El papel del Hardware *copyleft* en la Enseñanza de Sistemas Embebidos

Carlos I. Camargo Bareño
Universidad Nacional de Colombia,
Email: cicamargoba@unal.edu.co

***Abstract*—El gran avance de las técnicas de fabricación de Circuitos Integrados ha permitido que los sistemas embebidos sean parte fundamental de nuestras vidas, aún sin darnos cuenta diariamente interactuamos con decenas de ellos. Esto unido a la disponibilidad de herramientas software de desarrollo gratuitas abre grandes posibilidades comerciales para paises en vía de desarrollo ya que no son necesarias grandes inversiones de capital para la concepción, diseño, y fabricación de estos sistemas. Sin embargo, en la actualidad muy pocas universidades ofrecen cursos que permitan crear las habilidades necesarias para la realización de un producto comercializable, lo que se traduce en un abandono de la producción local y el aumento de la dependencia con la industria manufacturera asiática. por otro lado, las herramientas utilizadas en la actualidad (tanto SW como HW) proporcionan un nivel de abstracción relativamente alto impidiendo que el estudiante entienda el funcionamiento global de un sistema digital, lo que le impide generar habilidades (especificamente las relacionadas con la concepción e implementación) necesarias para realizar el proceso completo. En este artículo se presenta una metodología para la enseñanza de diseño de sistemas embebidos utilizando herramientas hardware y software abiertas que ayuda a resolver los problemas mencionados anteriormente.**

***Index Terms*—Sistemas Embebidos, educación en ingeniería, hardware copyleft.**

I. Introducción

El mercado de los sistemas embebidos continúa en aumento y su campo de acción se ha extendido en casi todas las actividades humanas. Según BBC, inc. el mercado para el software embebido creció de $1.6 billones a $3.5 billones en 2009, con una tasa de crecimiento anual (AAGR) del 16%, mientras la tasa de crecimiento para las tarjetas embebidas es del 10%; según *Venture Development Corporation (VDC)* más de un billón de dispositivos embebidos fueron vendidos en el 2004,. De acuerdo con VDC el porcentaje de dispositivos basados en sistemas operativos comerciales tiende a disminuir del 43.1% en 2001 a 37.1% en 2004, esta tendencia se debe a la utilización de herramientas de libre distribución GNU/Linux [1].

En la actualidad estamos presenciando una tendencia global a delegar las tareas de manufactura de sistemas digitales a paises asiáticos, donde la mano de obra calificada es abundante y barata; se presentan casos donde los creadores de una determinada tecnología no la desarrollan y dejan que estos paises se beneficien de sus descubrimientos [2] reduciendo de forma considerable la producción. Esta situación se agrava a medida que las grandes empresas manufactureras asiáticas como Foxconn capturan la producción de los grandes diseñadores como Apple, Nokia, DELL, HP y Microsoft, lo que genera el cierre de empresas manufactureras a lo largo del mundo, con la consecuencia de pérdida de empleo masivo. En la actualidad según *Bureau of Labor Statistics* y *Thomson Financial Extel Company Report* Foxconn emplea a más personas que Apple, Dell, Micorsoft, HP, Intel y Sony combinados; esta situación es más grave en paises en vía de desarrollo, donde no existe la plataforma tecnológica para diseñar dispositivos digitales, y son importadores de tecnología sin la capacidad de generar productos que satisfagan necesidades locales. Lo anterior lleva a preguntarse por la función y la situación de un profesional en áreas afines a la ingeniería electrónica en países donde no existe la capacidad de concepción y diseño.

La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en areas afines al desarrollo de dispositivos digitales [3] ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [2] generando desempleo en las personas menos capacitadas. Según John Hall presidente y CEO de Linux International " algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico" [3]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más.

Según el ex-director ejecutivo de Intel Andy Grove [2] la solución está en hacer de la creación de empleo la política económica gubernamental más importante y hacer que las demás giren en torno a ella. Además, es necesario volver a la producción interna con el fín de generar nuevos empleos, y volver a adoptar medidas que protejan la producción interna de los productos asiáticos. Sin embargo, para lograrlo es necesario crear en los profesionales las habilidades para implementar productos comercializables.

En este artículo presentamos un programa académico basado en la utilización de software y hardware libre para el área de electrónica digital que desarrolla las habilidades nece-

sarias para Concebir, Diseñar Implementar y operar sistemas digitales.

A. Flujo de diseño de sistemas embebidos

Los Sistemas Embebidos son sistemas heterogéneos que contienen componentes Software (microcontroladore, microprocesadores y DSPs) y Hardware (funciones implementadas en Dispositivos lógicos programables PLDs); por este motivo, es necesario adquirir habilidades en la utilización de lenguajes de programación como C o C++ para implementar las funciones software y Verilog o VHDL para la implementación de las tareas hardware; adicionalmente, deben conocer las diferentes formas de comunicación entre estos dos tipos de funciones. Aunque en el mercado existen herramientas que permiten la entrada de diseño utilizando lenguajes de alto nivel como *SystemC* o *SpecC* y proporcionan el código para implementar las tareas software, hardware y su interfaz de comunicación; no es recomendable utilizarlas en el ciclo de formación básico ya que impide que se conozca el flujo de diseño completo, suministrando un nivel de abstracción en el cual no es necesario conocer la arquitectura de la plataforma utilizada para la implementación.

En la figura 1 se muestran los conceptos que deben dominar los diseñadores de sistemas embebidos, y las tareas que deben realizarse para la concepción, diseño e implementación de un sistema embebido. En gran parte de los programas académicos se estudian únicamente los temas relacionados con la concepción y diseño centrándose en las especificaciones funcionales del sistema, utilizando herramientas comerciales o COTS (Commercial off-the-shelf) para su implementación. Esta combinación genera dependencia e impide la generación de habilidades necesarias para implementar un sistema digital teniendo en cuenta restricciones económicas, físicas, eléctricas, ergonómicas, comerciales, etc. Nuestra propuesta se basa en la utilización de herramientas abiertas tanto hardware como software que permiten recorrer todo el proceso de concepción, diseño e implementación y obtener un entendimiento integral del proceso sin generar dependencia a productos comerciales.

1) Dominios de Diseño y Niveles de abstracción: Existen tres dominios en los que se puede describir un sistema digital [5] *Funcional*: Describe el comportamiento funcional y temporal del sistema *Estructural*: Describe su composición a partir de bloques básicos y *Físico* relacionado con la estructura física del sistema a nivel de circuito integrado o placa de circuito impreso [6]. Cada uno de estos dominios puede ser descrito utilizando diferentes niveles de abstracción; un nivel de abstracción alto permite el uso de lenguajes de alto nivel facilitando la entrada de diseño al extraer la funcionalidad de la parte física; por otro lado, los niveles de abstracción bajo utilizan bloques constructores elementales.

En el mercado existen herramientas que permiten entradas de diseño a nivel funcional utilizando especificaciones y algorítmos y de forma automática y optimizada generan representaciones en diferentes niveles de abstracción en los dominios estructural y físico. Es decir, a partir de las especificaciones

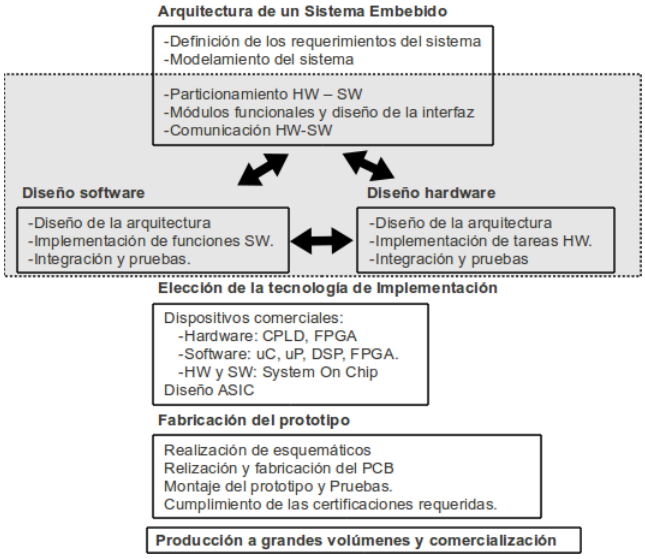

Fig. 1. Educación de sistemas embebidos. Tomada de: [4] y modificada

y algoritmos que indican el funcionamiento del sistema pueden generar archivos para la fabricación de un circuito integrado o archivos de configuración/programación que pueden ser utilizados en dispositivos comerciales. Desde el punto de vista comercial esto es muy útil ya que permite reducir el tiempo de diseño y los costos asociados. Sin embargo, presentan los inconvenientes mencionados anteriormente y por lo general son muy costosas.

II. Método para la enseñanza de Sistemas Embebidos

Basándose en la metodología de diseño para sistemas embebidos [7], en los dominios de diseño y niveles de abstracción de Gajski-Kuhn, se realizó una división de temas que busca crear habilidades de forma gradual e incremental. En la Figura 4 podemos observar esta división y las herramientas que se utilizarán en cada curso, como herramienta de desarrollo hardware se utilizará una plataforma que proporcione los archivos y documentos necesarios para replicarla, modificarla y pueda ser utilizada como base de desarrollos comerciales.

A. SIE: Plataforma abierta para el desarrollo de sistemas embebidos

En el mercado existe una gran variedad de plataformas que pueden ser utilizadas en el estudio de sistemas embebidos, sin embargo, no todas son adecuadas para la implementación del método que proponemos ya que se requiere: acceso a los esquemáticos y a los archivos de fabricación del PCB con posibilidad de modificación; acceso a la documentación completa del proceso de fabricación; acceso a la cadena de producción; utilización de herramientas abiertas para su programación; un PLD para la implementación de tareas HW; un procesador para la implementación de tareas SW; un canal de comunicación entre el procesador y el PLD; y una comunidad que desarrolle aplicaciones para dicha plataforma y que proporcione medios

para el intercambio de información a través de listas de correo y wikis.

Después de una búsqueda minuciosa no se encontraron plataformas que cumplieran con estas condiciones, en especial con las relacionadas con el proceso de diseño y de producción, esto es normal, ya que la mayoría de las empresas no quieren que se fabriquen sus plataformas y los proyectos individuales no poseen la infraestructura necesaria para la producción masiva. Por este motivo, se decidió crear una plataforma que cumpliera con los requerimientos (plataforma *SIE*), para ello se buscaron proyectos similares que permitieran su creación y que el producto creado sea una extensión de dicho proyecto. El proyecto Qi-Hardware [8] busca definir el concepto *copyleft hardware* basándose en el movimiento de software libre y código abierto (FOSS [9]) y proporciona un enlace con la industria manufacturera asiática.

1) Hardware copyleft: El proyecto SIE [10] fué creado para satisfacer las necesidades de los desarrolladores de hardware permitiendo la creación de aplicaciones comerciales bajo la licencia Creative Commons BY - SA [11] la que permite la distribución y modificación del diseño (incluso para aplicaciones comerciales), con el único requisito de que los productos derivados deben tener la misma licencia y deben dar crédito al autor del trabajo original. Lo que constituye la base de los productos *hardware copyleft*.

Al ser inspirado en el movimiento FOSS, los dispositivos *hardware copyleft* comparten la misma filosofía [9], y son el complemento perfecto del software libre. Para que un dispositivo HW sea reproducible y modificable es necesario: suministrar los archivos necesarios para la fabricación, es decir, los esquemáticos y los archivos de la placa de circuito impreso (preferiblemente para herramientas abiertas como Kicad o geda); la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; el código fuente de: el programa que inicializa la plataforma (*bootloader*), la herramienta que carga dicho programa en la memoria no volátil (*usbboot*), el sistema de archivos y aplicaciones (*openwrt*); documentación completa que indique como fué diseñada, construida, como utilizarla, desarrollar aplicaciones y tutoriales que expliquen el funcionamiento de los diferentes componentes. (esto puede ser descargado de [12] y [13]). Adicionalmente, se debe contar con la posibilidad de fabricación y montaje, lo que constituye la principal diferencia entre el software y el hardware libre.

2) Arquitectura: La Figura 2 muestra el diagrama de bloques de la plataforma SIE, en ella encontramos un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, controlar un LCD a color de 3 pulgadas, 2 entradas y salidas de audio stereo, 2 entradas análogas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor análogo digital de 8 canales. Existen tres canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (eliminando la necesidad de cables de programación); otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA y un puerto serial utilizado para depuración de softcores implementados en la FPGA. El procesador utilizado es un Ingenic JZ4725 (MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

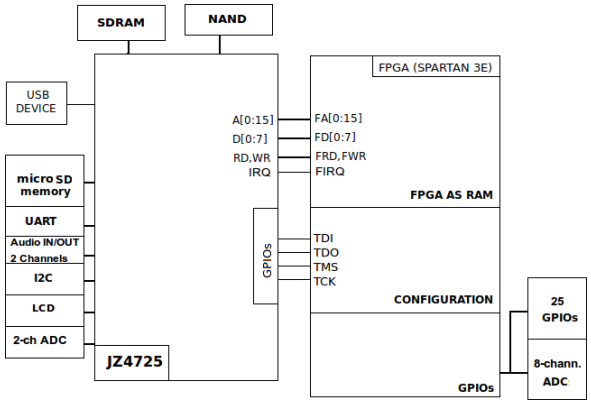

Fig. 2. Estructura de la plataforma de desarrollo SIE

3) Comunicaciones: SIE proporciona un canal de comunicación y alimentación a través del puerto USB-device, y es configurado para ser utilizado como una interfaz de red (*usb0*), permitiendo la transferencia de archivos y ejecución de una consola remota utilizando el protocolo ssh; este canal de comunicación también se utiliza para programar la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB.

4) Especificaciones físicas: Las dimensiones de SIE son 8cm de largo, 8 cm de ancho, 1cm de altura; su placa de circuito impreso es de dos capas, utiliza líneas de 8 mils, vías de 12 mils de diámetro, los componentes se encuentran en una sola capa y son TQFP o SMD, no posee componentes BGA o QFN lo que facilita el montaje manual. Todo esto hace que sea posible la reproducción y modificación de esta plataforma a un precio muy bajo. El costo de fabricación de esta tarjeta se estima en 70 usd para 50 unidades. La figura 3 muestra la apariencia de la plataforma de desarrollo SIE.

B. Curso básico

Para el curso básico se trabajará la mayor parte de los niveles de abstracción del dominio funcional; partiendo de unas especificaciones funcionales se generará un modelo del sistema utilizando algoritmos que describan el comportamiento de las diferentes tareas que implementan el sistema (bloques funcionales). Estos bloques serán implementados, en dispositivos lógicos programables como FPGAs o CPLDs. A partir de estos algoritmos se identifican las operaciones básicas aritméticas y lógicas que modifican los datos asociados a cada función para generar el camino de datos (*datapath*;

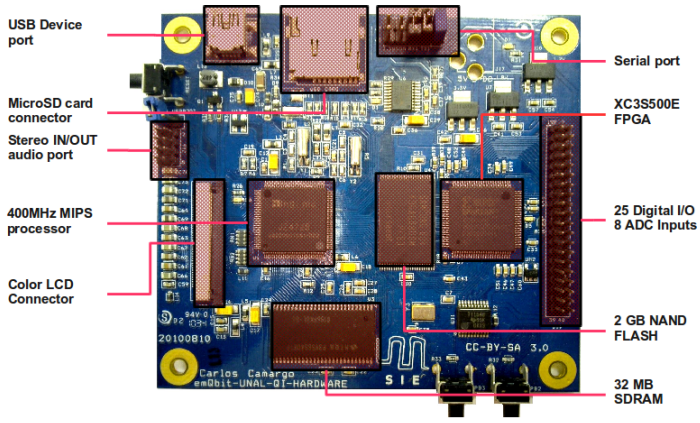

Fig. 3. Plataforma de desarrollo SIE

el datapath proporciona señales que controlan el instante en el que se ejecutan la operaciónes soportadas, dichas señales deben ser generadas por un módulo diseñado para implementar el algoritmo deseado; estos dos módulos se implementaran con bloques lógicos básicos y máquinas de estados finitos utilizando lenguaje de descripción de hardware (VHDl, verilog estándard). Estas descripciones son la entrada a herramientas que realizarán la transición al dominio estructural generando las compuertas lógicas, flip-Flops y las interconexiones que implementen la funcionalidad requerida. Durante el proceso es necesario realizar simulaciones (utilizando *icarus verilog* y *ghdl*) que permitan comprobar el cumplimiento de las especificaciones iniciales, si no se cumple alguna de ellas se debe volver a repetir el proceso.

Durante el desarrollo del primer curso se estudiaran los conceptos básicos de los sistemas digitales como sistemas numéricos, operaciones aritméticas y lógicas, lógica combinatoria y secuencial. Se utilizaran lenguajes de descripción de hardware como VHDL y verilog como entrada de diseño a herramientas que realizan la síntesis digital. Para evitar crear dependencia, se enseñará la forma de adecuada de implementar código re-utilizable que no utilice componentes específicos de un determinado fabricante.

1) Diseño de aplicaciones utilizando HDL y PLDs: Para generar las habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, se realizarán prácticas sencillas que ayuden al estudiante a entender los mecanismos de implementación de tareas HW en un dispositivo lógico programable utilizando la plataforma de desarrollo SIE; como puede verse en la figura 2 la FPGA solo controla un conversor análogo digital serial de 8 canales, y proporciona 25 GPIOs, esto se hizo de forma intencional para que los estudiantes se vean forzados a realizar las conexiones eléctricas de los dispositivos externos a sus aplicaciones; las plataformas comerciales proporcionan una gran variedad de dispositivos conectados a los PLDs, lo que no es muy recomendable ya que el estudiante no aprende a leer la hoja de especificaciones del fabricante de un determinado dispositivo para determinar su forma de conexión, y/o las condiciones que se deben tener en cuenta para su correcto funcionamiento; afectando la generación de habilidades necesarias para la elección de componentes, realización y lectura de esquemáticos y diseño de layouts.

El procesador de la plataforma SIE será utilizado como camino de configuración de la FPGA; el archivo de configuración será descargado al sistema de archivos de la plataforma SIE utilizando el protocolo *ssh* y la interfaz de red USB; un programa en espacio de usuario se encarga de configurar la FPGA con el archivo deseado. Adicionalmente, existe una aplicación basada en el proyecto *urjtag* ejecutándose en el procesador que permite la generación de vectores de prueba y recolección de resultados utilizando el puerto JTAG [14], lo que permite que el estudiante pueda probar su circuito a baja frecuencia sin instrumentos de medición adicionales.

C. Arquitectura de Computadores

En este curso se trabajará en el dominio estructural comenzando desde los componentes básicos de una CPU hasta llegar a la arquitectura de un sistema sobre silicio (SoC), esto con el fín de conocer y entender la arquitectura y funcionamiento de los dispositivos en los que se ejecutan las tareas software. Se utililizarán periféricos conectados a través de buses a la CPU para implementar tareas hardware. Se realizará la implementación de un Sistema sobre silicio (SoC) y se trabajará con herramientas de libre distribución (cadena de herramientas GNU [1]) para programar aplicaciones que involucren el uso de tareas HW y SW.

1) Arquitectura de una CPU y tareas SW: Utilizando procesadores *softcore* se estudiarán los componentes básicos de la CPU, el camino de datos: banco de registros, bloques aritméticos, lógicos y buses internos, se analizarán las diferentes instrucciones que proporciona la arquitectura bajo estudio y se analizará el funcionamiento de la máquina de control, identificando los componentes que permiten el almacenamiento y ejecución secuencial de instrucciones definidas por el usuario. En este punto se introducirán los conceptos de llamado a funciones, atención de interrupciones, direccionamiento directo e indirecto y acceso a memoria externa.

Para este estudio, se utilizarán procesadores implementados en lenguajes de descripción de hardware que cuenten con herramientas de programación de bajo (assembler) y alto nivel (C, C++), con el fin de realizar simulaciones, aplicaciones y modificaciones. Al finalizar el curso se pretende que el estudiante entendienda las diferencias entre tareas software y tareas hardware y podrá realizar experimentos que le permitan comparar las características de ambas implementaciones. En la actualidad se está trabajando con los SoC *plasma* basado en un procesador MIPS, *MICO 32* de lattice, y *openrisc* de OpenCores, implementados en VHDL o Verilog.

Se utilizarán los periféricos para la implementación de tareas HW y se estudiaran las diferentes formas que existen para comunicarse con las tareas SW (buses, interrupciones, polling) presentando los criterios de selección para la implementación de tareas SW y HW. Se introducirá el concepto de mapa de memoria, decodificador de direcciones, memorias volátiles

(ejecución de programas y de uso general) y memorias no volátiles (almacenamiento de programas). Se introducirá el concepto de *bootloader* y su uso en la carga de aplicaciones a las memorias volátiles internas y externas del SoC. Y finalmente se mostraran los diferentes métodos existentes para programar las memorias no volátiles.

Utilizando la cadena de herramientas GNU, se realizará el flujo de desarrollo para tareas SW desde la entrada de diseño utilizando el lenguaje C, hasta la generación del archivo binario que contiene las instrucciones (para la CPU en estudio) que implementan dicha tarea. Utilizando herramientas propias se generarán los archivos para inicializar la memoria de programa (implementada en la FPGA) con estas instrucciones.

2) Diseño de aplicaciones que involucren co-diseño HW-SW: Tanto la CPU, los periféricos, la memoria ram y la memoria de programa estarán implementados en la FPGA de SIE. Durante todo el periodo académico se desarrollará un proyecto de complejidad media que involucre el uso de tareas HW y SW, (en la página de la plataforma [13] se pueden observar los proyectos realizados hasta el momento); para esto se formarán equipos de trabajo de 3 personas que deben hacer una propuesta inicial (concepción y especificaciones), realizar la descripción funcional del sistema utilizando algoritmos (diseño y modelo del sistema), realizar el particionamiento en funciones HW y SW, implementar estas funciones y diseñar una placa de circuito impreso (utilizando *kicad*) con lo necesario para implementar la funcionalidad requerida (implementación). Se realizarán entregas periódicas para verificar el cumplimiento del cronograma propuesto por el equipo de trabajo y el proceso de diseño debe ser documentado en la página wiki del curso. Esta actividad generará habilidades de trabajo en equipo, elaboración de esquemáticos, fabricación de PCBs, escritura de documentos técnicos e implementación de sistemas digitales.

SIE permite conectar de forma fácil (usando dos jumpers) dos GPIOs de la FPGA a las señales de transmisión y recepción del procesador, lo que permite la comunicación serial entre el procesador implementado en la FPGA y el procesador MIPS, creando de esta forma un canal de depuración para las aplicaciones implementadas en la FPGA, eliminando la necesidad de equipo adicional.

D. Sistemas Embebidos

El tercer y último curso de la línea integra los conocimientos adquiridos en los cursos anteriores e introduce un elemento nuevo un SoC comercial, SIE posee un SoC basado en un procesador MIPS equipado con los recursos necesarios para ejecutar programas Linux; en este curso se estudiará la arquitectura de los SoC comerciales, las herramientas de programación abiertas disponibles (cadena de herramientas GNU, librerías GNU como libQT), cargadores de Linux (u-boot), el sistema operativo Linux, drivers de periféricos, distribuciónes para sistemas embebidos (openwrt, openembedded, debian). Con esto el estudiante estará en capacidad de crear aplicaciones que utilizan la gran variedad de librerías disponibles en

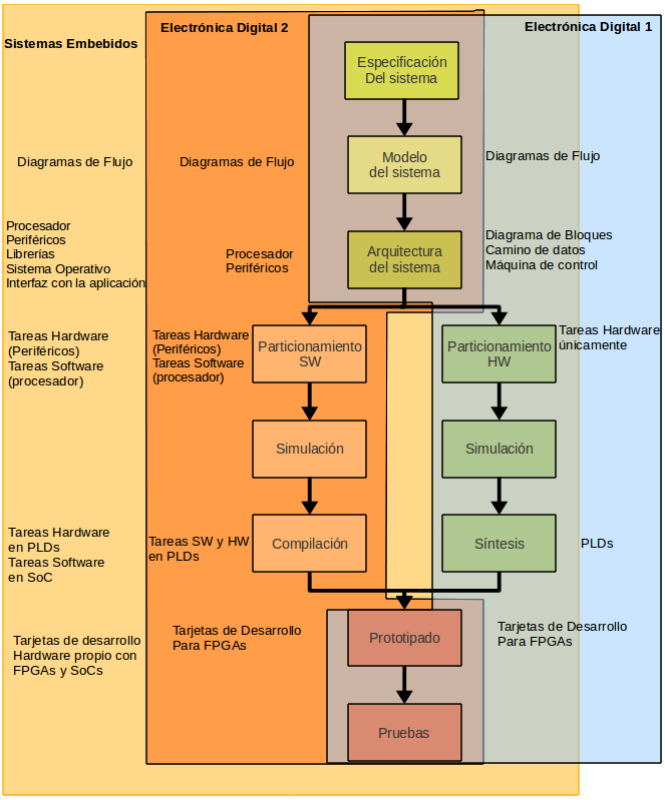

Fig. 4. Metodología de Diseño para el área de Sistemas Digitales

el proyecto FOSS, las mismas que utilizan Nokia, Motorola, Dell, Sony.

1) Creación de periféricos y drivers : Además de cubrir el tema de programación de SoC utilizando GNU/Linux, es necesario que se entienda no solo el funcionamiento del sistema operativo Linux, sino como este se comunica y controla los periféricos (tareas HW); para esto, se implementarán periféricos en la FPGA y se estudiará la forma de acceder a ellos utilizando el protocolo establecido por el kernel.

2) Concepción, Diseño, Implementación y Operación de Sistemas Embebidos: Durante el periodo académico los estudiantes deben diseñar, implementar y operar un sistema embebido concebido por un equipo de trabajo de tres personas, utilizando como base la plataforma SIE, diseñarán una tarjeta hija (utilizando kicad) que implemente la funcionalidad deseada, todos los proyectos deben integrar tareas HW en la FPGA con módulos del kernel para su control. Los proyectos realizados hasta el momento pueden encontrarse en la página del proyecto.

E. Comunidad hardware copyleft

Una parte importante de este método de enseñanza es la filosofía del proyecto hardware copyleft, por esta razón, cada grupo debe hacer un aporte, suministrando la información completa del proceso de desarrollo, los archivos necesarios para replicar y/o modificarlo, esto es una consecuencia de la licencia CC-BY-SA.

La experiencia del proyecto FOSS indica muchos miembros de estas comunidades ingresan para suplir necesidades, pero muchos de ellos continuan creando código y prestando servicios a la comunidad porque disfrutan programar. Estos *aficionados* realizan un papel muy importante dentro de la comunidad encargándose de tareas como mejora de la plataforma tecnológica, re-escribiendo secciones de código, documentandolo, respondiendo preguntas, preservando o mejorando la arquitectura [15]. Las actividades de documentación además de contribuir a mejorar las habilidades de escritura de reportes técnicos ayudan a formar una comunidad que contribuye al crecimiento del proyecto copylef harware, los estudiantes ingresan a las listas de desarrolladores aprendiendo a utilizar una herramienta muy poderosa en la que pueden compartir sus inquietudes con miembros más experimentados y mientras participan ayudan a crear un banco de preguntas que pueden ser útiles para futuros miembros. Adicionalmente se obliga a expresarse en un idioma diferente.

Crear estos hábitos ayuda a que los jóvenes sean conscientes de su papel dentro de la comunidad y piensen que sus acciones pueden ayudarla o perjudicarla, los proyectos realizados por ellos podrán ser parte de los recursos de la comunidad (si la calidad del trabajo lo amerita) y pueden ser la continuación de un esfuerzo prolongado o el punto de partida de un nuevo conocimiento; la licencia CC-BY-SA garantiza que todos los trabajos derivados de este recurso serán parte del mismo, lo que garantiza su crecimiento, la labor de los estudiantes es vital para el uso del recurso común y puede crear miembros que en un futuro formularán políticas y reglas de uso del recurso. Por otro lado, participar en este tipo de proyectos permite crear reputación, la cual puede ser útil para establecer relaciones profesionales, de negocios o personales. El entorno académico es ideal para atraer nuevos miembros a la comunidad hardware copyleft, ya que se trabaja con jóvenes con deseos de ser parte de un grupo y de adquirir conocimientos. Desde el punto de vista comercial este recurso es muy atractivo ya que permite ahorrar mucho tiempo, esfuerzo y dinero para la creación de nuevos productos. Por otro lado, el concepto de hardware copyleft es una herramienta poderosa para transferir tecnología y conocimientos a los países en vía de desarrollo donde la plataforma tecnológica no se lo suficientemente desarrollada.

III. Conclusiones

El hardware copyleft es una herramienta poderosa para la creación de habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, ya que proporciona la información necesaria para entender el ciclo completo de diseño, (lo cual no es posible obtener cuando se trabaja con plataformas de desarrollo comerciales); proporcionando información detallada sobre el proceso de diseño de plataformas abiertas, que pueden ser utilizadas como referencia para generar nuevos productos comerciales; el acceso a aplicaciones software que permiten la creación de aplicaciones; un canal de comunicación que permite utilizar a la industria manufacturera asiática para la producción en masa; conocimiento de los procesos de fabricación y producción.

Las actividades propuestas en las tres asignaturas del área tienen como objetivo generar en el estudiante las habilidades necesarias que le permitan diseñar sistemas digitales con grado de complejidad creciente, hasta llegar a un sistema que puede ser comercializable y satisface una necesidad de una determinada comunidad, con esto, se evita que el último paso en el proceso de enseñanza sea la simulación; se ilustra el proceso que debe seguirse para que un prototipo se convierta en un producto comercial, lo que contribuirá con la creación de nuevos productos y la generación de empleo.

La utilización de herramientas de bajo nivel permite que el estudiante conozca y controle los diferentes pasos de la metodología de diseño y sea capaz de ajustarlas para diferentes situaciones, esto hace que se adquiera un conocimiento sobre la tecnología sin crear dependencia hacia las herramientas comerciales que realizan la mayoría de los pasos de la metodología de forma automática.

References

[1] R. M. Stallman. *The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.

[2] A. Grove. How America Can Create Jobs. http://www.businessweek.com/magazine/content/10_28/b4186048358596.htm, May 2010.

[3] Jon Hall. POR GRANDES QUE SEAN...: ASEGURE EL FUTURO DE SU NEGOCIO. *Linux magazine,*, ISSN 1576-4079(58):92, 2009.

[4] H. Mitsui, H. Kambe, and H. Koizumi. Use of Student Experiments for Teaching Embedded Software Development Including HW/SW Co-Design. *IEEE TRANSACTIONS ON EDUCATION,*, 52(3), August 2009.

[5] A. Gerstlauer, D. Gajski., . Technical Report CECS-02-17, and 2002. CECS, UC Irvine. System-level abstraction semantics, Technical Report CECS-02-17. Technical report, CECS, UC Irvine, 2002.

[6] Gajski D.D., Abdi S., Gerstlauer A., and Schirner G. *Embedded System Design: Modeling, Synthesis, Verification*. Springer, 2009.

[7] Luis Alejandro Cortés. *Verification and Scheduling Techniques for Real-Time Embedded Systems*. PhD thesis, Linköpings universitet Institute of Technology, 2005.

[8] Qi Hardware. Qi Hardware Copyleft Hardware Project. URL: http://en.qi-hardware.com/.

[9] R. Stallman. Philosophy of the GNU project. URL: http://www.gnu.org/philosophy/, 2007.

[10] W. Spraul, C. Camargo, and A. Wang. Proyecto SAKC. *URL:http://en.qi-hardware.com/wiki/SAKC*.

[11] Creative Commons. Licencias Creative Commons. URL: http://creativecommons.org/licenses., 2004.

[12] C. Camargo. Proyecto SIE: Descargas. http://projects.qi-hardware.com/index.php/p/nn-usb-fpga/source/tree/master/, 2010.

[13] C. Camargo. Proyecto SIE: Documentación. http://en.qi-hardware.com/wiki/SAKC, 2010.

[14] Texas Instruments. IEEE Std 1149.1 (JTAG) Testability. *1997 Semiconductor Group*, 1996.

[15] S. Shah. Motivation, Governance, and the Viality of Hybrid Forms in Open Source Software Development. *Management Science*, July 2006.