MASKAY 2(1), Nov 2012
ISSN 1390-6712

Recibido (Received): 2012/05/02
Aceptado (Accepted): 2012/08/16

Diseño de Hardware y Software de Systems on Chip empleando tecnología Xilinx EDK

Julio Cadena S., Gabriel Mollocana L, Hugo Ortiz, y Vanessa Vargas V.

Resumen **El presente artículo resume el proceso empleado para obtener el primer *System on Chip* (SoC) diseñado, desarrollado, y emulado en la Escuela Politécnica del Ejército (ESPE) y en el Ecuador. Se demostrará que combinando las ventajas del diseño sobre *Field Programable Gate Arrays* (FPGAs) empleando la reutilización de *IP Cores* y plataformas, junto al uso de la tecnología de desarrollo Xilinx EDK, se puede diseñar tanto el *hardware* como el *software* de un *chip* de manera rápida y económicamente fiable. Además, se detalla el uso de la metodología *Platform Based Design* (PBD) y del concepto de co-diseño de *hardware* y *software* para diseñar las capas de *hardware*, sistema operativo y aplicación de un *chip*. La capa de *hardware* contiene una serie de *IP Cores* gobernados por un procesador *MicroBlaze* trabajando dentro de la arquitectura *CoreConnect* de IBM. Mientras que la capa de sistema operativo está conformada por *drivers*, librerías y el Sistema Operativo en Tiempo Real (RTOS) *Xilkernel*. Por último, la capa de aplicación tiene la funcionalidad de controlar una planta de temperatura, mediante la selección de dos técnicas de control: *ON-OFF* o PID. Cabe destacar que el co-diseño se desarrolló considerando un adecuado enfoque conceptual, arquitectural, y metodológico .**

Palabras Claves- SoC; IP Cores; Sistemas Embebidos; MicroBlaze; RTOS.

I. INTRODUCCIÓN

La tendencia de la tecnología actual está basada en dispositivos electrónicos o sistemas embebidos, que posean más funciones y un mayor rendimiento, consuman menos potencia, tengan un menor tamaño y un menor precio. Además, estos sistemas deben estar disponibles lo antes posible en el mercado de consumidores. Estas características motivaron a la industria electrónica a crear una nueva metodología en el diseño de circuitos integrados. De esta manera aparecen los *System on Chip* (SoC).

System on chip es una tendencia a la que se le ha dado gran importancia en países que basan su economía y desarrollo en la fabricación de productos de alta tecnología. El nivel más avanzado de la tecnología en el campo de diseño de *chips* se ha logrado, mediante la implementación de sistemas embebidos basados en SoCs sobre *Field Programmable Gate Arrays* (FPGA).

Julio Cadena S., Gabriel Mollocana L., Hugo Ortiz, Vanessa Vargas V., Carrera de Ingeniería en Electrónica, Automatización y Control, Departamento de Eléctrica y Electrónica, Escuela Politécnica del Ejército, ESPE, Sangolquí, Ecuador, E-mails: juliocadena20@hotmail.com, resistron@hotmail.com, hortiz@espe.edu.ec, vcvargas@espe.edu.ec.

Los FPGAs facilitan el desarrollo de nuevos productos gracias a su reprogramabilidad en el momento mismo del diseño. La finalidad de estos dispositivos es permitir a los diseñadores de circuitos integrados plasmar sus ideas, en un menor tiempo, realizando constantes pruebas y cambios, hasta llegar al objetivo deseado. Para esto, los fabricantes de FPGA se esmeran en promover el diseño de herramientas que faciliten el diseño de plataformas de *hardware*, así como el desarrollo de herramientas para el desarrollo del *software* que se ejecutará sobre esta plataforma.

Cabe señalar que el desafío de los diseñadores de dispositivos electrónicos es integrar un mayor número de elementos en un simple *chip*, sin incrementar el tamaño del mismo, disminuyendo el tiempo de salida del producto al mercado (*time-to-market*) y aumentando el tiempo del producto en el mercado (*time-in-market*). De allí la importancia de los sistemas embebidos basados en SoCs que están diseñados para hacer alguna tarea específica, en lugar de ser un computador de propósito general para múltiples tareas.

En la actualidad, los productos basados en SoCs se encuentran abundantemente en el mercado, y van desde dispositivos portátiles, como los relojes digitales, celulares, reproductores de MP3, hasta grandes instalaciones estacionarias, luces de tráfico, controladores industriales y sistemas de control de las centrales eléctricas.

Por otra parte, el Ecuador, considerado un país en vías de desarrollo, ha estado tradicionalmente limitado en el diseño de tecnología debido a costos y a la falta de profesionales capacitados en este campo. Sin embargo con el uso de FPGAs se puede iniciar con el estudio, diseño e implementación de SoCs de forma económicamente fiable. De esta manera se augura que en un futuro cercano el Ecuador se convierta en desarrollador y exportador de tecnología.

II. FUNDAMENTO TEÓRICO

A. System on Chip

De acuerdo a Martin y Chang 2003, SoC es un circuito integrado complejo que integra la mayoría de elementos funcionales de un producto final completo dentro de un simple *chip* [1]. El uso de SoCs permite crear sistemas embebidos de menor tamaño y que incorporen mayor tecnología. La idea fundamental es convertir lo que hoy en día es un *Printed Circuit Board* (PCB)[*] con

[*] ***Printed Circuit Board***: Tarjeta de Circuitos Impresos que contiene

componentes discretos en un simple SoC integrado. Tradicionalmente, diferentes componentes eran colocados e interconectados sobre una tarjeta PCB con la finalidad de cumplir una función específica (Fig. 1).

Con la utilización de *Intellectual Property Cores (IP Cores)*, los *chips* individuales que conformaban los componentes en *hardware* fueron reemplazados por componentes virtuales, que cumplen las mismas funciones (Fig. 2). Agrupar los componentes dentro de un mismo *chip* disminuye notablemente el tamaño y consumo de potencia de los productos ofrecidos, obteniéndose una gran ventaja respecto a sistemas tradicionales.

Cabe señalar que los SoCs se basan en el diseño y reutilización de los bloques de propiedad intelectual *IP Cores* [2]. En la Fig. 3, se observa varios elementos que conforman un SoC, entre los que se destacan un procesador programable, memorias *on chip*, unidades de aceleración implementadas en *hardware*, interfaces con dispositivos periféricos, y aunque no consta en el gráfico, en un futuro podrían incluirse componentes analógicos y *opto/microelectronic mechanical system* (O/MEMS) [1].

B. *IP Cores*

Los *IP Cores* o Núcleos de Propiedad Intelectual son bloques con funciones preestablecidas, previamente probadas y verificadas por empresas desarrolladoras, para que posteriormente puedan ser integrados en sistemas SoC.

Una ventaja adicional de la reutilización de *IP Cores*, es que ofrecen una gran reducción en el riesgo de diseño de nuevos dispositivos al basarse en módulos pre-probados.

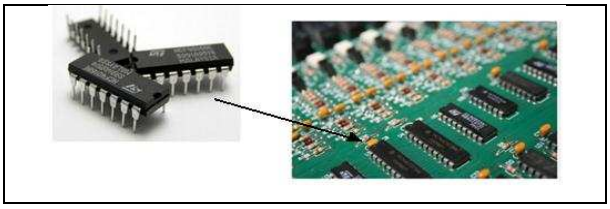

Fig. 1. Componentes Reales

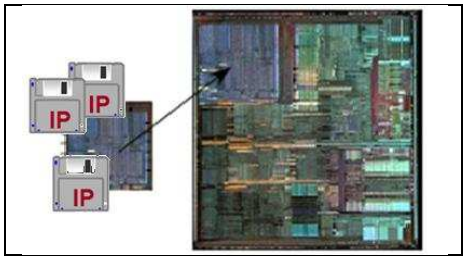

Fig. 2. Componentes Virtuales

varios componentes discretos interconectados a través de rutas o pistas de material conductor.

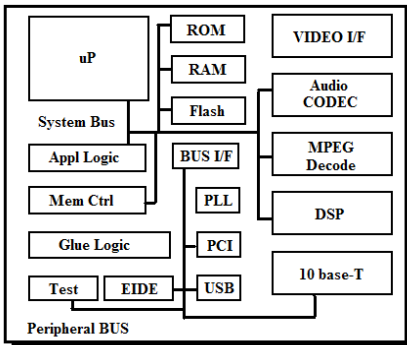

Fig. 3. *System on Chip*

Existen 3 tipos de *IP Cores*:

- *Hard Cores*
- *Soft Cores*
- *Firm Cores*

La Tabla I muestra un resumen de las características de cada tipo de *IP Core*.

C. *Metodologías de Diseño*

Las metodologías de diseño primarias de circuitos integrados se dividen en tres segmentos:

- Diseño Guiado por Tiempo (TDD)
- Diseño Basado en Bloques (BBD)
- Diseño Basado en Plataforma (PBD)

Los diseñadores prefieren utilizar la tecnología de Diseño Basado en Plataforma (PBD) puesto que permite disminuir el *time to market* expandiendo las oportunidades y la velocidad de distribución de sus productos derivados. Además, reduce varios riesgos involucrados en el diseño, facilitando la verificación de un SoC complejo debido a la gran reutilización de combinaciones de *IP Cores.*

Una plataforma está constituida por un conjunto de equipos y *software* básico, sobre el cual un grupo o familia de productos se pueden diseñar o construir gracias a que posee características comunes y una gestión integrada [1] (Fig. 4).

En la metodología PBD, el diseño agrega la reutilización de grupos de *IP Cores* en una arquitectura en lugar de mirar a la reutilización de *IP Cores* bloque por bloque [3]. La idea principal de la plataforma es simplificar el proceso de diseño.

Una vez analizados los conceptos de plataforma en la siguiente sección se va describir la plataforma utilizada en el desarrollo de este proyecto.

TABLA I. RESUMEN DE LAS CARACTERÍSTICAS DE IP CORES

Tipo	Soft Core	Firm Core	Hard Core
NIVEL DE ABSTRACCION	Register Transfer Level (RTL), *gate level*	*Gate level, layout*	*Layout*
DESCRIPCION	VHDL, *Verilog*	*Netlist* [a]	Descripción de transistores
PORTABILIDAD	A todas las tecnologías	Limitada a tecnologías probadas	Optimizada a una tecnología especifica
FLEXIBILIDAD	Alta	Limitada	Muy poca
PREVISIBILIDAD	Baja	Buena	Alta y definida por la tecnología
PROTECCION PROPIEDAD INTELECTUAL	Difícil	Fácil	Fácil

a. Netlist: Representación en lenguaje de descripción de *hardware* de la conectividad de un circuito.

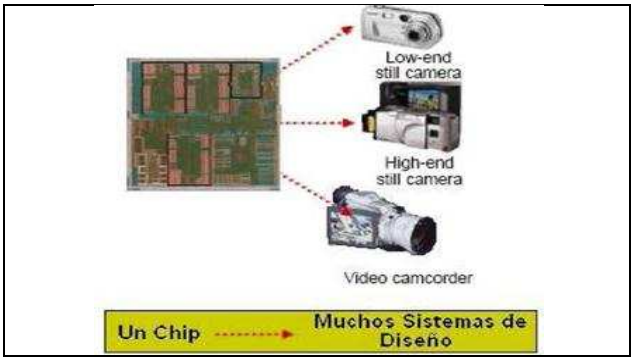

Fig. 4. Plataforma para diferentes equipos de electrónica de consumo

D. *Co-Diseño de Hardware y Software*

En el flujo o proceso de diseño convencional, grupos independientes de expertos diseñan el *hardware* y el *software* de un *chip,* sin que exista necesariamente cooperación entre ellos (Fig. 5a). Sin embargo en el diseño de SoCs se plantea un nuevo concepto, llamado "Co - Diseño", en el cual el *chip* es diseñado por grupos de expertos en cooperación (Fig. 5b).

En el Co–diseño, el *hardware* y el *software* de un sistema embebido se desarrollan en paralelo, realizando constantes realimentaciones entre los equipos de diseño. El resultado es que cada parte puede tomar ventaja de lo que la otra puede hacer. La explicación de las fases del co-diseño se realizará junto con la implementación en la sección IV.

III. PLATAFORMA DE DESARROLLO XILINX SPARTAN-6 FPGA EMBEDDED KIT

A. *Plataforma de Hardware*

El Xilinx Spartan-6 FPGA Embedded Kit tiene como tarjeta de desarrollo el modelo SP605 [4]. Esta tarjeta permite a los diseñadores de *hardware* y *software* emular sus diseños sobre el FPGA *Spartan* 6 LX45T (parte central de la Fig. 6). Cabe mencionar que este kit incluye toda la documentación del *MicroBlaze Processor Subsystem*, el cual es un SoC que puede ser utilizado como plataforma base para el desarrollo de proyectos, como es el caso del sistema implementado.

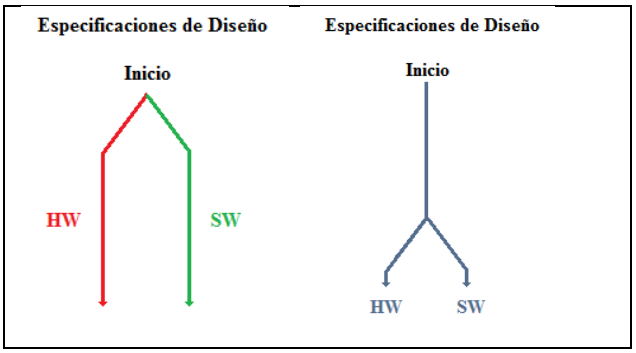

Fig. 5. (a) Flujo de Diseño Tradicional (b) Flujo de Co-Diseño

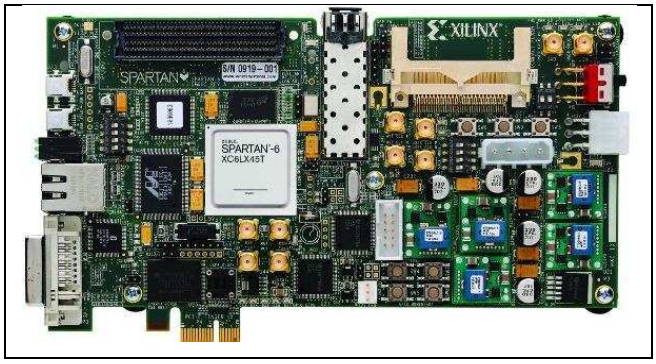

Fig. 6. Tarjeta de desarrollo SP605

B. *Plataforma de Software*

Está constituida por el *ISE Design Suite Embedded Edition* 12.1. Este *software* proporciona herramientas para el diseño embebido y una serie de *IP Cores* adaptados a las necesidades comunes de los desarrolladores. Una de sus herramientas principales es el *Embedded Development Kit* (EDK).

El EDK incluye ***Xilinx Platform Studio* (XPS)** (Fig. 7), para el diseño de *hardware,* y ***Software Development Kit* (SDK)**(Fig. 8) para el diseño de *software*. Así como, la documentación de la mayoría de *IP Cores* que se podrían necesitar en el diseño de SoCs con procesadores *PowerPC* y/o *MicroBlaze* [5 - 6].

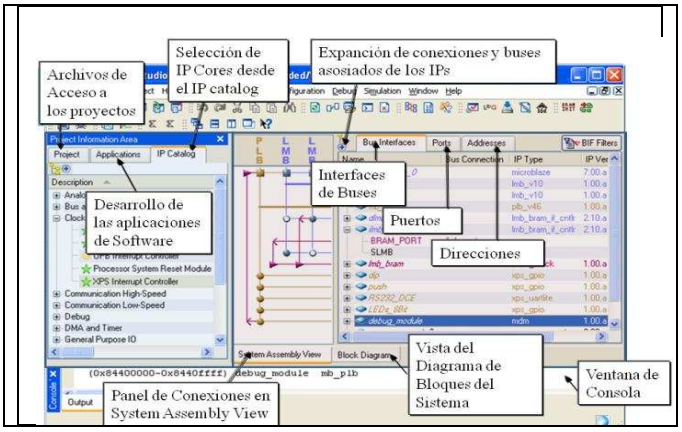

Fig. 7. Vista general del XPS

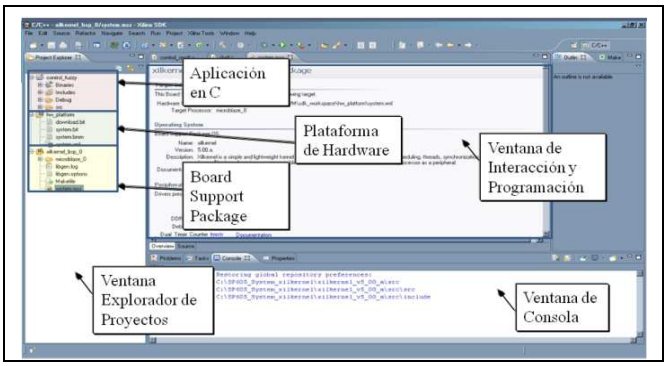

Fig. 8. Vista general del SDK

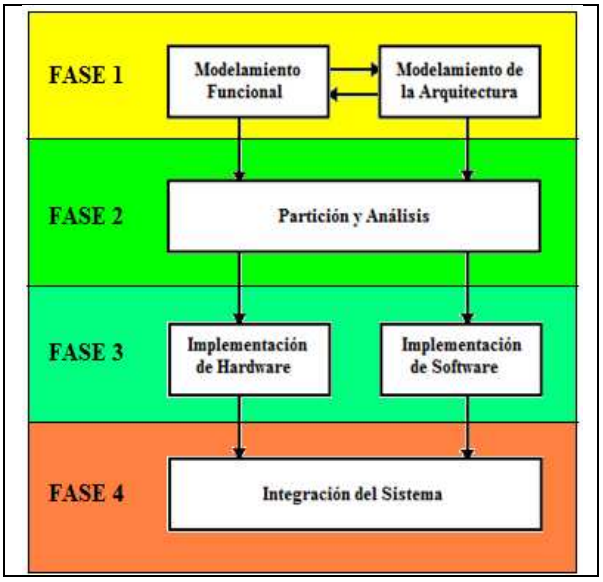

Fig. 9. Fases del Co-Diseño HW/SW

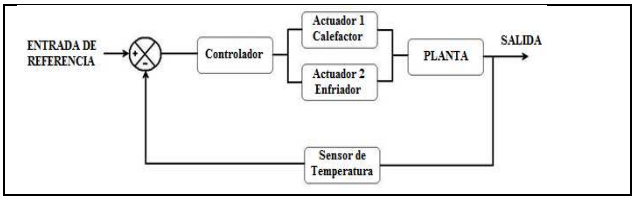

Fig. 10. Esquema de Proceso de Control para Planta de Temperatura

IV. IMPLEMENTACIÓN DEL CO-DISEÑO DE HARDWARE Y SOFTWARE

El SoC diseñado en este proyecto se basa en las fases del proceso de Co-diseño, (Fig. 9) propuestas por Martin y Chang en el libro *Surviving the SoC Revolution* de 1999.

A continuación se detallará cada fase con los resultados obtenidos en este proyecto:

A. *Fase 1*

1) *Modelamiento Funcional*

En esta fase se establece los requerimientos del producto, y se verifica las especificaciones del funcionamiento del sistema.

La planta que se desea controlar tiene las siguientes especificaciones técnicas: un rango de control de temperatura entre 40 y 65 grados centígrados, y un consumo de corriente de 1.35 [A].

El proceso de control de la planta se muestra en la Fig. 10. Donde, el elemento controlador está conformado por el SoC emulado sobre la tarjeta de desarrollo SP605.

Dadas las especificaciones de la planta, el SoC a diseñarse requiere:

En el *hardware*: Un procesador, memorias, buses, y periféricos de entrada/salida de propósito general, adquisición de datos, controlador de interrupciones, módulo de depuración, interfaz de comunicación serial y soporte para un Sistema Operativo en Tiempo Real (RTOS).

En el *software*: Un RTOS para el trabajo con hilos, semáforos e interrupciones, *drivers* para dispositivos de *hardware*, código de rutinas e hilos para realizar el control de la planta, y una Shell CLI (Com*mand Line Interface*) que permita la selección de varios sub-programas a través del ingreso de comandos.

2) *Modelamiento de la Arquitectura*

Una vez que las especificaciones funcionales están definidas se procede a escoger una arquitectura que ejecute las funciones del sistema. Generalmente esta arquitectura queda definida por la plataforma que se vaya a emplear.

Tal como se señaló la plataforma de desarrollo que se utilizó es XILINX SPARTAN-6 FPGA EMBEDDED KIT. Sus herramientas facilitan la implementación de una arquitectura de *hardware,* basada en el estándar

CoreConnect de IBM y en un procesador *Microblaze* para el procesamiento de la información y toma de decisiones. Esta arquitectura se define en el *software* XPS.

CoreConnect implementa un *Processor Local Bus* (PLB) para conectar el CPU a los periféricos, y un *Local Memory Bus* (LMB) para conectarlo a las memorias del sistema. Además, su topología es tipo bus, lo que significa que los *IP Cores* comparten una misma línea y protocolo de comunicación. La ventaja de utilizar esta topología es que si falla un elemento no genera el fallo de todo el sistema (Fig. 11).

B. Fase 2: Partición y Análisis

En esta fase se realiza una partición del modelo funcional sobre el modelo de la arquitectura. Es decir, se asigna las tareas del sistema a un recurso específico de *hardware,* o a un recurso de *software*.

El modelo funcional ha sido establecido en forma de capas, en base a las especificaciones necesarias del sistema, como se muestra en la Fig. 12.

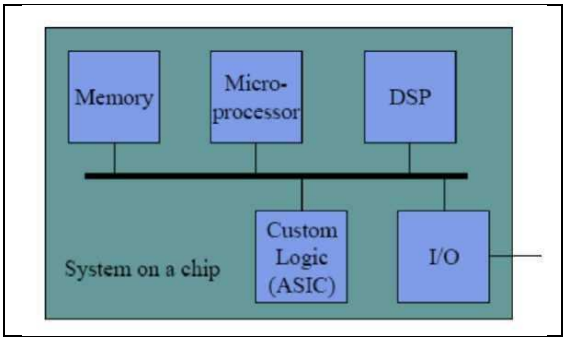

Fig. 11. SoC basado en topología BUS

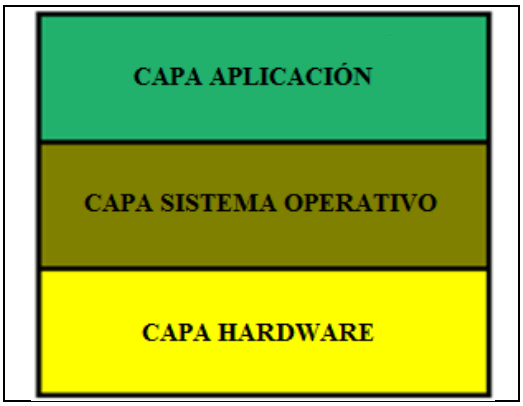

Fig. 12. Vista en Capas del Diseño

- **CAPA *HARDWARE*:** Diseño realizado en XPS de la plataforma de *hardware*.
- **CAPA SISTEMA OPERATIVO:** BSP creado en SDK de la plataforma de *software*.
- **CAPA APLICACIÓN:** Aplicación de *software* en lenguaje C desarrollada en SDK.

La Capa de *Hardware* ejecutará las siguientes tareas:

- Adquisición de datos del sensor de temperatura – *XPS ADC core*.
- Manejo de actuadores de calefacción (Foco) y de enfriamiento (Ventilador) – *XPS GPIO core.*
- Interfaz de usuario RS232 – *UART_16550_core.*
- Controlador de memoria externa DDR3 – *MPMC DDR3 SDRAM*.
- Reloj del sistema – *clock generator*.
- Soporte para RTOS – *XPS Timer.*
- Generación de periodo de muestreo – *XPS Timer.*
- Controlador de interrupciones – *XPS Interrup Controler*.
- Depuración del sistema – *Debug Module*.

Por otro lado, el *software* del sistema contiene las capas de Sistema Operativo y de Aplicación. Las funciones que cumple la Capa de Sistema Operativo, a través de las librerías del RTOS Xilkernel son:

- *Scheduling*
- APIs para hilos POSIX
- APIs para semáforos
- APIs para interrupciones
- *Drivers* de periféricos de *hardware*

En tanto que a la Capa de Aplicación, realizar las siguientes rutinas de *software* en lenguaje C:

- Interfaz de línea de comandos (*Shell* CLI).
- Control de temperatura ON-OFF.
- Control de temperatura PID.
- Gestión del reloj del sistema.
- Gestión de interrupciones.
- *Hardware Setup*.

C. Fase 3

1) Implementación de Hardware

Esta fase abarca el diseño de nuevos bloques de *hardware* y la integración de bloques reusables o *IP Cores*. Finaliza con la síntesis del código VHDL de la plataforma de *hardware* resultante.

Para crear el SoC controlador se tomó como referencia la plataforma *MicroBlaze Processor Subsystem,* basada en el procesador *MicroBlaze.* De esta plataforma se eliminó los *IP Cores* que no eran de interés. Además, se añadió otros *IP Cores* para complementar el sistema y se realizó nuevas configuraciones en algunos de los ya existentes.

a) Capa de Hardware

El procedimiento de diseño de la capa de hardware empleando las herramientas XPS es el siguiente:

- Personalización del *Microblaze Processor Subsystem*
- Asignación De Pines Del FPGA *Spartan 6* en el Archivo UCF
- Generación del Archivo .bit (*Bitstream*) de la Plataforma De *Hardware*
- Exportación de la Plataforma de *Hardware* al SDK

El SoC creado tiene el diagrama de bloques (Fig. 13) expresado a partir de la vista RTL que ofrece el XPS. En este diagrama constan los *IP Cores* listados en la Tabla II.

El mapa de memoria a través del cual el procesador *MicroBlaze*, accede a los registros internos de cada uno de estos *IP Cores* se muestra en la Tabla III.

Los resultados de la síntesis de este *hardware* se muestran en la Tabla IV.

De acuerdo a los datos proporcionados por el XPS se puede acotar que el grado de utilización del FPGA Spartan 6 fue aproximadamente un 25% de su , como se observa en la Tabla V. Es decir, esta tarjeta brinda la posibilidad de agregar gran cantidad de *hardware* que aumente las funcionalidades del sistema.

TABLA II. IP CORES PRESENTES EN LA CAPA DE HARDWARE

Nombre	IP Core	Versión
Debug_Module	mdm	1.00.g
Microblaze_0	microblaze	7.30.a
mb_plb	plb_v46	1.04.a
Ilmb	lmb_v10	1.00.a
Dlmb	lmb_v10	1.00.a
LocalMemory_Cntlr_D	lmb_bram_if_cntlr	2.10.b
LocalMemory_Cntlr_I	lmb_bram_if_cntlr	2.10.b
lmb_bram	bram_block	1.00.a
RS232_Uart_1	xps_uart16550	3.00.a
clock_generator_0	clock_generator	4.00.a
DDR3_SDRAM	mpmc	6.00.a
proc_sys_reset_0	proc_sys_reset	2.00.a
Interrupt_Cntlr	xps_intc	2.01.a
timer_xilkernel	xps_timer	1.02.a
timer_clock	xps_timer	1.02.a
timer_sample	xps_timer	1.02.a
PWM	xps_timer	1.02.a
ADC	xps_deltasigma_adc	1.01.a
gpio_ventilador	xps_gpio	2.00.a

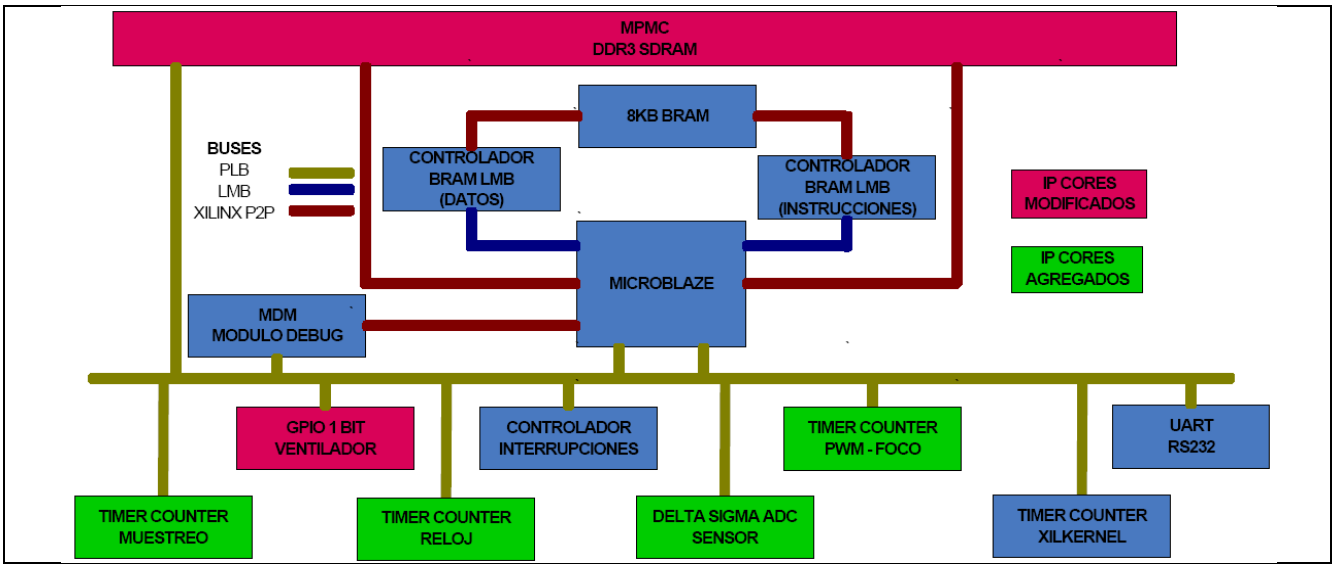

Fig. 13. Diagrama de Bloques del SoC

TABLA III. MAPA DE MEMORIA PARA MICROBLAZE

Nombre Módulo	Dirección Base	Dirección Superior
LocalMemory_Cntlr_D	0x00000000	0x00001fff
LocalMemory_Cntlr_I	0x00000000	0x00001fff
Debug_Module	0x84400000	0x8440ffff
RS232_Uart_1	0x83e00000	0x83e0ffff
DDR3_SDRAM	0x88000000	0x8fffffff
Interrupt_Cntlr	0x81800000	0x8180ffff
timer_xilkernel	0x83c00000	0x83c0ffff
timer_clock	0x83c40000	0x83c4ffff
timer_sample	0x83c20000	0x83c2ffff
PWM	0x83c60000	0x83c6ffff
ADC	0x80400000	0x8040ffff
Gpio_ventilador	0x81400000	0x8140ffff

TABLA IV. RESULTADOS DE LA SÍNTESIS DEL SOC

Resumen Síntesis XPS			
Reporte	Uso Flip Flops	Uso LUTs	Uso BRAMs
system	5403	5882	14
clock_generator_0_wrapper		1	
gpio_ventilador_wrapper	77	44	
adc_wrapper	217	183	
pwm_wrapper	361	344	
timer_sampler_wrapper	361	344	
timer_clock_wrapper	361	344	
timer_xilkernel_wrapper	361	344	
interrupt_cntlr_wrapper	192	162	
proc_sys_reset_0_wrapper	67	52	
ddr3_sdram_wrapper	918	1083	
rs232_uart_1_wrapper	370	423	

lmb_bram_wrapper			4
localmemory_cntrl_i_warppe	2	6	
localmemory_cntrl_d_warppe	2	6	
dlmb_wrapper	1	1	
ilmb_wrapper	1	1	
mb_plb_wrapper	160	423	
microblaze_0_wrapper	1833	2001	10
debug_module_wrapper	119	120	

TABLA V. RESUMEN DE UTILIZACIÓN DEL DISPOSITIVO FPGA

Resumen de Utilización del Dispositivo			
Utilización de bloques lógicos	**Uso**	**Disponible**	**Uso**
Number of Slice Registers	4,810	54,576	8%
Number used as Flip Flops	4,809		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used asAND/OR logics	1		
Number of Slice LUTs	5,455	27,288	19%
Number used as logic	4,343	27,288	15%
Number used as Memory	263	6,408	4%
Number occupied Slices	2,076	6,822	30%
Number of LUT Flip Flop pairs used	5,753		
Number with an unused Flip Flop	2,167	5,753	37%
Number with an unused LUT	298	5,753	5%
Number of fully used LUT-FF pairs	3,288	5,753	57%
Number of unique control sets	384		
Number of slice register sites lost to control set restrictions	1,500	54,576	2%
Number of bonded IOBs	58	296	19%
Number of LOCed IOBs	58	58	100 %
IOB Flip Flops	1		
Number of RAMB16BWERS	14	116	12%
Number of RAMB8BWERS	0	232	0%
Number of BUFIO2/BUFIO2_2CLKS	2	32	6%

2) Implementación de Software

En esta fase se realiza:

- La programación de la aplicación de *software* a través de un IDE, utilizando los *drivers* y librerías necesarios
- La compilación del código del programa, dado en lenguajes como C o C++, y
- Su almacenamiento en el núcleo del procesador.

El procedimiento de diseño empleando las herramientas SDK es el siguiente:

- Creación de un *Workspace* en SDK
- Importación de la Plataforma de *Hardware*
- Creación y Configuración del *Board Support Package* (BSP)
- Creación o Importación un proyecto de *Software*
- Descarga del archivo .bit (*bitstream*) de *hardware* al FPGA para emulación.

Los resultados en las capas Sistema Operativo y Aplicación fueron los siguientes:

a) Capa Sistema Operativo

El Board Support Package de esta capa contiene los *drivers* y librerías para el manejo de las funciones de *hardware*, y el RTOS *Xilkernel* para el trabajo con hilos, semáforos e interrupciones.

El sistema operativo es el xilkernel versión 5.00.a. En tanto que los drivers de los dispositivos periféricos del BSP se enlistan a continuación:

- ADC (dsadc)
- DDR3_SDRAM (mpmc)
- Debug_Module (uartlite)
- Interrup_Cntlr (intc)
- LocalMemory_Cntlr_D (bram)
- LocalMemory_Cntlr_I (bram)
- PWM (tmrctr)
- RS232_Uart_1 (uartns550)
- gpio_ventilador (gpio)
- timer_clock (tmrctr)
- timer_sample (tmrctr)
- timer_xilkernel (tmrctr).

La configuración de los módulos de *Xilkernel* se define a través de parámetros en el archivo *system.mss* del proyecto.

b) Capa Aplicación

En la Capa de Aplicación del sistema se desarrolló un proyecto de *software* que contiene los ficheros que se describen en la Tabla VI. El fichero *Shell.c* contiene una *Shell* CLI que ejecuta varios comandos para interactuar con el usuario. Además permite enlistar y ejecutar hilos previamente probados al sistema. En este caso se enlistó los hilos de control *On-Off* y PID a la aplicación que se ejecutan con los comandos *run 0* y *run 1* respectivamente.

D. Fase 4: Integración del Sistema

Con el *hardware y software* desarrollados, se procede a ensamblar el sistema completo para la realización de pruebas de laboratorio. El FPGA Spartan 6 de la tarjeta SP605 fue configurado con el *bitstream* de *hardware* generado en XPS y el código de *software* creado en SDK.

La implementación de un producto final suele incluir emuladores y prototipos rápidos antes de la fabricación en serie, para verificar las funciones de *hardware* y *software* del sistema. En este proyecto se llega únicamente hasta la emulación del producto sobre la tarjeta SP605.

La Fig. 14 muestra en un diagrama de bloques la disposición física de elementos de la sistema completo.

TABLA VI. DESCRIPCIÓN DEL PROYECTO DE LA CAPA DE APLICACIÓN

ARCHIVO	DESCRIPCION
shell.c	Este código implementa una Shell CLI que contiene comandos básicos para interactuar con el sistema operativo, es capaz de cargar y ejecutar nuevos hilos. Se le ha cargado los hilos para control *on-off* y PID.
clock.c	Implementa un hilo que sirve de reloj del sistema capaz de ser seteado. Se ejecuta siempre en paralelo al resto de hilos.
control_on_off.c	Contiene el hilo para control on-off de temperatura y su bucle de control. Se lo ejecuta a través del ingreso del comando "*run* 0" en la Shell, y termina con la interrupción externa del pulsador de la tarjeta SP605.
control_pid.c	Contiene el hilo para control PID de temperatura y su bucle de control. Se lo ejecuta a través del ingreso del comando "*run* 1" en la Shell, y termina con la interrupción externa del pulsador de la tarjeta SP605.
control_header.c	Contiene funciones para realizar *setup* de *hardware*, lectura del sensor a través del ADC, inicializar el *timer* de muestreo y para el ingreso de números enteros. Estas funciones son usadas en diversas partes del código por lo que conforman una librería extra, asociada a la aplicación a través del archivo de cabecera "control_header.h"
control_header.h	Archivo de cabecera que contiene los prototipos de las funciones del archivo control_header.c, además de definiciones generales de la aplicación.
Lscript.ld	*Linker Script* de la aplicación. Asocia todas las secciones del programa a la memoria DDR3 externa de la tarjeta SP605. Define un tamaño de pila (stack size) de 2Kb necesario para ejecutar los hilos de control.

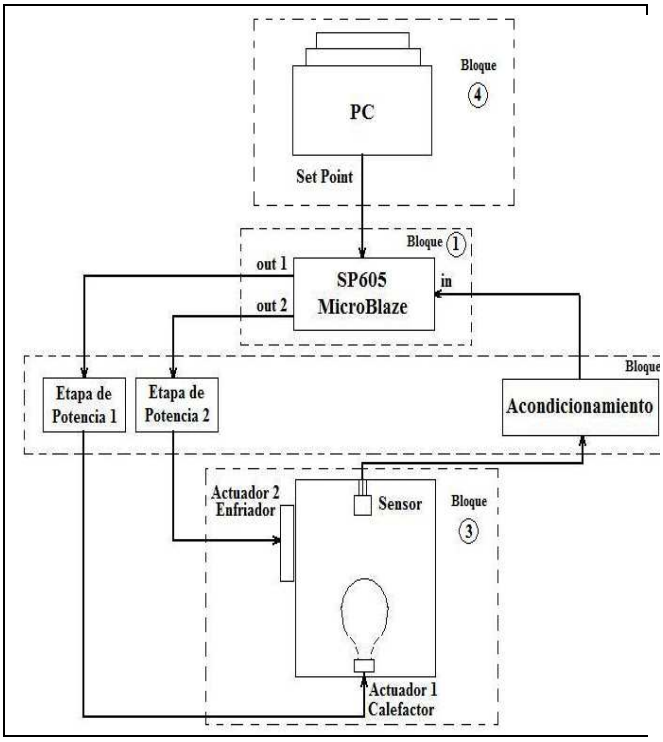

Fig. 14. Disposición Física de Elementos de la Aplicación

1) Descripción de Bloques

a) Bloque 1.- Este bloque corresponde a la plataforma de emulación SP605, sobre la cual se encuentra el SoC controlador que ha cumplido con las especificaciones del sistema.

b) Bloque 2.- Este bloque corresponde a la tarjeta de acondicionamiento de entradas y salidas, que contendrá los circuitos necesarios para trabajar con la planta de temperatura y la tarjeta SP605. Los circuitos externos implementados fueron:

- Dos etapas de potencia para los actuadores.
- Acondicionamiento para el sensor de la planta.
- *Hardware* externo necesario para trabajar con los *IP Cores* del SoC.

c) Bloque 3.- Corresponde a la planta.

d) Bloque 4.- Consiste de una terminal RS232 que sirva como interfaz de usuario, y permita el ingreso de comandos y la visualización de resultados. Se puede utilizar la consola del SDK o la *hiperterminal* de *Windows* para este propósito.

2) Ejecución del Sistema Integrado

Para ejecutar el sistema se debe configurar el FPGA con el *bitstream* de *hardware* y cargar el código del programa en la memoria. En la terminal RS232 se observará lo siguiente:

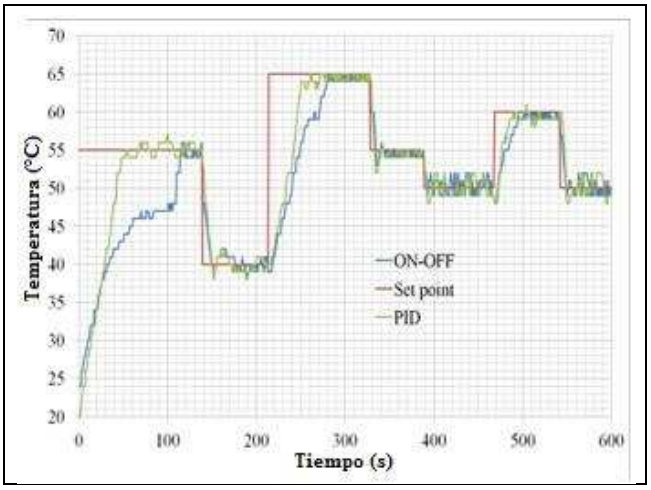

Fig. 15. Resultado control On-Off y control PID

SETUP: Plataforma de hardware inicializada correctamente.

Iniciando Xilkernel...

SHELL: Xilkernel inicializado

SHELL: Inicializando reloj...

RELOJ: Registrado gestor de interrupciones para el timer del reloj.

RELOJ: Configurando timer del reloj para generar interrupciones cada segundo ..

RELOJ: Interrupcion de reloj habilitada ...

shell> _

Para ejecutar las rutinas de control On-Off y PID se ingresa los comandos *run 0* y *run 1* respectivamente. Los resultados del control On-Off y del control PID se muestran en la Fig. 15.

Es importante señalar que si bien los resultados de ambos tipos de control son aceptables, se debería trabajar en la sintonización del controlador PID. Cabe recalcar que no se puso énfasis en esta parte puesto que el objetivo principal de este proyecto fue el crear un SoC bajo el enfoque de la metodología PBD. La aplicación de control de la planta de temperatura es solo un ejemplo de la capacidad y flexibilidad que se tiene al realizar el co-diseño de *hardware* y *software*.

V. CONCLUSIONES

El objetivo de crear un *System on Chip* (SoC) orientado a una aplicación de automatización y control que incluye un Sistema Operativo en Tiempo Real (RTOS), mediante el co-diseño de *hardware* y *software* fue cumplido a cabalidad.

La plataforma del sistema de control diseñado es fácilmente escalable para el desarrollo de nuevas aplicaciones de control.

La versatilidad que ofrece diseñar sobre *Field Programmable Gate Array* (FPGA) debe ser aprovechada en el diseño de sistemas que satisfagan las necesidades de la sociedad ecuatoriana.

Se demostró que el concepto de co-diseño y el uso FPGAs permiten al diseñador intervenir en la implementación de cada capa de un sistema embebido (*hardware*, sistema operativo, y aplicación). Esta característica permite realizar optimizaciones en cualquier capa y en cualquier momento del diseño, de forma económicamente fiable.

Por otro lado, se concluye que las herramientas de diseño con altas prestaciones, como las de Xilinx, constituyen un factor clave en la rápida implementación de SoCs. La ayuda que brindan al generar automáticamente la arquitectura de *hardware*, el mapa de direcciones, la asignación de pines, el *Board Support Package* (BSP) y los *test* de memorias y periféricos, disminuye enormemente la complejidad de un diseño.

Por último, se concluye que el presente proyecto fue el primer paso del Departamento de Eléctrica y Electrónica (DEEE) en la incursión hacia un nuevo modelo de negocios, que se basa en el desarrollo y exportación de tecnología SoC. Dentro de este ámbito se encuentran empresas desarrolladoras de *IP Cores*, arquitecturas y plataformas de *hardware*, RTOS, y soluciones basadas en FPGA.

VI. TRABAJOS FUTUROS

Utilizando como base el presente proyecto se presenta una lista de líneas de trabajo en el campo de diseño de SoCs que se recomiendan abordar. Estas son:

- Buses PCI con *IP Core PCIPLBv46 RC/EP* y *Bridge for PCI Express*
- *Networking* con *IP Core XPS LL TEMAC*.
- Sistema con dos procesadores utilizando *IP Cores XPS Mailbox* y *XPS Mutex*.
- Uso de protocolos de comunicación: Bus CAN, Ethernet Industriales, USB entre otros.
- Sistema que incluya *IP Cores* creados por el usuario.
- Depuración de *hardware* con *ChipScope Pro*
- Profundizar en el uso de Xilkernel
- Estudio de otros RTOS como Petalinux.
- Implementar técnicas de control adaptativo, difuso, o por redes neuronales en un SoC.
- Desarrollar control y monitoreo de procesos con HMI sobre un SoC.
- Estudio de la arquitectura AMBA de ARM.

RECONOCIMIENTO

Al ingeniero Byron Navas, por ser el mentor de este proyecto y la persona guía durante la etapa inicial.

BIBLIOGRAFÍA

[1] MARTIN, grant y CHANG, henry, *Winning the SoC Revolution - Experiences in Real Design*, Kluwer Academic Publisher, Estados Unidos 2003, 311 páginas.

[2] NAVAS, Byron, *Chips Diseñados en Ecuador,* Revista E-Ciencia ESPE, Edición 2, Diciembre 2009

[3] MARTIN, grant y CHANG, henry, *Surviving the SoC Revolution - A Guide to Platform – Based Design*, Kluwer Academic Publisher, Estados Unidos 1999, 235 páginas.

[4] XILINX, Inc., Getting Started with the Spartan-6 FPGA SP605 Embedded Kit, documento UG727 (v1.1), June 21, 2010.

[5] XILINX, Inc., *Hardware* and Demonstration Setup Guide, documento UG526 (v1.4), Septiembre 24, 2009.

[6] XILINX, Inc., EDK Concepts, Tools, and Techniques, documento UG683, 2009.