DISEÑO DE HARDWARE Y SOFTWARE DE SYSTEM ON CHIP EMPLEANDO TECNOLOGIA XILINX EDK

Julio César Cadena Salazar
Juan Gabriel Mollocana Lara

Director: Ing. Hugo Ortiz
Codirector: Ing. Vanessa Vargas

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
ESCUELA POLITÉCNICA DEL EJÉRCITO
Av. El Progreso s/n, Sangolquí, Ecuador

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. Dando un paso fundamental para el diseño de chips, facilitando e incentivando futuras investigaciones que permitan contribuir con el desarrollo tecnológico y económico del país. A la vez que brinda la posibilidad de abrir un nuevo modelo de negocios en este campo.

El objetivo principal de este proyecto fue crear un SoC orientado a una aplicación de automatización y control. El SoC diseñado tiene la funcionalidad de controlar una planta de temperatura, mediante la selección de dos técnicas de control: ON-OFF o PID. Durante las fases de diseño se utilizó la metodología *Platform Based Design* (PBD) y el procedimiento de co-diseño de *hardware* y *software* empleando herramientas de Xilinx. Cabe destacar que el co-diseño se desarrolló considerando un adecuado enfoque conceptual, arquitectural, y metodológico. El *hardware* del sistema conforma la integración de *IP Cores* gobernados por un procesador (*MicroBlaze*). En tanto que el *software* fue diseñado en base a un Sistema Operativo en Tiempo Real (RTOS - *Xilkernel*).

Palabras Claves: SoC, *IP Cores*, Sistemas Embebidos, *MicroBlaze*, RTOS.

ABSTRACT

The first System on Chip (SoC) designed, developed and emulated in Ecuador was obtained by the authors of this article. This gives a fundamental step in chip design. Motivating and easing future researches that will contribute to the economical and technological growth for the country and also opening the possibility of a new business model on this field.

The main objective of this project was to create a SoC oriented to an automation and control application. The SoC designed has the functionality of controlling the temperature of a process by selecting two control techniques: ON-OFF or PID. While developing this, it was employed the Platform Based Design (PBD) methodology and hardware/software co-design process, using Xilinx tools. Worth to mention is the fact that, the co-design was developed with proper approaches of concepts, architectures, and methodologies. The system hardware includes the IP Cores integration ruled by a processor (MicroBlaze). Whereas, the software was based on a Real Time Operating System (RTOS-Xilkernel).

Key Words: SoC, IP Cores, Embedded Systems, 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. Estos sistemas son diseñados sobre *Field Programmable Gate Arrays* (FPGA). Los FPGAs facilitan el desarrollo de nuevos productos gracias a su reprogramabilidad en el momento mismo del diseño. Además, permitirán que en un futuro cercano el Ecuador se convierta en desarrollador y exportador de tecnología. 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.

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.

El estado del arte o nivel más avanzado de la tecnología en el campo de diseño de *chips*, se ha logrado mediante la implementación de sistemas embebidos basados en *Systems on Chip* (SoC) sobre FPGA. 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. Esto brinda la ventaja de adaptar el diseño a la necesidad concreta del sistema a implementarse.

Lo que actualmente buscan los diseñadores de dispositivos electrónicos es integrar un mayor número de elementos en un simple *chip*, siempre y cuando no se incremente el tamaño del mismo, disminuya el tiempo de salida del producto al mercado (*time-to-market*) y aumente el tiempo del producto en el mercado (*time-in-market*).

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

En general los sistemas embebidos basados en SoCs están diseñados para hacer alguna tarea específica, en lugar de ser un computador de propósito general para múltiples tareas.

II. 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 PCB[1] con 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 (Figura 1).

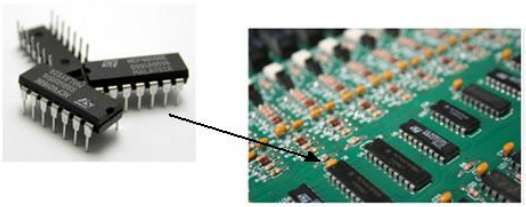


Figura 1. Componentes Reales

Con la utilización de *IP Cores*, los *chips* individuales que conformaban los componentes en *hardware* fueron reemplazados por componentes virtuales, que cumplen las mismas funciones (Figura 2).

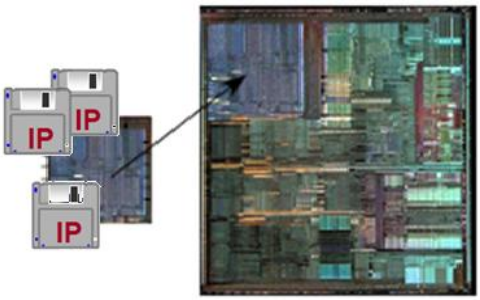


Figura 2. Componentes Virtuales

Esto brinda la ventaja de agrupar los componentes dentro de un mismo *chip* disminuyendo notablemente el tamaño y consumo de potencia de los productos ofrecidos.

En resumen, lo que en el pasado era un PCB hoy en día es un SoC. Adicionalmente lo que antes era un componente discreto en el presente es un *IP Core*.

En la Figura 3, se observa varios elementos que conforman un SoC, entre los que se destacan un

[1] ***Printed Circuit Board***: Tarjeta de Circuitos Impresos que contiene varios componentes discretos interconectados a través de rutas o pistas de material conductor.

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 incluir componentes analógicos y *opto/microelectronic mechanical system* (O/MEMS) [1]

Los SoC se basan en el diseño y reutilización de los bloques de propiedad intelectual *IP Cores* [2].

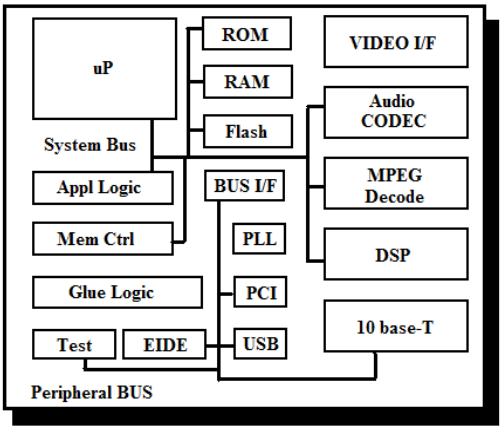


Figura 3. ***System on Chip***

III. IP CORES

Los *IP Cores* (*Intellectual Property 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, ya que utilizan módulos pre-probados.

Tipos de IP CORES

Existen 3 tipos de *IP Cores*:

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

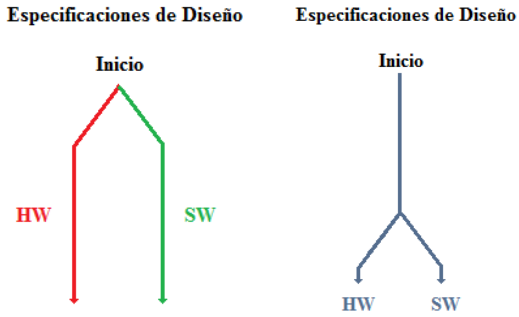
La Tabla 1 muestra un resumen de las características de cada tipo de *IP Core*.

TIPO	SOFT CORE	FIRM CORE	HARD CORE
NIVEL DE ABSTRACCION	RTL, *gate level*	*Gate level, layout*	*Layout*
DESCRIPCION	VHDL, *Verilog*	*Netlist*[2]	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

Tabla 1. Resumen de las Características de *IP Cores*

IV. 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 (Figura 4a). 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 (Figura 4b).



Flujo de Diseño Tradicional (a) Flujo de Co-Diseño (b)
Figura 4.

En el Co–diseño, el *hardware* y el *software* de un sistema embebido se desarrollan en paralelo, realizando constantes realimentaciones entre los

[2] ***Netlist:*** Representación en lenguaje de descripción de *hardware* de la conectividad de un circuito

equipos de diseño. El resultado es que cada parte puede tomar ventaja de lo que la otra puede hacer.

IV.1 FASES DEL CO-DISEÑO

De acuerdo a cada fase del proceso de Co-diseño, (Figura 5) propuestas por Martin y Chang en el libro *Surviving the SoC Revolution* de 1999, se detallarán los resultados obtenidos en este proyecto:

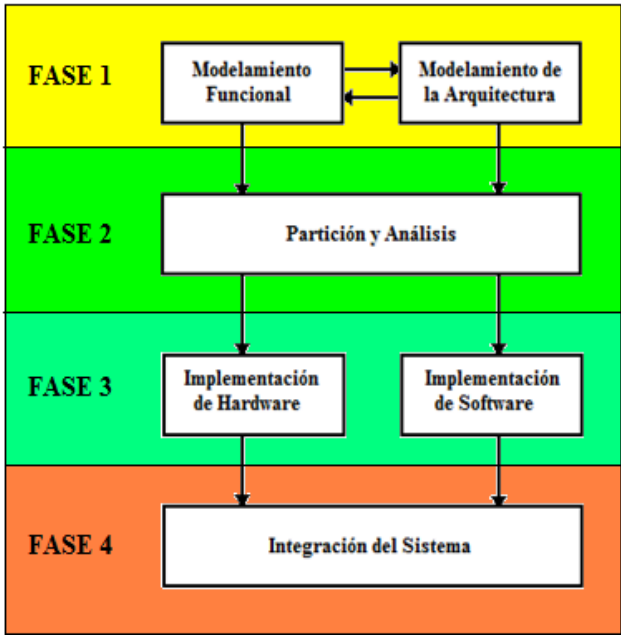


Figura 5. Fases del Co-Diseño HW/SW

FASE 1

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

La planta tiene las siguientes especificaciones técnicas: un rango de control de temperatura entre 40 y 65 grados centígrados, y su consumo es de 1.35 A (1.25A foco+ 0.1A ventilador).

El sistema que se desea obtener tiene el lazo de control que se muestra en la Figura 6.

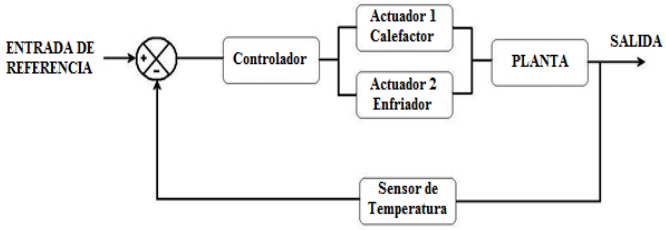


Figura 6. Esquema de Proceso de Control para Planta de Temperatura

Donde el elemento controlador es conformado por el SoC emulado sobre la tarjeta de desarrollo SP605. Este SoC, dadas las especificaciones de la planta necesita:

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 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.

Antes de continuar, es necesario mencionar las metodologías de diseño existentes y luego puntualizar el diseño basado en plataforma que es el de mayor interés.

METODOLOGIAS DE DISEÑO

Las metodologías de diseño primarias se dividen en tres segmentos:

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

Diseño Basado en Plataforma (PBD)

Una plataforma es "una gestión integrada, con un conjunto de características comunes, en el que un grupo o familia de productos se pueden construir. Una plataforma es un componente virtual (VC)"[3]. En definitiva, la plataforma es "un conjunto de equipos y *software* básico sobre el cual va a funcionar uno o varios sistemas a diseñar"[4] (Figura 7)

[3] MARTIN, grant y CHANG, henry, *Winning the SoC Revolution - Experiences in Real Design*.
[4] Universidad de Concepción, *Plataforma*, 2010

Figura 7. Ejemplo de Plataforma para diferentes equipos de electrónica de consumo

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

Una vez analizados los conceptos de plataforma se puede definir la plataforma utilizada que se detalla a continuación:

PLATAFORMA DE DESARROLLO XILINX SPARTAN-6 FPGA EMBEDDED KIT

Plataforma de *Hardware*

El Xilinx Spartan-6 FPGA Embedded Kit, es un conjunto de varios elementos para el desarrollo de SoCs, entre los que destaca la tarjeta de desarrollo 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 Figura 8).

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 futuros proyectos, como es el caso del sistema implementado.

Figura 8. Tarjeta de desarrollo SP605

Plataforma de *Software*

Esta plataforma 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)**, para el diseño de *hardware*, y ***Software Development Kit* (SDK)** para el diseño de *software*. Así como, toda 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].

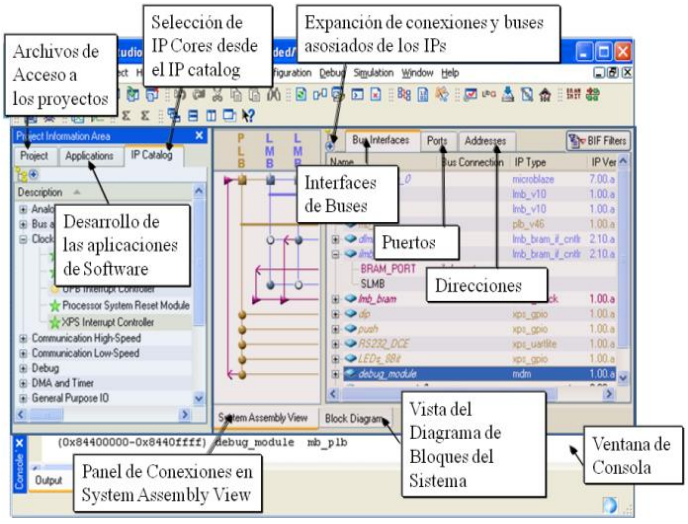


Figura 9. Vista general del XPS

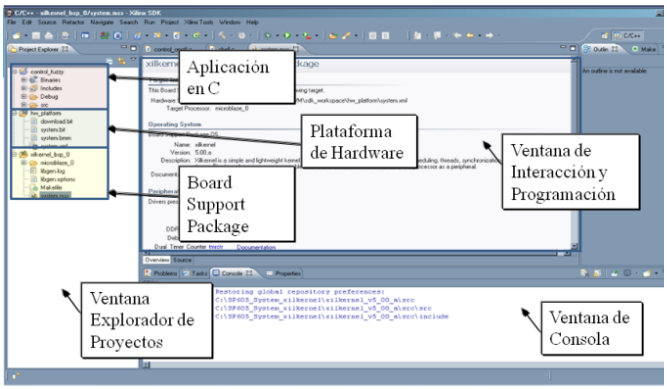


Figura 10. Vista general del SDK

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.

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.

Las herramientas de la plataforma de desarrollo XILINX SPARTAN-6 FPGA EMBEDDED KIT 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.

El estándar *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 (Figura 11).

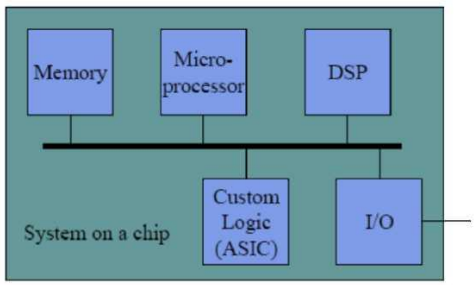


Figura 11. SoC basado en topología BUS

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.

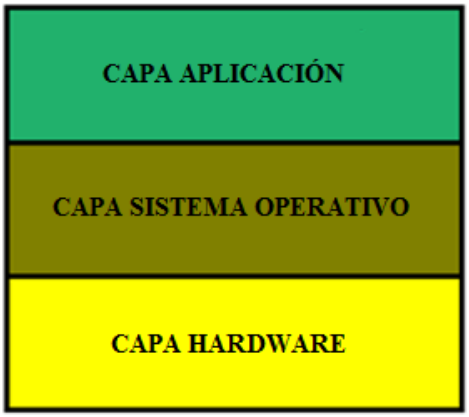


Figura 12. Vista en Capas del Diseño

- **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 cuanto a la Capa de Aplicación, las tareas que lleva a cabo a través de rutinas de *software* en lenguaje C son:

- 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*.

FASE 3

***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 ahí su nombre. 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.

CAPA DE *HARDWARE*

El procedimiento de diseño 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 (Figura 13) expresado a partir de la vista RTL que ofrece el XPS. En este diagrama constan los *IP Cores* listados en la Figura 14. 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 Figura 15.

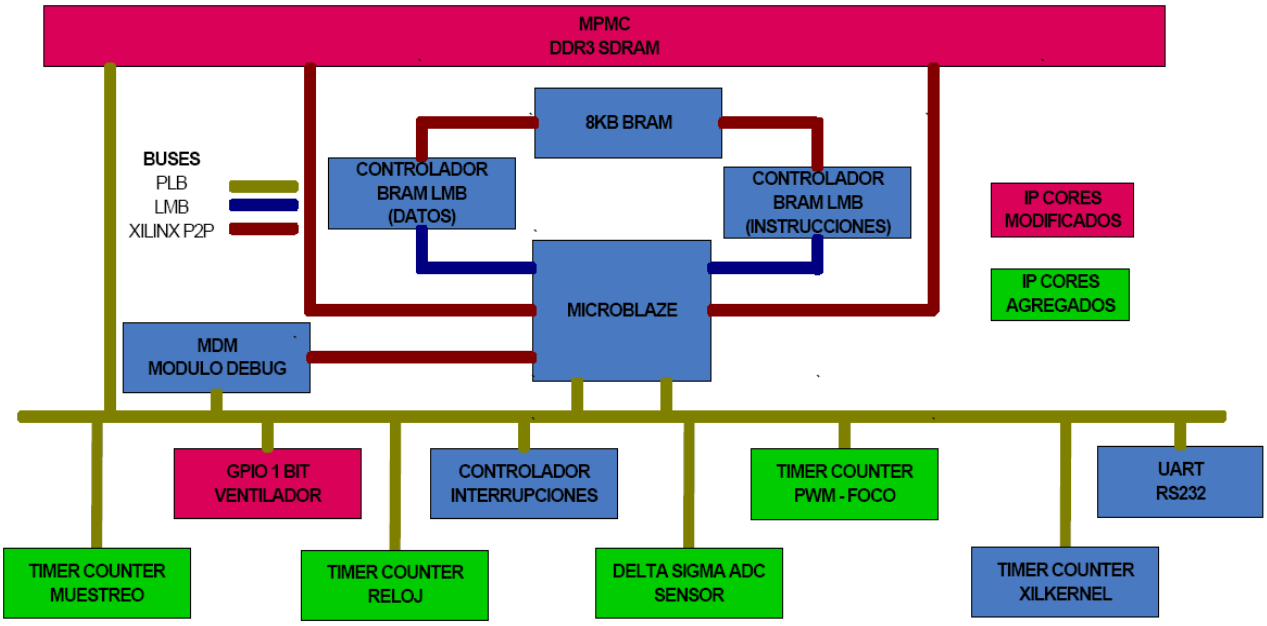


Figura 13. Diagrama de Bloques del SoC

IP Cores presentes en el diseño

Nombre Módulo	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

Figura 14. IP Cores presentes en la capa de *hardware*

Mapa de Memoria para microblaze_0

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

Figura 15. Mapa de memoria para MicroBlaze

Los resultados de la síntesis de este *hardware* se muestran en la Figura 16 y los resultados de la utilización del FPGA en la Figura 17, estos datos son proporcionados por el XPS. Se puede apreciar que se ha utilizado aproximadamente un 25% de la capacidad del FPGA Spartan 6. Esto brinda la posibilidad de agregar gran cantidad de *hardware* que aumente las funcionalidades del sistema.

XPS Synthesis Summary				
Report	**Generated**	**Flip Flops Used**	**LUTs Used**	**BRAMS Used**
system	mar 18. oct 21:40:17 2011	5403	5882	14
clock_generator_0_wrapper	mar 18. oct 21:39:25 2011		1	
gpio_ventilador_wrapper	mar 18. oct 20:35:26 2011	77	44	
adc_wrapper	mar 18. oct 20:35:16 2011	217	183	
pwm_wrapper	mar 18. oct 20:34:59 2011	361	344	
timer_sample_wrapper	mar 18. oct 20:34:45 2011	361	344	
timer_clock_wrapper	mar 18. oct 20:34:28 2011	361	344	
timer_xilkernel_wrapper	mar 18. oct 20:34:15 2011	361	344	
interrupt_cntlr_wrapper	mar 18. oct 20:34:01 2011	192	162	
proc_sys_reset_0_wrapper	mar 18. oct 20:33:50 2011	67	52	
ddr3_sdram_wrapper	mar 18. oct 20:33:43 2011	918	1083	
rs232_uart_1_wrapper	mar 18. oct 20:32:07 2011	370	423	
lmb_bram_wrapper	mar 18. oct 20:31:48 2011			4
localmemory_cntlr_i_wrapper	mar 18. oct 20:31:39 2011	2	6	
localmemory_cntlr_d_wrapper	mar 18. oct 20:31:30 2011	2	6	
dlmb_wrapper	mar 18. oct 20:31:22 2011	1	1	
ilmb_wrapper	mar 18. oct 20:31:15 2011	1	1	
mb_plb_wrapper	mar 18. oct 20:31:10 2011	160	423	
microblaze_0_wrapper	mar 18. oct 20:30:49 2011	1833	2001	10
debug_module_wrapper	mar 18. oct 20:29:45 2011	119	120	

Figura 16.Resultados de la Síntesis del SoC

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
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 as AND/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 of 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%	

Figura 17. Resumen de Utilización del dispositivo FPGA

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. Finaliza con 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 son los siguientes:

CAPA SISTEMA OPERATIVO

El BSP 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 (Figura 18). La configuración de *Xilkernel* se define en el archivo *system.mss* del proyecto, y es la siguiente:

```
BEGIN OS
PARAMETER     OS_NAME = xilkernel
PARAMETER     OS_VER = 5.00.a
PARAMETER     PROC_INSTANCE = microblaze_0
PARAMETER     STDIN = RS232_Uart_1
PARAMETER     STDOUT = RS232_Uart_1
PARAMETER     SYSTMR_SPEC = true
PARAMETER     SYSTMR_DEV = timer_xilkernel
PARAMETER     SYSINTC_SPEC = Interrupt_Cntlr
PARAMETER     SYSTMR_INTERVAL = 100
PARAMETER     PTHREAD_STACK_SIZE = 20000
PARAMETER     SCHED_TYPE = SCHED_PRIO
PARAMETER     N_PRIO = 6
PARAMETER     CONFIG_SEMA = true
PARAMETER     STATIC_PTHREAD_TABLE =
              ((shell_main,1))

END
```

Estos parámetros definen las configuraciones de los módulos de *Xilkernel*. Por ejemplo, el módulo *scheduling* trabajará en base a 6 posibles niveles de prioridad de un hilo. Para más información revisar la hoja de datos de *Xilkernel* disponible en SDK.

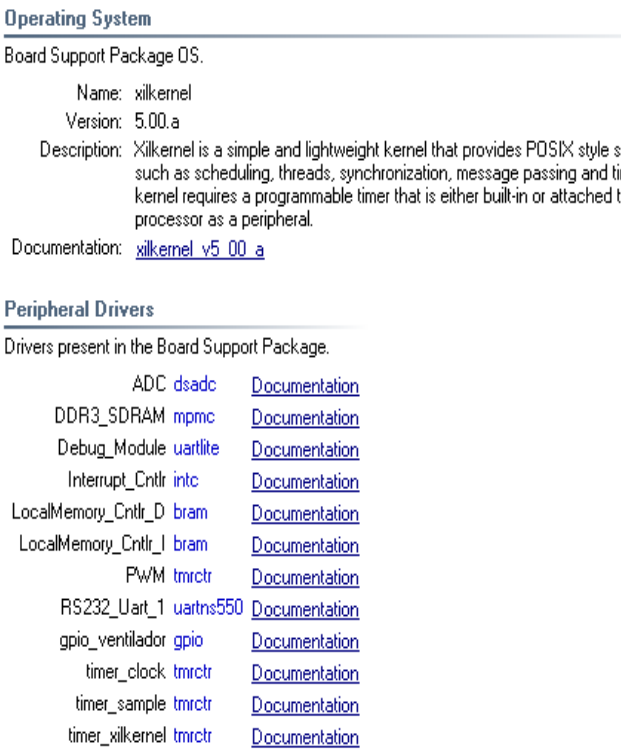


Figura 18. Sistema Operativo y *Drivers* del BSP.

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 2. 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.

DESCRIPCIÓN	ARCHIVO
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	Este archivo 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	Este archivo 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	Este archivo 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	Este archivo 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.

Tabla 2. Descripción del proyecto de la capa de aplicación

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. La implementación de un producto puede incluye emuladores y prototipos rápidos para verificar las funciones de *hardware* y *software*. En este proyecto se llega hasta la emulación del producto sobre la tarjeta SP605.

La Figura 19 muestra en un diagrama de bloques la disposición física de elementos de la sistema completo.

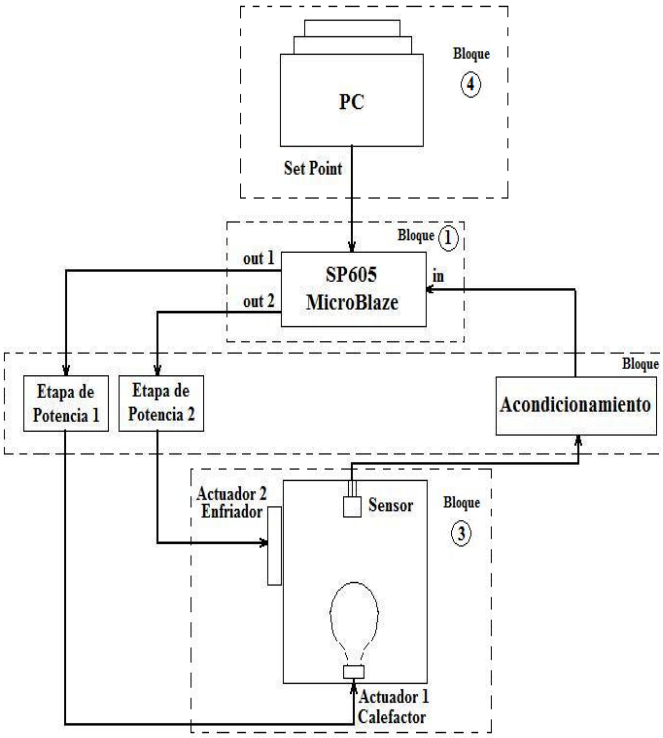


Figura 19. Disposición Física de Elementos de la Aplicación

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.

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 que se implementados son:

- 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.

BLOQUE 3

Corresponde a la planta que tiene las características antes mencionadas.

BLOQUE 4

Este bloque contiene 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.

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:

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 se muestran en la Figura 20 y los del control PID en la Figura21.

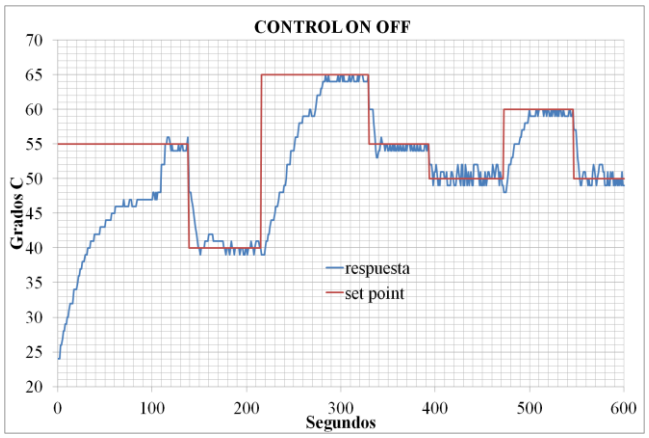


Figura 20. Resultado control On-Off

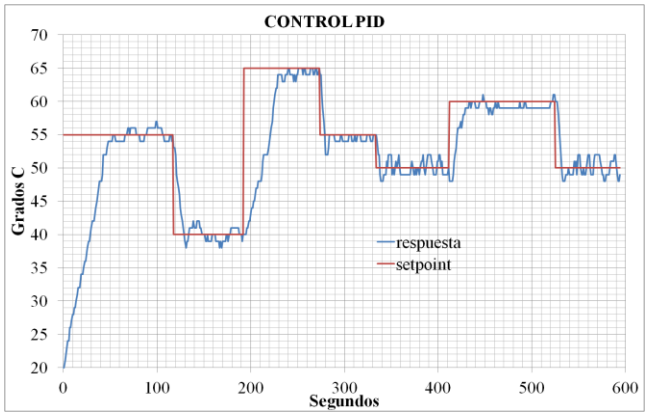


Figura 21. Resultado control PID

Se puede observar que la mejor respuesta se la obtiene en el control PID, ya que presenta menor tiempo de establecimiento (50 segundos) y sobre impulso (1 grado C). Sin embargo, el control On-Off también presenta un tiempo de establecimiento (110 segundos) y sobre impulso (2 grados C) aceptable. En este caso ambas técnicas de control logran el seguimiento del *set point*.

Cabe recalcar que no se realizó una sintonización óptima del controlador PID mediante los métodos comunes, ya 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, mediante el co-diseño de *hardware* y *software*, empleando herramientas de Xilinx, fue cumplido a cabalidad al implementar sobre la tarjeta de desarrollo SP605, un SoC que incluye un Sistema Operativo en Tiempo Real (RTOS) para un sistema de control de temperatura.

Se concluye que el sistema de control diseñado es fácilmente escalable para el desarrollo de nuevas aplicaciones de control, debido a la versatilidad que ofrece diseñar sobre *Field Programmable Gate Array* (FPGA).

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 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

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

- Bus CAN con *IP Core XPS_CAN_Controller*.
- Protocolos Ethernet Industriales con *IP Core XPS Ethernet Lite MAC*.
- *Networking* con *IP Core XPS LL TEMAC*.
- Comunicación USB con *IP Cores XPS USB Host Controller* y *XPS USB2 Peripheral*.
- Bus PCI con *IP Core PCIPLBv46 RC/EP Bridge for PCI Express*.

- Sistema con dos procesadores utilizando *IP Cores XPS Mailbox* y *XPS Mutex*.
- Sistema que incluya *IP Cores* creados por el usuario, lo que incluye la creación de *drivers*.
- 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.

VII. BIBLIOGRAFIA

[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.

VIII. SOBRE LOS AUTORES

Julio César Cadena Salazar, nace en la ciudad de Guayaquil en el año 1986. En el año 2004, obtiene el título de Bachiller Técnico Industrial especialidad en Electrónica en el Instituto Tecnológico Superior Central Técnico. Egresado de la Escuela Politécnica del Ejército en el año 2010 de Ingeniería Electrónica especialidad en Automatización y Control.

Juan Gabriel Mollocana Lara, nace en la ciudad de Quito en el año 1987. En el año 2005, obtiene el título de Bachiller Físico Matemático en el colegio María Eufrasia. Egresado de la Escuela Politécnica del Ejército en el año 2010 de Ingeniería Electrónica especialidad en Automatización y Control.