudiante de Ingeniería electrónica de la Universidad Industrial de Santander, Bucaramanga, COLOMBIA
jorca723@hotmail.com

[2]Estudiante de Ingeniería electrónica de la Universidad Industrial de Santander, Bucaramanga, COLOMBIA
jorgeap_07@hotmail.com

[3]Director del Proyecto, Universidad Industrial de Santander, Bucaramanga, COLOMBIA
cafa_78@yahoo.com

Resumen- En este artículo se describe el proceso de diseño de un "Verificador de Precios", el cual se ha implementado sobre el sistema de desarrollo Spartan 3A DSP de la empresa Xilinx. El diseño se ha realizado utilizando la metodología de codiseño, desarrollando en hardware el manejo de los periféricos (teclado, VGA, lector de código barras) y el proceso de visualización de los códigos en el VGA se ha diseñado en software utilizando el procesador soft-core MicroBlaze proporcionado también por la empresa Xilinx.

***Palabras Clave*— Embedded Systems (ES), MICROBLAZE, FPGA, VHDL, EDK.**

I. INTRODUCCIÓN

EN la actualidad, los sistemas empotrados[1] (ES) han logrado un gran auge gracias a sus amplios campos de aplicación y a su vez, por los bajos costos comparados con sistemas de cómputo tradicionales [1] haciéndose evidente en innumerables áreas tan diversas como el hogar, salud, energía, industria automotriz, telecomunicaciones, entretenimiento, automatización entre otros.

Debido a este incremento de la tecnología empotrada las grandes empresas de la informática (IBM, Apple, Hewlett Packard, Sun, Sony) están investigando sobre las necesidades específicas de los usuarios para satisfacerlas con los mejores productos y hoy en día han volcado sus esfuerzos en el diseño de toda una serie de sistemas empotrados [2].

Con la actual arremetida del mundo de la electrónica digital se hacen posible la implementación de nuevos sistemas basados en estrategia de diseño que integran Hardware (HW) y Software (SW) en un único circuito mediante la metodología de codiseño haciendo posible la aparición de sistemas complejos en un solo chip (SoC[2]).

Por otra parte el uso de FPGAs en Sistemas Empotrados está en aumento gracias a que ofrece una serie de ventajas en cuanto a: tiempos de desarrollo, facilidad de reconfiguración de hardware, y costos cuando el volumen de producción es bajo. Además ofrece el beneficio de reutilizar módulos prediseñados en otros proyectos, evitando que el diseñador tenga que volver a desarrollar y probar partes del sistema, razón por la cual las empresas dedicadas al diseño de ES desarrollan diseño de sistemas complejos [3]

El enfoque de diseño seleccionado es el de un procesador IP implementado sobre una FPGA, al cual se le añaden módulos descritos en VHDL que controlan diferentes periféricos.
Teniendo en cuenta el enfoque anterior, el ES que se propone es un verificador de precios donde el usuario tiene la posibilidad de realizar la consulta del valor del artículo ya sea por teclado o por Lector de código de barras. El usuario no solo dispondrá de la opción de consulta si no que también podrá llevar el valor de la cuenta de los artículos de su interés.

En el presente artículo se puede encontrar:

a. Descripción general del ES y la metodología de diseño implementada.
b. Breve resumen del entorno de desarrollo EDK (Embedded Development Kit) que permite el diseño de ES usando la metodología del codiseño.
c. Para cada uno de los módulos implementados se describe:
   - El diseño del HW
   - Adición al procesador Softcore MicroBlaze

[1] *U*n sistema basado en un procesador, diseñado para cumplir un rango específico de funciones
[2] System on chip

- Test de funcionalidad o SW que aprueba el funcionamiento del módulo incorporado.

d. Por último se realiza la integración de los módulos IP para crear el ES "Verificador de precios".

Este trabajo busca apoyar una investigación a nivel de maestría titulada Apropiación Tecnológica del Diseño de Embedded Systems implementados sobre FPGAs.

II. ARQUITECTURA GENERAL DEL PROYECTO

El *ES* que se expone en este artículo fue diseñado utilizando la metodología de codiseño, la cual busca complementar las ventajas que ofrece el hardware (mayor velocidad, posibilidad de realizar tareas de manera concurrente, menor consumo de potencia) como las del software (mayor flexibilidad, facilidad para ejecutar órdenes secuencialmente, menos tiempo en el diseño) incorporándolas en un solo ES [3].

El ES propuesto consiste en un verificador de precios donde se utiliza un interfaz VGA como medio de visualización, el lector de código de barras como método de consulta rápida y por último el teclado como medio opcional de consulta manual. El diseño se implementa en él FPGA 3SD1800A-FG676 [4] que se encuentra dentro del sistema de desarrollo SPARTAN 3A-DSP de la empresa Xilinx [4]. Utilizando el software de desarrollo EDK de Xilinx.

La arquitectura general del ES se muestra en la figura 1. El *ES* contiene el procesador *Softcore*[3] *MicroBlaze* en donde se implementará la parte de software del diseño, adicionalmente el *ES* posee tres módulos de hardware descritos en VHDL a manera de *IP cores*[4].

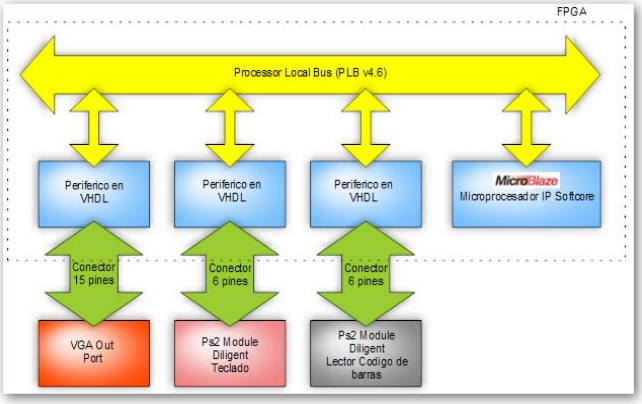


Figura 1 Arquitectura del ES

Módulos IP del Diseño:

a. *Controlador VGA (Video Graphics Array)*, o "colección de gráficos de video" es una norma de visualización de gráficos creado por IBM a fines de los años 80 para sus computadoras y procesadores, emplea una interfaz básica con resolución de 640x480 de 8 colores con señales de sincronismo horizontal y vertical para la conformación de campos. [5].

b. *Puerto Ps2*, protocolo introducido por las computadoras personales IBM, siendo una amplia interfaz utilizada para comunicar algunos dispositivos como (Teclado, Lector de código Barras, Mouse, entre otros) con un sistema digital en general (PC, microcontrolador, maquina de estado, etc) [5]. En el presente trabajo se utilizan dispositivos con interfaz unidireccional *Teclado y Lector código de barras* ambos conectados a un puerto PS2

III. ENTORNO DE DESARROLLO EDK (*EMBEDDED DEVELOPMENT KIT*).

Una herramienta que permite el desarrollo de *ES* utilizando la metodología de codiseño es el EDK, donde se encuentra la Plataforma de Estudio de Xilinx (XPS), que contiene una interfaz gráfica con un conjunto de herramientas que dan la posibilidad de diseñar un sistema de procesamiento empotrado basado en un procesador *softcore* como el *MicroBlaze* o *hardcore*[5] como el *Power PC* [6]. Esta herramienta adiciona al procesador módulos a manera de IP[6], integrando así el SW y HW en un mismo diseño [4]. El flujo de diseño para desarrollar un ES en la plataforma XPS se muestra la Figura 2 el cual contiene dos plataformas Hardware y software.

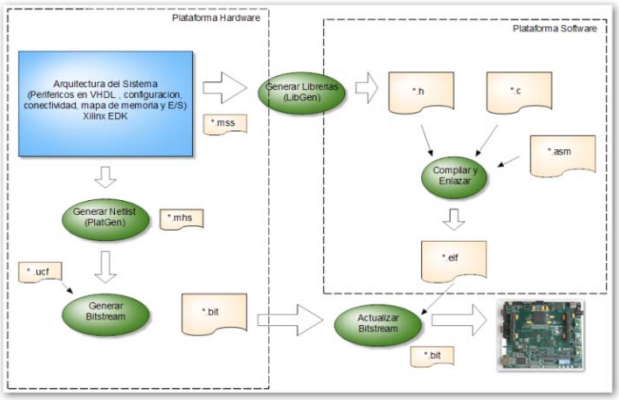


Figura 2 Flujo de diseño en XPS

En la plataforma hardware el diseñador define la arquitectura del sistema la cual incluye la configuración del microprocesador, los periféricos, la declaración de puertos entre otros. Una vez la arquitectura del sistema se encuentra

[3] *SoftCore:* Unas de las categorías de los IP cores siendo un modelo especifico descrito en Nelist o VHDL especifico para ser implementados en PLD´s [7]

[4] *IP Cores:* Bloques básicos con propiedad intelectual para usar en FPGA's o ASIC que describen la funcionalidad sin su parte física de la lógica de los IC[8]

[5] *Hardcore*: Unas de las categorías de los IP cores. Son bloques ya integrados en el silicio del dispositivo que viene configurado por el fabricante y que no puede ser reprogramado por el usuario. [16]

[6] IP Intelectual Propiety

definida, mediante la herramienta *PlatGen* (Hardware Platform Generation) dando click en la opción *Generate Netlist*, se construye el sistema de procesamiento empotrado como un conjunto de listas de conexionado hardware (es decir ficheros HDL y netlists de implementación). Para finalizar con la plataforma de Hardware se procede a la generación del *Bitstream* para la configuración de la FPGA [9]

Para finalizar el diseño del ES en la plataforma Software se crean librerías que contienen Drivers que permiten controlar los periféricos adicionados. Estas librerías se generan mediante la plataforma *LibGen* y son utilizadas por las aplicaciones software del sistema empotrado. Para el desarrollo software de la aplicación el XPS incorpora un compilador en lenguaje C que permite obtener el archivo ejecutable (.elf) que manejara el Microprocesador. Una vez terminado el diseño de la plataforma Software, se integran la parte de SW y HW para su posterior implementación en la FPGA, para esto en la pestaña de XPS *Device Configuration* primero se da un click en la opción *Update Bitstream* y después descarga en la tarjeta con *Download Bitstream.*[9]

En las siguientes secciones se expondrá el modelo seguido para desarrollo del ES propuesto, utilizando la metodología de codiseño basado en el procesador MicroBlaze.

En resumen para cada uno de los módulos se discutirá sobre su descripción en VHDL, procedimiento para su adición al procesador utilizando el entorno de desarrollo EDK y el desarrollo de una aplicación en software que permita validar el funcionamiento de dicho módulo. Finalmente se integran los tres módulos IP en un mismo ES con el fin de obtener el diseño mostrado en la figura 1.

IV. DISEÑO DE LOS MODULOS EN HARDWARE

Como se mencionó anteriormente, la primera fase del diseño es la descripción en VHDL de cada uno de los módulos IP, estas descripciones se tomaron del libro de P. Chu [5], dichos códigos son adecuados para ser utilizados en el *ES*, los periféricos teclado y lector de código de barras Ps2 aprovechan el mismo código en VHDL del circuito receptor y decodificador SCAN a ASCII ver figura 3, para el caso de la pantalla VGA su descripción en VHDL se implementa en dos circuitos para la sincronización y la generación de video figura 6. En este caso se utilizó el software ISE de Xilinx para verificar el funcionamiento de los módulos antes de la incorporación al microprocesador MicroBlaze.

1. Teclado y lector Código de Barras Ps2

La idea principal del diseño del modulo es la recepción del código *scan* [10] enviado por la tecla oprimida por el teclado caracterizándolo con un código en lenguaje ASCII para su recepción en el MicroBlaze. Basándose en este conocimiento la descripción en hardware debe contar de dos circuitos el receptor (**Ps2_rx**) y el codificador a ASCII (**ascii_code**).

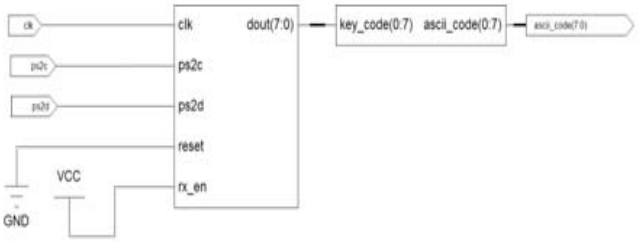


Figura 3 Receptor Ps2_rx y codificador ascii_code

Cada envío unidireccional que el periférico Ps2 hace al modulo IP contiene un BIT inicial, 8 bits de dato, un BIT de paridad y un BIT de parada [10], como se muestra en la figura 4.

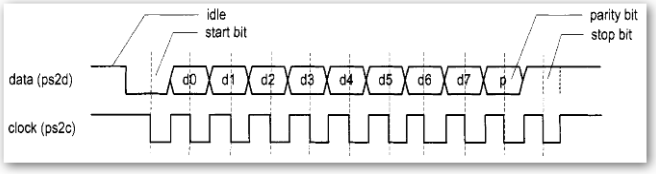


Figura 4 Trama enviada por el protocolo Ps2[3]

Esta recepción por parte del modulo se describe por medio de una maquina de estados como se muestra en la figura 5,

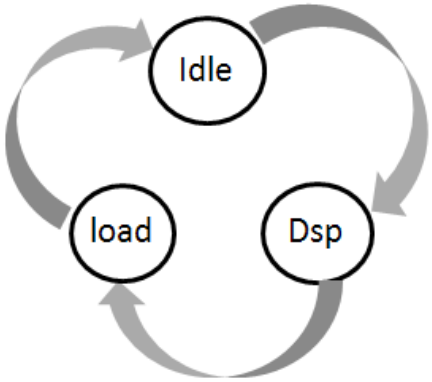


Figura 5 Maquina de estados del receptor Ps2_rx

En el estado *idle* o de inicio se espera un BIT de entrada proveniente del pin Ps2d el cual es transferido con el flanco descendente del reloj ps2c y el BIT de habilitación del receptor, a demás se activa la condición inicial en "1001" para el contador interno, que se encuentra en el siguiente estado "*dsp"*, donde se acumulan los 8 bits que conforman el dato final mas el BIT de paridad. Debido a la transmisión unidireccional el receptor siempre estará activo esperando el envío de información por parte del dispositivo Ps2.

Esta operación de acumular los bits de entrada en el registro se finaliza al completar los 9 bits (8 bits de datos + 1 paridad), indicando con un BIT de parada la finalización de dicho proceso, por último se requiere de un estado o ciclo de reloj *load* necesario para completar el envío al codificador ascii_code que se encarga de enviar al procesador el código ASCII del carácter oprimido.

2. Controlador VGA

La pantalla VGA se puede ver como un arreglo matricial de 480x640, donde cada uno de las celdas equivale a un pixel. El controlador VGA consta de una unidad para la generación de video y otra de sincronismo, como se muestra en la figura 5. Se define por coordenadas cartesiana el área de trabajo para hacer algún tipo desplazamiento vertical u horizontal a través de la pantalla. La unidad de sincronismo es la encargada de realizar dicho desplazamiento [5]

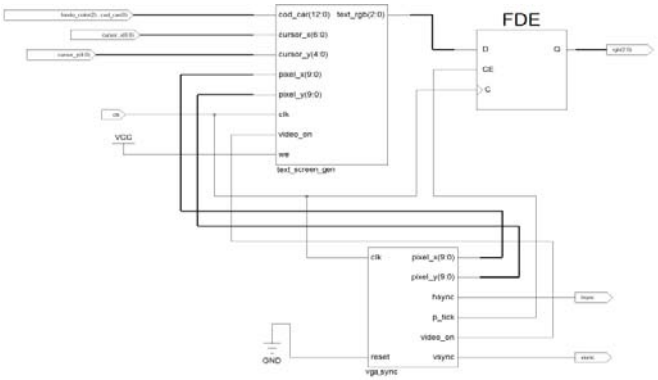


Figura 6 Conexión VGA

Se utilizan 8 colores por pixel (3 bits) lo cual con lleva a usar 921600 bits (640*480*3 bits) de memoria. Para reducir los requerimientos de memoria se usa un esquema de partes gráficas en la que cada parte gráfica tiene una resolución de 8x16 pixeles situación que reacomoda la resolución de la pantalla a 80 x 30 donde cada parte gráfica se maneja como una unidad de pantalla. Esto hace que el requisito de Memoria RAM sea de 2400 posiciones.

En la unidad de generación de video se puede encontrar dos tipos de memoria (ver figura 7): ROM (*Font_rom* contiene la patrones de partes gráficas) y RAM (memoria de acceso al contenido ROM).
La memoria RAM es el componente que permite almacenar la imagen en forma de datos, es decir donde para cada parte gráfica se definen color de fondo, código y color del carácter. Con esos tres elementos el circuito de generación de video puede reproducir la imagen propiamente dicha en la pantalla. El circuito de sincronización siempre está realizando lecturas periódicas a velocidad constante de todas las posiciones de la memoria. De la RAM sale el código ASCII almacenado en una posición de memoria que guarda relación uno a uno con la posición de la pantalla (es decir la posición en la que se almacena el carácter en la RAM es la misma posición en la que debe aparecer el carácter en la pantalla) y entra en la *Font_rom* donde se encuentran los mapas de bits de cada carácter.

El circuito generador de video entiende 0 como fondo y 1 como texto en el patrón del carácter o parte grafica. Estos se pueden modificar a conveniencia del programador.

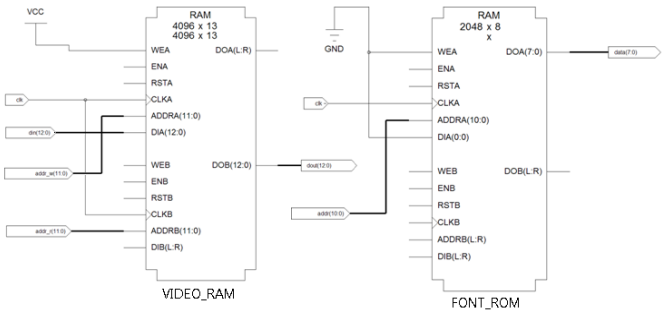


Figura 7 ROM (Font_rom contiene la alfabeto) VIDEO_RAM (memoria de acceso)

El valor agregado que se le dio al diseño de la descripción en VHDL del controlador VGA consiste en la posibilidad de darle una posición directa al cursor donde se va a escribir, además se dio la posibilidad de variar el color al texto y al fondo ampliando la memoria RAM y definiéndolos como salidas para que mediante unos registros se modifiquen por software.

V. ADICION DE MODULOS IP AL MICROBLAZE UTILIZANDO EL ENTORNO DE DESARROLLO EDK

Con el hardware de los módulos IP ya desarrollados exitosamente se continúa con la conexión de cada uno de éstos al MicroBlaze. En la figura 8 se muestra los archivos que se deben generar o actualizar para el proceso de conexión siguiendo el modelo para la implementación de los periféricos en EDK.

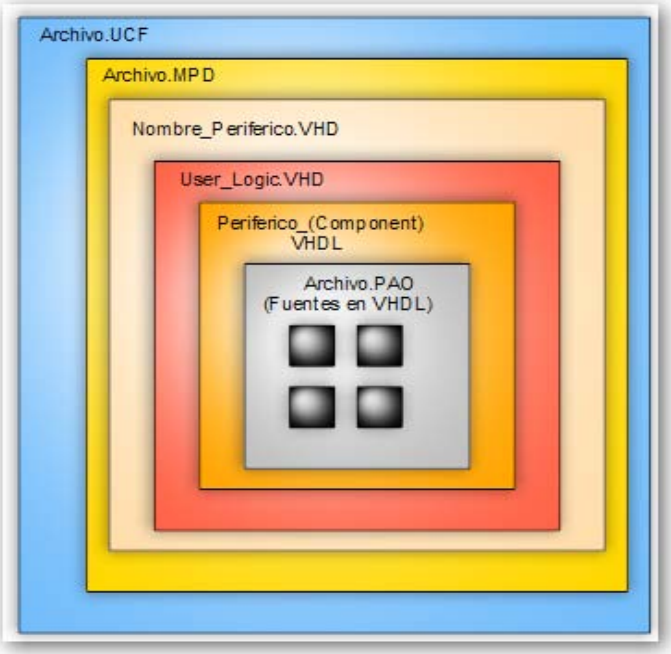


Figura 8 Módelo para la Implementación de los periféricos en EDK.

Dicho proceso de conexión empieza una vez termina la secuencia de pasos encontrada en la pestaña *Hardware, Create and import peripheral Wizard (*creación o importación

de periféricos) de EDK donde se define el nombre, versión, conexión al bus, numero de registros necesarios para su funcionamiento, entre otras características, allí se crean archivos que describen el hardware del periférico y su conexión con el Microprocesador MicroBlaze. Esta secuencia de pasos están descritos en *Lab 3: Adding Custom IP to an Embedded System Lab* [11].

Los archivos actualizados y generados en el transcurso del proceso de adición de módulos IP al procesador MicroBlaze se describen a continuación:

MPD (*Microprocessor Peripheral Description*)

Hallado en la dirección *pcores/Nombre_periferico/data/nombre_periferico.mpd* donde se encuentran todos los puertos disponibles con sus parámetros de hardware y descripción (dirección y tipo) del periférico [13].

NOMBRE_PERIFERICO.vhd

Ahora el archivo que se debe modificar es *nombre_periferico.vhd* se encuentra en la carpeta, pcores*/Nombre_periferico/hdl/* donde se debe definir los puertos externos del periférico y la instanciación al archivo de más bajo nivel USER_LOGIC.VHD [11].

USER_LOGIC.vhd

El archivo USER_LOGIC es uno de los archivos más importantes en el proceso ya que éste es la interfaz entre el microprocesador y el periférico. En la entidad se definen los puertos de conexión con el archivo de nivel superior (*nombre_periferico.vhd*). En la arquitectura del *user_logic* se desarrolla la conexión del periférico con el MicroBlaze, donde se declara el componente mediante la palabra COMPONENT definiendo las entradas y salidas del periférico y con la clave PORTMAP se define la conexión a los puertos, además se hace la conexión del periférico con los registros esclavos necesarios para la comunicación con el microprocesador. [11].

***PAO* (*Peripheral Analyze Order*)**

El siguiente archivo que se actualiza es el *.PAO* (*Peripheral Analyze Order*). Donde se define el orden de lista de archivos HDL necesarios para la síntesis y simulación [13] Cuando el periférico diseñado en VHDL tiene varias fuentes la forma de instanciarlas en EDK es a través de este archivo, donde se copia un código estándar en cada línea (por ejemplo *lib teclado_v1_00_a key2ascii vhdl*) que hace referencia a librería, *nombre_periférico*, nombre de la fuente y extensión. El orden en que se deben escribir los archivos es en forma descendente teniendo en cuenta la jerarquía o como están instanciados.
Estas fuentes deben estar copiadas en la carpeta *pcores/nombre_periferico/hdl*.

UCF (*User Contrains File)*

Por último el archivo UCF generado por *Base System Builder Wizard* de acuerdo a la plataforma de desarrollo en la cual se esté implementando el ES, en este caso la Spartan 3A DSP Board, este archivo se actualiza con los nuevos periféricos, teniendo en cuenta el tipo de módulo que se va a implementar. En el software EDK existe un estándar para la de declaración de puertos externos [12] donde para cada pin se define la localización del puerto físico y nivel de tensión [13].

A continuación se describe la modificación que se le hacen a los archivos de cada uno de los módulos IP presente en el proyecto para la adición al MicroBlaze

1. *Teclado ps2 y Lector de código de barras Ps2*

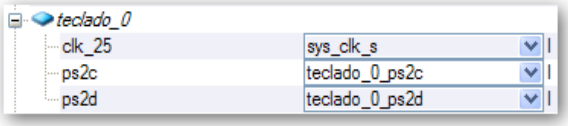
Aprovechando que los dos periféricos tienen la misma descripción en VHDL, los archivos que se actualizan y se generan comparten la misma estructura, pero no la misma fuente, Estos archivos se modifican siguiendo el modelo para la Implementación de los periféricos en EDK (Tabla 1).

Descripción de puertos en el archivo teclado_v2_1_0.mpd

```
## Ports
PORT ps2d = "", DIR = I
PORT ps2c = "", DIR = I
PORT clk_25  = "", DIR = I
```

El periférico requiere de dos puertos externos ps2d y pc2c.

Conexión del reloj en la ventana System Assembly View-Ports



El puerto del reloj *clk_25* no es asignado como pin externo porque este se conecta al reloj del procesador de manera interna. Esta conexión se realiza en la pestaña *ports* del *System Assembly View*.

Asignación de puertos en el archivo *teclado.vhd*

```
163     --USER ports added here
164      clk_25: in std_logic;
165      ps2d: in std_logic;
166      ps2c: in  std_logic;
```

Instanciación con el user_logic en el archivo teclado.vhd

```
392     --USER ports mapped here
393     ps2d                          => ps2d,
394     ps2c                          => ps2c,
395     clk_25                        => clk_25,
```

Declaración de puertos del user_logic.vhd

```
100   --USER ports added here
101    clk_25                        : in std_logic;
102    ps2d                          : in  std_logic;
103    ps2c                          : in  std_logic;
```

Entradas y salidas del periférico con el microprocesador mediante el archivo user_logic.vhd

```
 --USER signal declarations added here, as needed for user logic
COMPONENT kb_code
  PORT(
     ps2c           : IN std_logic;
     ps2d           : IN std_logic;
     reset          : IN std_logic;
     clk            : IN std_logic;
     ascii_code: out  std_logic_vector (0 to 7)
     );
  END COMPONENT;
```

Conexión del periférico con el microprocesador mediante el archivo user_logic.vhd

```
begin
  --USER logic implementation added here
Inst_teclado : kb_code
   PORT MAP(
      ps2d       => ps2d,
      ps2c       => ps2c,
      reset       => '0',
      clk         => clk_25,
      ascii_code  => slv_reg0(24 to 31)
   );
```

La salida *ascii_code* se conecta con el registro esclavo del procesador, y es por este registro por donde se envía el código ASCII de la tecla oprimida. La entrada reset se conecta a cero '0' para tener siempre habilitado el circuito receptor del periférico en modo de espera del código ASCII

Lista de documentos HDL en el archivo teclado_v2_1_0.pao

```
lib plbv46_slave_single_v1_00_a plbv46_slave_single vhdl
lib teclado_v1_00_a key2ascii vhdl
lib teclado_v1_00_a ps2_rx vhdl
lib teclado_v1_00_a keyboard vhdl
lib teclado_v1_00_a user_logic vhdl
lib teclado_v1_00_a teclado vhdl
```

Asignación de pines en el archivo system.ucf

```
#### Device keyboard

Net teclado_0_ps2d_pin LOC = L18 ;
Net teclado_0_ps2d_pin IOSTANDARD=LVCMOS33;
Net teclado_0_ps2c_pin LOC = E24 ;
Net teclado_0_ps2c_pin IOSTANDARD=LVCMOS33;
```

Tabla 1 Archivos modificados para el teclado y lector de código de barras.

2. *Controlador VGA*

Al final para el Controlador VGA las modificaciones a los archivos se realizan siguiendo la misma estrategia que para los periféricos anteriores (tabla 2)

Descripción de puertos en el archivo vga_color_v2_1_0.mpd

```
## Ports
PORT hsync = "", DIR = O
PORT vsync = "", DIR = O
PORT rgb = "", DIR = O, VEC =[0:2]
```

Los puertos externos para el controlador VGA son tres *hsync* (Sincronismo horizontal), *vsync* (Sincronismo vertical), la señal *rgb* (colores).

Asignación de puertos en el archivo vga_color.vhd

```
      hsync,vsync : out   std_logic;
      rgb         : out   std_logic_vector(0 to 2);
```

Instanciación con el user_logic en el archivo vga_color.vhd

```
      -- MAP USER PORTS BELOW THIS LINE
      hsync => hsync,
      vsync => vsync,
      rgb => rgb,
```

Declaración de puertos del user_logic.vhd

```
    --USER ports added here
    hsync :   out std_logic;
    vsync :   out std_logic;
    rgb   :   out std_logic_vector(0 to 2);
```

Entradas y salidas del periférico con el microprocesador mediante el archivo user_logic.vhd

```
component text_screen_top is
   port(
      clk: in std_logic;
      we : in std_logic;
      cursor_x: in std_logic_vector (6 downto 0);
      cursor_y: in std_logic_vector (4 downto 0);
      cod_car : in std_logic_vector (6 downto 0);
      text_color,fondo_color : in std_logic_vector (2 downto 0);
      hsync, vsync: out  std_logic;
      rgb: out std_logic_vector(2 downto 0)
   );
end component;
```

Conexión del periférico con el microprocesador mediante el archivo user_logic.vhd

```
begin
vga_module :  text_screen_top
   port map(
      clk=>Bus2IP_Clk,
      we =>slv_reg0(8),
      cursor_x=>slv_reg0(25 to 31), -- primer byte
      cursor_y=>slv_reg0(19 to 23),  -- segundo byte
      cod_car =>slv_reg0(9 to 15),   -- tercer byte
      text_color=>slv_reg1(29 to 31),
      fondo_color =>slv_reg2(29 to 31),
      hsync=>hsync,
      vsync=>vsync,
      rgb=>rgb
   );
```

Se utilizan tres registros esclavos por donde se trasmiten al periférico el carácter, posición de escritura, color del fondo, color del carácter. El reloj se conecta internamente al *Bus2IP_Clk* .

Lista de documentos HDL en el archivo vga_color_v2_1_0.pao

```
lib plbv46_slave_single_v1_00_a plbv46_slave_single vhdl
lib vga_color_v1_00_a video_ram vhdl
lib vga_color_v1_00_a font_rom vhdl
lib vga_color_v1_00_a text_scr_gen vhdl
lib vga_color_v1_00_a vga_sync vhdl
lib vga_color_v1_00_a text_scr_top vhdl
lib vga_color_v1_00_a user_logic vhdl
lib vga_color_v1_00_a vga_color vhdl
```

Asignación de pines en el archivo system.ucf

```
#### Device VGA
 ## RED
NET "vga_color_0_rgb_pin<0>"     LOC = "L20"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_0_pin<0>"   LOC = "M20"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_1_pin<0>"   LOC = "F25"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_2_pin<0>"   LOC = "F24"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
 ## GREEN
NET "vga_color_0_rgb_pin<1>"     LOC = "M19"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_0_pin<1>"   LOC = "M18"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_1_pin<1>"   LOC = "J23"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_2_pin<1>"   LOC = "J22"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
 ## BLUE
NET "vga_color_0_rgb_pin<2>"     LOC = "L22"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_0_pin<2>"   LOC = "K21"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_1_pin<2>"   LOC = "G23"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_rgb_2_pin<2>"   LOC = "G24"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "vga_color_0_hsync_pin"      LOC = "K26"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "vga_color_0_vsync_pin"      LOC = "K25"   | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
```

Tabla 2 Archivos modificados para el controlador VGA

VI. TEST DE FUNCIONALIDAD (APLICACIÓN EN C++)

Una vez adherido el Hardware del periférico al procesador. En la plataforma XPS se crea un test de funcionalidad o Software el lenguaje C++ para cada periférico implementado en el MicroBlaze, dicho test hace el papel de *DRIVER*, permitiendo controlar el funcionamiento del periférico para dar origen a una aplicación. El XPS genera para este propósito funciones que admiten la escritura y la lectura de cada registro del periférico. Los archivos donde se encuentran las librerías y los nombres de las funciones están almacenados en la carpeta *drivers/ Nombre_del_periferico/scr*. Estas funciones se crean al dar click en Generate Libraries and BSPs en la pestaña Software [11,13].
Los datos enviados por los módulos teclado y lector de código de barras son leídos en el hyperterminal[7], esta comunicación se logra con la adición del puerto serial RS232[8]. Se comprueba la conexión del teclado al procesador, con el diseño en C, de un código que manipula la función *TECLADO_mReadSlaveReg0* para leer el código ASCII enviado por el modulo de la tecla oprimida y así poder mostrarla en los Led's de la tarjeta Spartan 3A DSP y en el hyperterminal.

Como el Lector de código de Barras es un emulador de teclado, envía los caracteres del código leído de forma instantánea, es decir, como si las teclas fueran oprimidas de forma rápida, se logra la identificación del código enviado por el Lector en el hyperterminal por medio de la función en C *xil_printf*. En la trama de datos se observa que el Lector envía un carácter del código leído varias veces y finaliza el envió del código con el carácter 0x0D (ENTER en código ACSII), teniendo en cuenta estas características el código en C se diseña para que realice durante un tiempo de programación definido un conteo cuando se registra el cambio del dígito, validando así la recepción de un carácter almacenado los 13 dígitos del Código Ean 13[9] en un vector.
La lectura de dicho código se realiza mediante la función *LECTOR_CODIGO_DE_BARRAS_mReadSlaveReg0*.

Por último para el controlador VGA las funciones de escritura y lectura necesarias para controlar los tres registros son:

- *VGA_COLOR_mWriteSlaveReg0*: Donde se indica la posición del carácter a escribir (Coordenadas X y Y del cursor), y el código ASCII del carácter.
- *VGA_COLOR_mWriteSlaveReg1: Donde se escribe el color del texto.*
- *VGA_COLOR_mWriteSlaveReg2: Donde se escribe el color del fondo.*

En la aplicación en C++ se diseña un código en cual se define en la pantalla una región de visualización para mostrar las cadenas de caracteres enviadas en distintas posiciones, con diferentes colores de texto y fondo.

VII. INTEGRACIÓN DE LOS MODULOS IP

En ciertas ocasiones cuando se sale de compras al supermercado no es fácil llevar el control de los productos que se quieren adquirir en relación al presupuesto con el que se cuenta antes de pasar por la caja, una solución a este problema está dada por el ES propuesto, donde este sistema quiere realizar la misma labor que un verificador de precios comercial e incluso puede llegar a adaptarse como el software de una caja registradora.

Con los módulos IP ya implementados y verificados en el procesador MicroBlaze se crea el ES con la integración de los tres periféricos en un solo diseño como se puede ver en el diagrama de bloques de la figura 9 donde se observa que dicha conexión se hace mediante el Bus PLB [6].

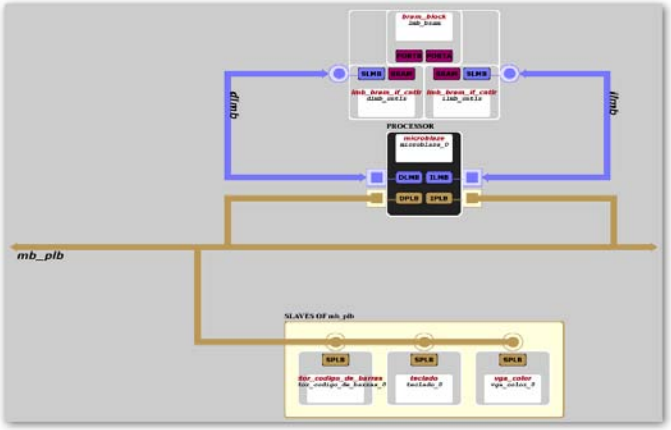


Figura 9 Diagrama de bloques en EDK del ES.

El código ejecutado por el microprocesador para esta aplicación íntegra las funciones creadas por la herramienta LibGen para los módulos teclado, lector de código de barras e interfaz VGA y otras funciones creadas por los autores de este artículo para la optimización y la mejora de los recursos del compilador en C que posee la XPS

De acuerdo a lo anterior cada una de las funciones realiza:

- Definir la región de visualización mediante coordenadas (x,y) imprimiendo el carácter espacio en toda esta área.
- Imprimir una cadena de caracteres en una ubicación a conveniencia del programador.
- Dos funciones de retardo.
- Almacenar los datos leídos en un vector por el lector de código de barras.
- Lectura de las teclas oprimidas en el teclado
- Guardar los datos digitados en el teclado en un vector e ir visualizando cada carácter en el VGA
- Conversor de valores hexadecimales a valores enteros.
- Conversor de valores enteros a hexadecimales.
- Sumar los valores de los productos consultados.

[7] Hyperterminal es un programa que se puede utilizar para conectar con otros equipos mediante un modem, cable de Puerto serial o UART entre otros [14].
[8] RS232 interfaz incorporada por XPS.
[9] Código Ean 13, forma de codificación de la información utilizada en el comercio [15].

Con las funciones anteriormente mencionadas se da fiabilidad y estabilidad al ES propuesto, además en la función principal de la plataforma en software se realiza una base de datos la cual contiene el nombre, precio y el código de acceso del artículo que corresponde a los últimos 5 códigos de acuerdo con las propiedades del código Ean 13 [15] del código de barras, para acceder a la base de datos el usuario debe pasar el lector de código de barras o digitar por medio del teclado el código de dicho artículo este código entra por medio de consulta a la base de datos si el código es correcto en la interfaz VGA se visualizara el detalle y el precio si el código ingresado es erróneo o no existe en la base de datos se muestra el mensaje "*articulo no encontrado*", para calcular el valor de varios artículos a cada uno se le asigna su valor en pesos como variable entera para poder efectuar las operaciones matemáticas necesarias para el cálculo de dicha cuenta y la visualización en la VGA.

La arquitectura física de la aplicación se muestra en la figura 10 donde se utiliza un monitor de 15" con conexión VGA de 15 Pines, un teclado con puerto Ps2, un lector de código de Barras Ps2, dos módulos PS2 Diligent y la Spartan 3A DSP.

Figura 10 Arquitectura Física del ES

Se muestra la interfaz gráfica que maneja el usuario en la figura 11, con el color de texto y fondo definido, mostrando también el código, el detalle y el valor del último artículo consultado, además parte inferior se sitúa el valor de la cuenta total realizada en la consulta, el usuario tiene la opción de realizar una nueva cuenta presionando la tecla N.

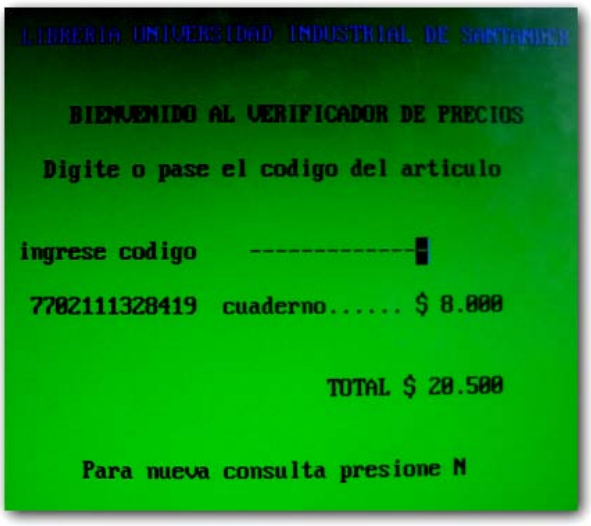


Figura 11 Interfaz Grafica del ES

VIII. CONCLUSIONES

- La plataforma SPARTAN 3A DSP de Xilinx, aumenta sus posibilidades de desarrollo de sistemas empotrados con la incorporación del procesador Sofcore MicroBlaze y el desarrollo e inclusión de periféricos propios al sistema.

- Al trabajar con FPGAs se tiene la posibilidad de obtener diseños flexibles, de alta Confiabilidad, reconfiguración y capacidades para desarrollar prototipos rápidamente, a demás debido a la reprogramación se puede contar con la facilidad de probar, verificar e implementar cambios en el diseño, de acuerdo a las condiciones específicas de la aplicación aun después de que éste en manos del usuario.

- La ventaja de utilizar FPGA radica en que todo el desarrollo se lleva a cabo en un solo ambiente de trabajo, dando la posibilidad al diseñador de proponer funciones lógicas y mediante el uso de Lenguaje de Descripción de Hardware, definir los parámetros de su problema.

- El uso de herramientas CAD facilita el diseño sobre FPGAs reconfigurándola las veces que sea necesaria, además permite realizar la simulación del diseño permitiendo de esta forma verificar su funcionamiento.

- Con la implementación del ES *Verificador de Precios* se comprobaron las posibilidades que ofrece el codiseño pues esta metodología permite aprovechar las ventajas tanto del Hardware como del Software en un solo diseño.

- El presente diseño es factible de ampliar fácilmente en futuros trabajos, gracias a su diseño modular.

IX. AGRADECIMIENTOS

Esta tesis está dedicada primero a Dios, por estar en nuestro corazón y estar en cada paso quedamos, a nuestras familias por su cariño y comprensión durante toda la carrera, a nuestros amigos que nos acompañaron durante todo el proceso de aprendizaje. Debemos agradecer de manera especial y sincera a nuestro Director Carlos por la colaboración, apoyo y conocimiento durante el desarrollo de la tesis.

REFERENCIAS

[1] R. sanchéz, Estado del Arte de los Sistemas Embebidos http://coitt.es/res/revistas/Antena161_05_Reportaje.pdf. Revisado nov 2008

[2] A. Helmerich. et. al. "Study Of Worldwide Trends And R&D Programmes In Embedded Systems Introduction", European Communities 2005 ftp://ftp.cordis.europa.eu/pub/ist/docs/embedded/final-study-181105_en.pdf

[3] CA. Fajardo "Apropiacion tengologica del diseño de Embededd Systems Implementados sobre FPGAs y CPLDs"

[4] Spartan-3A DSP Starter Platform User Guide http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf

[5] P. Chu. "FPGA prototyping by VHDL Examples". Jhon Wiley & Son. 2008.

[6] MicroBlaze Soft Processor Core http://www.xilinx.com/tools/microblaze.htm

[7] Floyd Thomas, Digital fundamentals, Ninth Edition, Pearson Education International, 2006

[8] IP core: htpp://whatis.techtarget.com/definition/0,,sid9_gci759036,00.html

[9] J. Viejo y colaboradores. "Diseño E Implementación Óptima De Periféricos De DSP Con *System Generator* Para *MicroBlaze*" Departamento de Tecnología Electrónica-Universidad de Sevilla. 2005 http://www.dte.us.es/id2/OFU/publicaciones.php

[10] *Lab8. Puerto Ps2 departamento tecnológico U Rey Juan Carlos* http://laimbio08.escet.urjc.es/assets/files/docencia/DCSE/dcse_p8_ps2.pdf

[11] *Lab 3: Adding Custom IP to an Embedded System Lab* https://intranet.etc.upt.ro/~DocTehnice/Xilinx/XUP/embedded/labdocs/lab3.pdf

[12] Starter_kit_constraints.ucf http://www.xilinx.com/products/boards/s3astarter/files/s3astarter.ucf

[13] EDK Concepts,tolos, and Techniques http://www.xilinx.com/support/documentation/sw_manuals/edk10_ctt.pdf

[14] Introducción a HyperTerminal http://technet.microsoft.com/es-es/library/cc736511(WS.10).aspx

[15] *El código Ean 13* https://forja.rediris.es/docman/view.php/136/134/Gu%C3%ADa%20EAN%20de%20codificaci%C3%B3n%20b%C3%A1sica.pdf lector codigo de barras Ean 13 rediris .es

[16] Xilinx Programmable Logic Data Book 2000

Jorge A. Pachón Cañas nació en Bogotá, Colombia, en 1986. Recibe su grado de Ingeniería electrónica en 2010 de la Universidad Industrial de Santander.

Juan G. Ortiz Cala nació en el Socorro, Colombia, en 1986. Recibe su grado de Ingeniería electrónica en 2010 de la Universidad Industrial de Santander.

Carlos A. Fajardo Ariza, es Ingeniero Electrónico y Especialista en Docencia de la Universidad Industrial de Santander, en la cual es catedrático desde el año 2002. Actualmente se encuentra adscrito al grupo de investigación *Conectividad y procesamiento de Señales* (CPS) y adelanta estudios de Maestría en Ingeniería Electrónica esta misma universidad. Sus campos de interés son los FPGAs, los Embedded System y la aceleración de algoritmos por medio de FPGAs.