

1 Implementación de la Metodología de Transferencia Tecnológica Propuesta en el diseño de Sistemas Embebidos

1.1. Introducción

1.1.1. Qué es un Sistema Embebido

Un Sistema Embebido (Sistema Embebido (ES)) es un sistema de propósito específico en el cual, el computador es encapsulado completamente por el dispositivo que el controla. A diferencia de los computadores de propósito general, los sistemas embebidos son diseñados para una aplicación específica, es decir, estos sistemas realizan un grupo de funciones previamente definidas y una vez el sistema es diseñado, no se puede cambiar fácilmente su funcionalidad; debido a su interacción con el entorno deben cumplir restricciones temporales estrictas, el término *sistemas de tiempo real* es utilizado para enfatizar este aspecto; son heterogéneos, es decir, están compuestos por componentes hardware (Programmable Logic Device (PLD)s, Application-specific integrated circuit (ASIC)s) y software (μ -controladores, μ -procesadores, Digital signal processor (DSP)s); tienen grandes requerimientos en términos de confiabilidad.

1.1.2. Arquitectura

En la Figura 1.1 se muestra la arquitectura típica de un Sistema Embebido. La cual integra un componente hardware (HW), implementado ya sea en un (Complex programmable logic device (CPLD), Field Programmable Gate Array (FPGA)) o en un ASIC, (conocido con el nombre de tarea hardware) y un componente software (SW) capaz de ejecutar software; la parte del procesador está dividida en la Central processing unit (CPU) (En algunos casos posee una caché) y las unidades de Memoria.

Al momento de diseñar un sistema embebido se encuentran diferentes opciones de implementación, la más adecuada, resultará de un análisis económico donde se valora el costo de la solución ante el cumplimiento de los requerimientos del sistema; estas opciones son:

- Componente HW y SW integrado en un dispositivo semiconductor (System on Chip - System On a Chip (SoC), Circuito integrado de aplicación específica - ASIC): En la actualidad existen muchas compañías que fabrican procesadores de 32 bits conectados a una gran variedad de periféricos y fabricados en un mismo circuito integrado, lo que simplifica el diseño y reduce costos de materiales. Este tipo de implementación es muy popular en los dispositivos de consumo masivo (reproductores de MP3, consolas de juego, teléfonos celulares, etc.), debido a los grandes niveles de producción (del orden de millones de unidades) resulta más económico contar con un dispositivo que integre el mayor número de funcionalidades que disminuye el costo de componentes y reduce el área de circuito impreso.

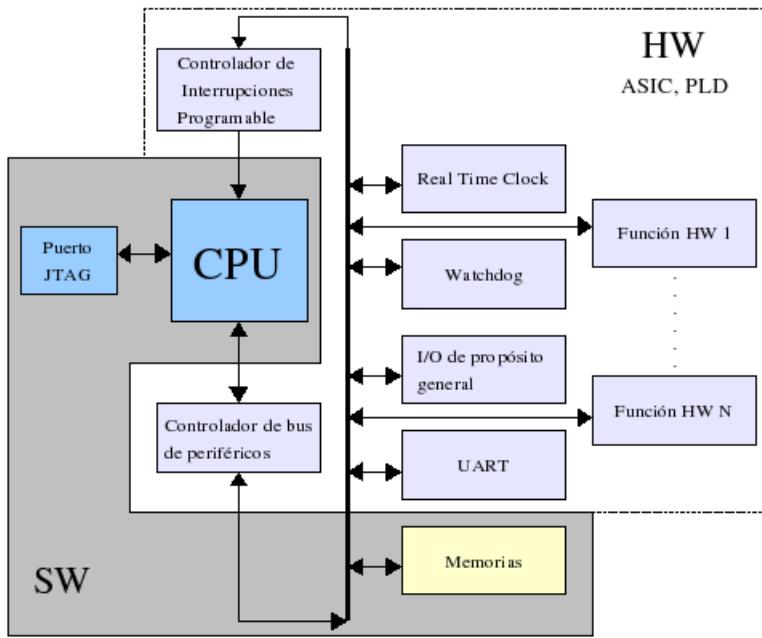


Figura 1.1: Arquitectura de un Sistema Embebido

- Componente SW en un SoC y componente HW en un dispositivo lógico programable (CPLD, FPGA): Cuando no existe en el mercado un SoC con la cantidad de periféricos requerida para una determinada aplicación, o con una funcionalidad específica, es necesario recurrir a la utilización de dispositivos comerciales que implementen dicha función; en algunas ocasiones el periférico puede realizar funciones poco comunes y no se proporciona comercialmente, la solución es entonces, implementar estas funcionalidades en un dispositivo lógico programable (PLD). También se recomienda la utilización de PLDs en sistemas que requieren la utilización de la misma funcionalidad un gran número de veces (puertos seriales, pines de entrada/salida). Esta decisión está atada al nivel de producción, ya que al incluir un PLD aumenta el costo global del proyecto y el consumo de potencia (el consumo de las FPGAs actuales las hace pocas prácticas para aplicaciones móviles).
- Componente SW y HW en una FPGA: Esta es la opción más flexible, pero la de menor desempeño, ya que al utilizar los recursos lógicos de la FPGA para la implementación del procesador (*softcore*) la longitud y capacitancia asociada a los caminos de interconexión entre los bloques lógicos aumentan el retardo de las señales, lo que disminuye la máxima velocidad de funcionamiento. Los procesadores *softcore* más populares en la actualidad son: Microblaze y Picoblaze de Xilinx, Leon de Gaisler Research y Lattice-Mico32 de Lattice Semiconductors.

1.1.3. Aplicaciones

Los sistemas embebidos se encuentran en casi todas las actividades humanas, a diario se interactúa con ellos, aún sin darse cuenta, ya sea porque son parte de la vida diaria o porque hacen parte de aparatos que se utilizan a diario. La figura 1.2 muestra los campos de aplicación de los

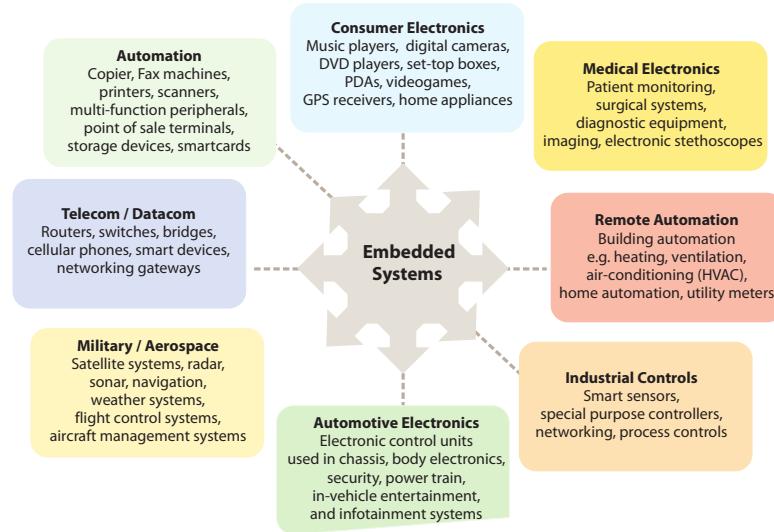


Figura 1.2: Aplicaciones de los Sistema Embebidos Fuente: TATA Consultancy services

sistemas embebidos.

1.1.4. Metodología de Diseño

El proceso de diseño de un sistema embebido comienza con la *especificación del sistema*, (ver Figura 1.3), en este punto se describe la funcionalidad y se definen las restricciones físicas, eléctricas y económicas del sistema. Esta especificación debe ser muy general y no deben existir dependencias tecnológicas de ningún tipo, se suelen utilizar lenguajes de alto nivel, como Unified Modeling Language (UML), Modeling and Analysis of Real-Time and Embedded Systems (MARTE) C++, System-C, Spec-C. La especificación puede ser verificada a través de una serie de pasos de análisis cuyo objetivo es determinar la validez de los algoritmos seleccionados, por ejemplo, determinar si el algoritmo converge o si sus resultados satisfacen las especificaciones. Desde el punto de vista de la re-utilización, algunas partes del funcionamiento global pueden tomarse de una librería de algoritmos existentes.

Una vez definidas las especificaciones del sistema, se debe realizar un modelamiento que permita extraer de estas su funcionalidad. El modelamiento es crucial en el diseño ya que de él depende el paso exitoso de la especificación a la implementación. Es importante definir que modelo matemático debe soportar el entorno de diseño; cada modelo posee propiedades matemáticas que pueden explotarse de forma eficiente para responder preguntas sobre la funcionalidad del sistema sin llevar a cabo dispendiosas tareas de verificación. Todo modelo obtenido debe ser verificado para comprobar que cumple con las restricciones del sistema.

Una vez se ha obtenido el modelo del sistema se procede a determinar su *arquitectura*, esto es, el número y tipo de componentes y su inter-conexión; este paso no es mas que una exploración del espacio de diseño en búsqueda de soluciones que permitan la implementación de una funcionalidad dada, y puede realizarse con varios criterios en mente: costos, confiabilidad y viabilidad comercial.

Utilizando como base la arquitectura obtenida en el paso anterior las tareas del modelo del sistemas son implementadas en los componentes; esto es, asignación de funciones a los componentes

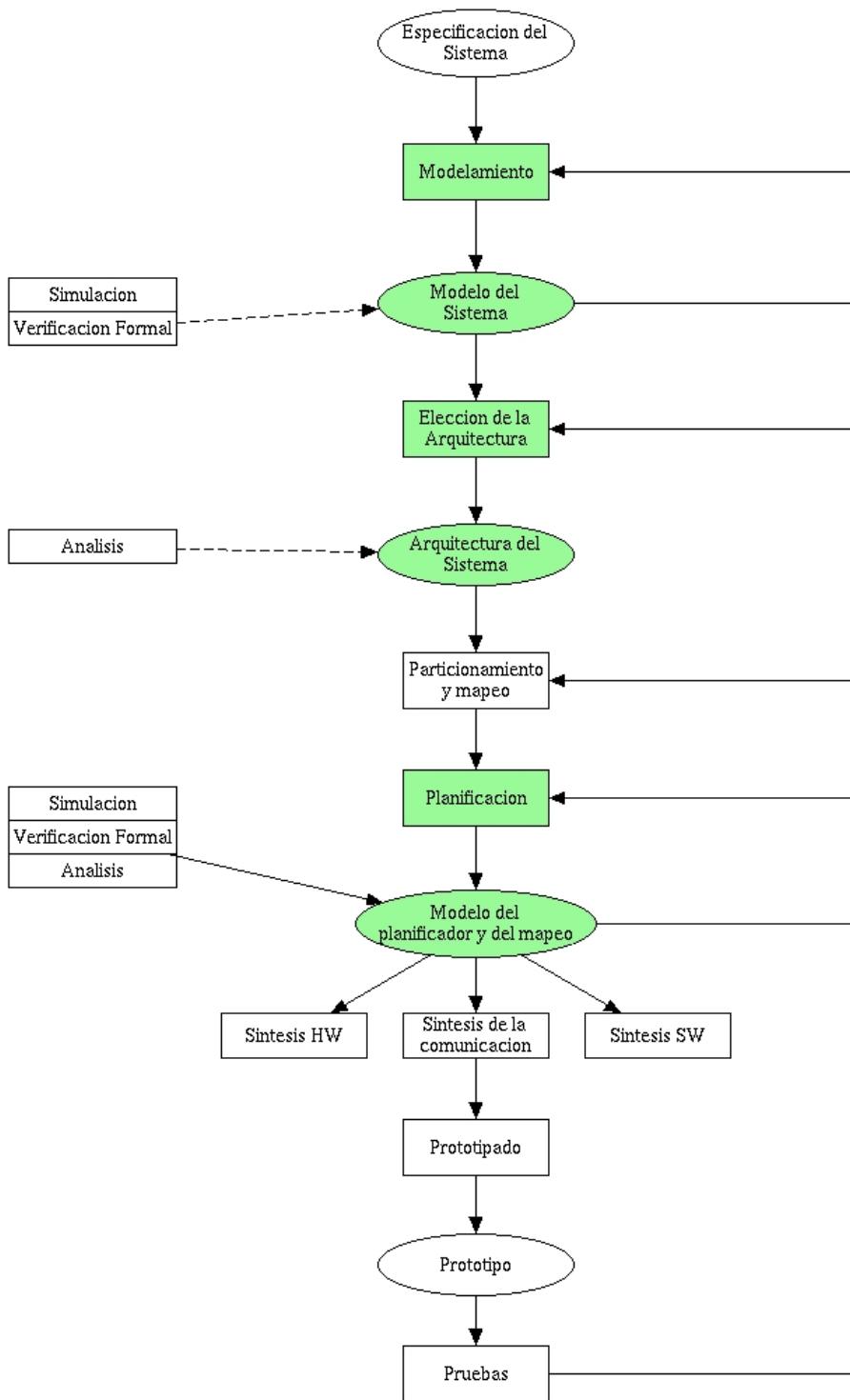


Figura 1.3: Flujo de diseño de un sistema embebido [1]

de la arquitectura. Existen dos opciones a la hora de implementar las tareas o procesos:

1. Implementación software: La tarea se va a ejecutar en un micro-procesador, micro-controlador o DSP.
2. Implementación hardware: La tarea se va a ejecutar en un sistema digital dedicado ASIC o PLD.

Para cumplir las especificaciones del sistema algunas tareas deben ser implementadas en hardware, esto con el fin de no ocupar al procesador en tareas cíclicas o que requieran mucho poder computacional, un ejemplo típico de estas tareas es la generación de bases de tiempos o la implementación de protocolos de comunicación. La decisión de que tareas se implementan en SW y que tareas se implementan en HW recibe el nombre de *particionamiento*; esta selección depende fuertemente de restricciones económicas y temporales.

Las tareas software deben compartir los recursos que existan en el sistema (procesador y memoria), por lo tanto se deben tomar decisiones sobre el orden de ejecución y la prioridad de estas. Este proceso recibe el nombre de *planificación*. En este punto del diseño el modelo debe incluir información sobre el mapeo, el particionamiento y la planificación del sistema.

Las siguientes fases corresponden a la implementación del modelo, para esto las tareas hardware deben ser llevadas al dispositivo elegido (ASIC, FPGA, micro-controlador, micro-procesador, DSP) y se debe obtener el *archivo binario* con las instrucciones que implementan la funcionalidad de las tareas software y un *archivo de configuración* para implementarla funcionalidad de las tareas hardware en el PLD, este proceso recibe el nombre de *síntesis HW y SW* respectivamente, así mismo se deben sintetizar los mecanismos de comunicación entre las tareas hardware y software.

El proceso de prototipado consiste en la realización física del sistema, finalmente el sistema físico debe someterse a pruebas para verificar que cumple con las especificaciones iniciales.

Como puede verse en el flujo de diseño existen realimentaciones, estas permiten depurar el resultado de pasos anteriores en el caso de no cumplirse con las especificaciones iniciales.

1.2. Implementación de la Metodología Propuesta para la Transferencia Tecnológica en Diseño de Sistemas Embebidos

1.2.1. Selección de la Tecnología

El Plan Estratégico del Programa Nacional de Desarrollo Tecnológico Industrial y Calidad 2005 - 2015 [2] identifica 5 *tecnologías* con mucho potencial en el área de la electrónica, estas son: microelectrónica, diseño y construcción de circuitos integrados, diseño digital con FPGAs, Printed Circuit Board (PCB) multi-capa y dispositivos de montaje superficial. Aunque todos ellos no representan líneas de estudio, todas hacen parte del mismo tema: diseño y fabricación de sistemas digitales. Se nota cierto desconocimiento en el tema ya que separan el área de la microelectrónica y el diseño de circuitos integrados (la microelectrónica estudia el diseño y fabricación de ASICs); las otras tecnologías en realidad no lo son, el diseño de PCBs, uso de dispositivos de montaje superficial y uso de FPGAs, no representan líneas de investigación, sino herramientas útiles para la implementación de sistemas digitales. Esto demuestra la desconexión que existe entre los entes generadores de políticas y la academia.

Niveles de complejidad de la tecnología

Existen varias alternativas para la implementación de un sistema embebido: dispositivos lógicos programables (FPGAs, CPLDs), sistema sobre silicio (SoC), micro-controlador, micro-procesador, SoC + FPGA y ASIC; su utilización está determinada por el cumplimiento de restricciones temporales, funcionales y económicas. La opción tecnológicamente más avanzada es el uso de un Circuito Integrado de Aplicación Específica (ASIC) que implemente las tareas hardware y software en un solo dispositivo semiconductor; sin embargo, se estima que solo a partir de 10 mil unidades es conveniente utilizar un ASIC para reducir los costos de producción; esta es una cantidad muy alta para las pequeñas industrias electrónicas nacionales, y hasta donde llega el conocimiento del autor de este trabajo, por el momento no se conoce el primer circuito integrado fabricado y comercializado por una empresa colombiana o de existir, no es una práctica común. Por otro lado, las herramientas de desarrollo para el diseño de circuitos integrados son muy costosas y el grado de conocimientos de los diseñadores es mayor que en otro tipo de implementación.

Un proyecto vigente promovido por la Unión Europea llamado *Iberchip* empezó desde hace 17 años un proceso de transferencia tecnológica en el diseño de circuitos integrados de aplicación específica (ASICs) hacia los países iberoamericanos; gracias a esta iniciativa se incluyeron asignaturas relacionadas con el diseño de los sistemas embebidos en la mayoría de las carreras de centros de formación consolidados. Sin embargo; esto no ayudó a aumentar la demanda de estos dispositivos por parte de la industria.

En todas las universidades consolidadas del país es común el uso de los lenguajes de descripción de hardware (Hardware Description Language (HDL)) como herramienta para la implementación de aplicaciones; es normal encontrar trabajos de pregrado y posgrado que utilizan familias de FPGA que incluyen procesadores *hardcore* o *softcore* con periféricos dedicados en aplicaciones de procesamiento de imágenes, o de señales. Sin embargo, aún existen muchos centros de formación que continúan utilizando tecnologías y metodologías de diseño obsoletas y pocas industrias locales reportan el uso de estos dispositivos para desarrollar productos comerciales. Por lo tanto, es necesario crear mecanismos que permitan difundir aún más el uso de las FPGAs. Aunque en la actualidad existe una oferta considerable de cursos de extensión para la capacitación en el uso de dispositivos lógicos programables y lenguajes de descripción de hardware, el uso de esta tecnología debe ser justificada por los requerimientos de la aplicación. Debido a que el uso de estas tecnologías requiere un mayor nivel de conocimiento de los sistemas digitales, es necesario realizar rigurosos procesos de verificación para comprobar su correcto funcionamiento; su depuración es un poco tediosa (en comparación con las tareas software). En conclusión, si se desea impulsar el uso de esta tecnología se debe ser muy cuidadoso al momento de elegir las aplicaciones que serán implementadas, esto con el fin de no desalentar a los usuarios de la misma.

Las FPGAs proporcionan una alternativa flexible para prototipado de ASICs, ya que permiten cumplir de forma rápida con los requerimientos del mercado (el proceso de fabricación de un ASIC toma varios meses). Sin embargo, para que un producto sea viable económico es necesaria una solución ASIC de bajo costo; en la actualidad existe la posibilidad de bajar los costos de producción gracias a la demanda de los mismos y a la utilización de una tecnología intermedia llamada *arreglo de compuertas*, la arquitectura de estos dispositivos proporciona una gran cantidad de transistores en arreglos genéricos en un substrato común; y pueden ser utilizados para la implementación de *celdas estándar* o diseños *full custom*; utilizando esta técnica, es posible reducir el número de unidades necesarias para encontrar un punto económico viable de 5000 unidades (fuente Silicon-Pro).

Según Smith [3], la opción más económica para bajos volúmenes de producción son las FPGAs,

a medida que la producción aumenta se produce un punto de quiebra entre las FPGAs y el arreglo de compuertas cerca a las 5000 unidades, y el segundo punto de ruptura se produce alrededor de las 50000 unidades, donde es más rentable la producción de un ASIC basado en celdas estándar. Es muy importante tener en mente estas cifras ya que ellas determinan la tecnología a utilizar. No obstante, vale la pena aclarar que en esta comparación no se tiene en cuenta la utilización de SoC, micro-controladores o micro-procesadores comerciales, por lo que no es necesariamente cierto que a bajos niveles de producción la opción más rentable sea la utilización de FPGAs; adicionalmente, debido a su alto consumo de potencia (del orden de 10 veces mayor que un ASIC) no es posible su utilización en aplicaciones móviles.

En Colombia es muy común el uso de micro-controladores y micro-procesadores de 8 bits para la implementación de sistemas digitales, la mayor parte de los centros de educación del país (de todo nivel) proporcionan cursos de programación y es posible encontrar en el mercado un suministro continuo de ciertas referencias. Gracias a esto, existen numerosos desarrollos basados en ellos; sin embargo, debido a los limitados recursos de estos dispositivos (velocidad, periféricos, herramientas de programación) no es posible utilizarlos para la implementación de aplicaciones con las necesidades actuales como conectividad con diferentes redes, manejo de pantallas de cristal líquido, aplicaciones multimedia, y diversos medios de almacenamiento de información.

Los *System on Chip* (SoC) proporcionan una excelente alternativa para la implementación de aplicaciones modernas; integran un procesador de 32 bits o un DSP que corre a frecuencias que van desde los 75 MHz hasta los 800 MHz y poseen periféricos que permiten controlar directamente una gran cantidad de dispositivos; muchos de ellos están diseñados para aplicaciones que requieren manejo de pantallas táctiles de cristal líquido, conexión a internet, diferentes medios de almacenamiento, reproducción de audio, manejo de sensores de imagen, entre otros; muchas de estas tareas son realizadas por procesadores dedicados diferentes al procesador principal del SoC. Adicionalmente, existe una gran gama de productos ofrecidos por diversos fabricantes como Freescale, NXP, Ingenic, Analog Devices, Altera, Marvell; por otro lado, su uso en aplicaciones de consumo masivo ha reducido el costo de estos dispositivos y es posible comprarlos en cualquier cantidad a precios que oscilan entre 4 y 20 United States dollar (USD).

Diagnóstico de la industria local

La empresa emQbit desarrolló una serie de proyectos y actividades que ayudaron a entender e identificar los siguientes obstáculos para el desarrollo y comercialización de sistemas digitales: Falta de proveedores de bienes y servicios relacionados con la actividad (venta de dispositivos semiconductores, fabricación de placas de circuito impreso, montaje automático de componentes, etc); desconocimiento de la tecnología (alcances y limitaciones) debido al uso de tecnologías y metodologías de diseño obsoletas; competencia con productos asiáticos de muy bajo costo; falta de confianza en los productos nacionales; desconexión de la academia con el sistema productivo; inexistencia de reglamentación de la industria de manufactura electrónica; profesionales con pocos conocimientos en procesos de diseño y fabricación. Estas observaciones coinciden con los resultados de estudios consultados [4] [5] [6] [7] [8] [9].

Diagnóstico de la academia

Basándose en conversaciones con encargados de la línea de electrónica digital de varias universidades, se puede decir que la tendencia moderna en los programas académicos a la utilización

de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales [10] ocasiona que los profesionales no adquieran ciertas habilidades necesarias para completar la cadena concepción - diseño - implementación - 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 técnicos especializados para operarlas; ; limitando la generación de empleo local a personas con alto nivel de formación [11] generando desempleo en las personas menos capacitadas. Según John Hall presidente y Chief executive officer (CEO) de Linux International “ algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico” [10]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más.

Por otro lado, en muchas instituciones educativas de poca consolidación se utilizan: tecnologías y metodologías de diseño obsoletas (Familias 74XXX o 40XXX, lenguaje ensamblador, mapas de karnaugh); y programas académicos centrados en el análisis y no el diseño, donde el paso final es la simulación. Esto, unido al poco contacto por parte del personal docente con el sector productivo, origina una deficiencia de habilidades necesarias para realizar el proceso completo para el diseño de dispositivos, lo que se traduce en profesionales que no disponen de las herramientas necesarias para resolver los problemas del país y al mismo tiempo competir con los productos asiáticos.

Centros de desarrollo tecnológico

Según el XII Encuentro Colombia compite (patrocinado por el Ministerio de comercio industria y turismo - 2010) el sector electrónico cuenta con solo un centro de desarrollo el CIDEI (Centro de Investigación y Desarrollo de la Industria Electro Electrónica e Informática), el cual ha sido beneficiado con recursos cercanos a los dos mil millones de pesos para la ejecución de proyectos de desarrollo empresarial y con 1200 millones para proyectos de desarrollo tecnológico. Proporciona 2 servicios a la comunidad: diseño, fabricación y montaje de PCBs y capacitación. Los conocimientos generados en estos proyectos no son difundidos a las universidades ni a la sociedad de forma gratuita, lo que limita la difusión de estos conocimientos a quien pueda pagar por ello; por otro lado, la Universidad Nacional proporcionó la capacitación para algunos de estos proyectos, lo que indica que este centro no genera conocimiento propio y los temas de sus investigaciones ya han sido tratados por varias universidades del país.

Discusión

Como se vio anteriormente, la opción más económica para niveles de producción inferiores a 5000 unidades son las FPGAs; sin embargo; esto es cierto únicamente si no existe un dispositivo comercial como SoC, DSP, micro-controlador, o micro-procesador que permita el cumplimiento de las restricciones del sistema. Esto, debido a que el costo de las FPGAs es más elevado y como se dijo anteriormente su consumo de potencia impide su utilización en aplicaciones móviles (por esta razón no es común encontrar FPGAs en dispositivos de consumo masivo). Una revisión de los circuitos integrados utilizados en dispositivos como reproductores MP3, juegos, routers, y algunos teléfonos móviles, revela el uso de SoCs comerciales de diferentes fabricantes. Los fabricantes de SoC se adaptan rápidamente a los requerimientos del mercado y proporcionan dispositivos con los periféricos necesarios para una determinada aplicación; un ejemplo de esto se puede observar en las diferentes

versiones de reproductores MP3 de la compañía *AINOL*, la primera versión evaluada contenía un DSP de Analog Devices, un codec de audio externo y un controlador para el puerto Universal Serial Bus (USB) (Universal Serial Bus); en la siguiente versión evaluada solo se encontró un circuito integrado del fabricante *Ingenic*; una revisión de la hoja de especificaciones de este último circuito integrado indicaba que este ya poseía el codec de audio y el controlador de la interfaz USB como periféricos internos; adicionalmente, su costo es de 3.5 USD en grandes volúmenes. Los grandes fabricantes como Atmel, Freescale, Marvell, NXP, *Ingenic*, etc, de forma dinámica ajustan sus productos a los requerimientos del mercado, agregando el soporte que demandan las aplicaciones. Gracias a esto y a la creciente demanda es posible encontrar SoCs muy baratos, con grandes capacidad de cómputo que pueden ser utilizados en muchas aplicaciones. Por esta razón y por la poca utilización de los SoC en el país, se trabajará en el estudio de técnicas de fabricación y metodologías de diseño basadas en estos dispositivos; esto, sin descuidar el fomento de la utilización de las FPGAs a nivel académico e industrial y el estudio de técnicas de diseño y fabricación de circuitos integrados para estar preparados a un aumento en la producción que justifique su uso.

La hipótesis de este trabajo es que el aumento de diseños locales que utilicen SoC y metodologías de diseño modernas permitirán a la industria electrónica local competir con productos importados; debido a que, el país no cuenta con una oferta considerable de bienes y servicios relacionados con la industria manufacturera de dispositivos digitales (sólo existen 2 empresas), es necesario, utilizar inicialmente los servicios de la industria asiática para construir dispositivos diseñados en el país, que puedan ser configurados para las necesidades exactas del entorno social local y de esta forma aumentar la demanda interna. Finalmente, se debe continuar con el estudio de técnicas de fabricación de circuitos integrados para una futura demanda de la industria local.

1.2.2. Adquisición de Tecnología

Herramientas de desarrollo

Las herramientas de desarrollo son fundamentales en el proceso de diseño, de su estado y capacidades depende el tiempo necesario para completar un determinado diseño; la disponibilidad de aplicaciones y librerías que permitan acelerar el proceso de diseño son puntos claves a la hora de seleccionar el entorno de desarrollo; otro factor importante es su costo, ya que pequeñas y medianas empresas no pueden invertir grandes sumas de dinero en su adquisición; adicionalmente, es crucial contar con una adecuada documentación e información que ayude a resolver problemas que se presenten en el ciclo de diseño. Se pueden clasificar estas herramientas en *propietarias y abiertas*, las primeras requieren la compra de licencias para su uso y es necesario pagar por soporte; las segundas, son distribuidas de forma gratuita y existe una gran cantidad de listas de discusión donde puede encontrarse respuesta a una gran variedad de problemas o pueden ser formuladas nuevas preguntas a un grupo especializado de usuarios.

La utilización de herramientas abiertas reduce de forma considerable la inversión en la plataforma de desarrollo; pero, ¿es posible realizar el flujo completo de concepción, diseño e implementación utilizando software abierto?, ¿el estado de desarrollo de las mismas facilita el diseño?, ¿existen dispositivos comerciales desarrollados con estas herramientas?. Para resolver estas dudas se consultaron varias encuestas realizadas por compañías y sitios especializados para observar la tendencia en utilización de sistemas operativos; los sitios consultados (Venture Development Corp, linuxfordevices) indican que el 27.9 % de los diseñadores utiliza sistemas operativos licenciados comercialmente, el 23.5 % sistemas operativos obtenidos públicamente, 15.9 % desarrollan su propio

sistema operativo, el 12.1 % utiliza sistemas operativos comerciales basados en proyectos abiertos y el 30 % restante no utiliza un sistema operativo (ver figura 1.4; el porcentaje de utilización de sistemas operativos basados en proyectos abiertos es del 35.6 %, lo que supera a los sistemas operativos comerciales; es interesante observar que casi el 70 % de los encuestados utilizan algún tipo de sistema operativo, lo que da un claro indicio de la necesidad de este en el ciclo de diseño).

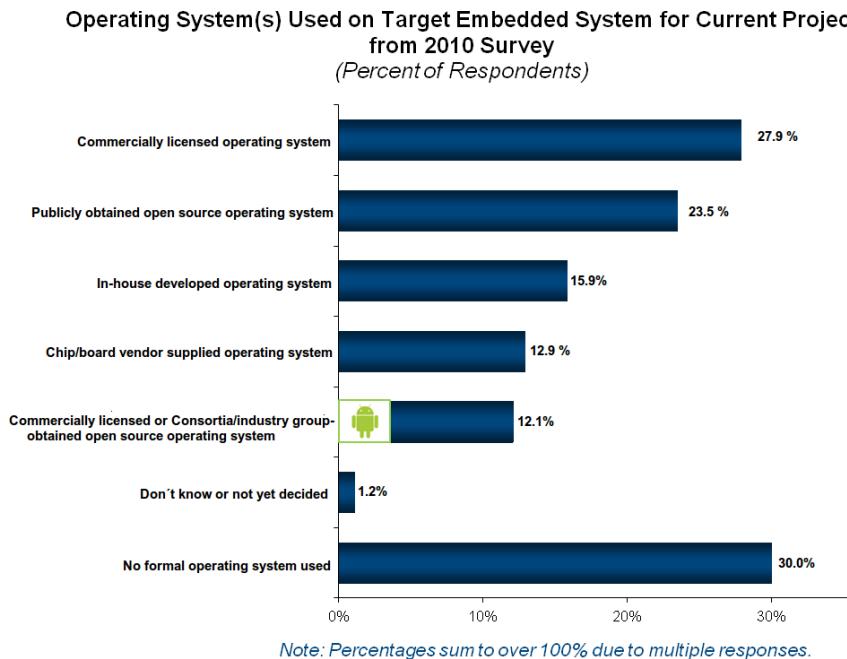


Figura 1.4: Comparación del uso de sistemas operativos Fuente: Venture Development Corp

De lo anterior se puede afirmar que más de la mitad de los diseñadores que utilizan sistemas operativos para sus aplicaciones eligen proyectos abiertos, lo que indica que estas herramientas tienen el grado de madurez necesaria para su uso en aplicaciones comerciales; por otro lado, una revisión del mercado de los teléfonos móviles realizada por *Admob* indica que android superó a los sistemas operativos de Apple y RIM (utilizado en los blackberry) y se proyecta que en el 2014 igualará a Symbian de Nokia. Android utiliza el kernel de Linux como base de sus aplicaciones y utiliza herramientas abiertas para su desarrollo; otras empresas como Motorola y Nokia utilizan Linux como plataforma de algunas de sus aplicaciones; así mismo, muchos routers basados en procesadores ARM o MIPS; una gran variedad de reproductores multimedia, tablets y mini-laptops; todo esto, unido a la disponibilidad de foros de discusión donde programadores expertos y creadores de una gran variedad de aplicaciones brindan soporte a quien este interesado, hace de las herramientas abiertas y de Linux, una alternativa muy atractiva para desarrollar una metodología de diseño en torno a ella y adaptarla a las necesidades del país.

Linux Foundation publicó un estudio [12] donde calcula que el valor del kernel de Linux es de USD\$1400 millones; y son necesarios USD\$10.800 millones para desarrollar el stack completo de componentes desde cero; por este motivo, el uso de Linux reduce de forma considerable los costos finales del proyecto, *Black Duck Software*¹ posee la más completa base de datos de proyectos

¹<http://www.blackducksoftware.com> Líder mundial en el suministro de productos y servicios que aceleran el desarrollo

abiertos, representados en 200.000 proyectos, 4.9 billones de líneas de código; utilizando su detallado conocimiento de los proyectos abiertos y aplicando técnicas estándar de estimación de costos, calculan que el costo de desarrollo total del proyecto FOSS excede los USD\$387000 millones y representa la inversión colectiva de mas de dos millones de desarrolladores al año. Un análisis adicional, estima que el 10% de las aplicaciones utilizadas para el desarrollo de aplicaciones en tecnología de la información se pueden reemplazar por proyectos abiertos², lo que ahorraría mas de USD\$22 billones al año.

Los proyectos de código abierto permiten a las organizaciones ahorrar tiempo y dinero en muchos aspectos, al no tener que pagar por las herramientas de desarrollo y por librerías y aplicaciones que pueden utilizar para la implementación de nuevos productos; permitiendo la inversión de tiempo y esfuerzo en proyectos que pueden ser comercializados rápidamente.

Dispositivos semiconductores

En la actualidad existe una gran oferta de SoCs, grandes compañías proporcionan constantemente nuevos dispositivos con una gran variedad de periféricos para diferentes aplicaciones. El procesador más utilizado para aplicaciones embebidas es el procesador ARM (Advanced RISC Machine). ARM no fabrica circuitos integrados, suministra sus diseños en forma de *netlist* a nivel de compuertas o a nivel de Lógica de Transferencia de Registros (Register-transfer level (RTL)) en un lenguaje de descripción de hardware. Estas descripciones pueden ser utilizados en el proceso de diseño ASIC, permitiendo su integración con una gran variedad de núcleos Intellectual Property (IP) (Intellectual Property); compañías como Atmel, Marvell, Freescale, NXP, Cirrus Logic, Samsung, Texas Instruments adquieren licencias que les permiten utilizar estos núcleos lógicos en la fabricación de sus SoCs.

Encuestas realizadas a diseñadores por el portal *linuxfordevices*² sobre sus preferencias en el procesador utilizado en sus proyectos; como se dijo anteriormente. ARM es el más utilizado (30 %) seguido de cerca por los basados en x86 (25 %), la arquitectura POWERPC (15 %), MIPS (10 %), DSPs (5 %). Por este motivo, en esta investigación se utilizaron dispositivos basados en procesadores ARM (AT91RM9200 y SAM7 de Atmel, imx233 de Freescale), MIPS (JZ4740 de Ingénierie) y el DSP de Analog devices BF532, cubriendo de esta forma un gran campo de aplicaciones.

Dispositivos disponibles

Existen dos tipos de dispositivos que pueden ser utilizados en el proceso de transferencia tecnológica y de conocimientos: productos de consumo, productos fabricados a grandes volúmenes y que se pueden adquirir fácilmente a bajo precio; y dispositivos *Commercial off-the-shelf* (*Commercial off-the-shelf* (COTS)), diseñados para ser utilizados como punto de partida en el desarrollo de una aplicación, se pueden encontrar en la forma de tarjetas de desarrollo, las cuales incluyen una gran variedad de dispositivos hardware (puertos de comunicación, Light-Emitting Diode (LED)s, Displays, Liquid Crystal Display (LCD)s, pulsadores) que permiten explorar la capacidad de un determinado SoC, o se pueden encontrar en forma de unidades genéricas que proporcionan las conexiones para su funcionamiento básico (alimentación, memorias), proporcionando todas las señales que controlan periféricos externos para que sean utilizadas en una tarjeta que integre los componentes requeridos

software utilizando software libre

²Linuxfordevices es un sitio de encuentro y centro de noticias para la comunidad que utiliza Linux en dispositivos digitales.

en una determinada aplicación. El término Original Equipment Manufacturer (OEM) (Original Equipment Manufacturer) se aplica a las organizaciones que compran estos artículos y los revenden; en algunas ocasiones se realizan mejoras como valor agregado.

En la actualidad existe una gran oferta de este tipo de dispositivos a nivel mundial, muchas compañías realizan diseños para que sean utilizados como punto de partida de productos comerciales, o que sean parte de ellos, lo que reduce los costos y tiempos de desarrollo; desafortunadamente en el país no existe una gran demanda, por lo que no existen proveedores locales y deben ser importados, lo que hace que su costo se eleve por lo menos en un 26 % (arancel e IVA).

Durante la primera etapa de este estudio se utilizaron los siguientes dispositivos: GameBoy + Xport, es una combinación de la consola de juegos ©NINTENDO (basado en ARM7) y la plataforma *XPORT* de la compañía norteamericana Charmedlabs (basado en una FPGA Spartan3); la plataforma de desarrollo de la compañía china *Embest* basada en el SoC AT91R40008 (ARM7); la agenda electrónica Zaurus de Toshiba (basado en el StrongARM de Intel SA-1110); el dispositivo chumby (basado en un ARM de freescale iMX21); el MP4 de Ainol V2000 que posee un procesador MIPS fabricado por la compañía china Ingénic; la agenda iPAQ H3600 fabricada por Compaq basada en un procesador Strong ARM de Intel; el porta-retratos digital de la empresa china *SUNGALE* basada en un procesador MIPS de la empresa Ingénic y finalmente el lector de libros electrónicos de Barnes & Noble basado en un procesador ARM fabricado por Samsung.

Discusión

En la actualidad es posible adquirir dispositivos semiconductores y demás componentes necesarios para la fabricación de sistemas digitales, sin embargo, para poder competir en precios con productos del mercado asiático, es necesario entender la dinámica de su industria manufacturera, y establecer relaciones con los proveedores de bienes y servicios en esta región. El mercado asiático se ha especializado en el aprovechamiento de recursos y posee diferentes niveles de calidad de los componentes y productos; por esta razón, es posible encontrar teléfonos móviles desde USD \$10 hasta los más costosos de cientos de USDs; los bajos costos son posibles gracias al reciclaje de componentes, una gran cantidad de trabajadores desmontan circuitos integrados y componentes de dispositivos desecharados, y son re-utilizados en nuevos productos, el nivel de reciclaje es tan alto que únicamente las placas de circuito impreso son descartadas; obviamente la confiabilidad de estos dispositivos es menor que la de los dispositivos nuevos, por lo cual es muy importante contar con fuentes de suministros confiables. Por otro lado, en el mercado asiático es posible conseguir dispositivos semiconductores que no se conocen en occidente, y normalmente son más baratos que productos similares de compañías reconocidas mundialmente, sin embargo, muchos de estos dispositivos se encuentran disponibles únicamente para ciudadanos chinos, lo que hace imprescindible el establecimiento de relaciones con facilitadores asiáticos.

1.2.3. Adaptación

En esta etapa se utilizaron las plataformas comerciales mencionadas anteriormente y las herramientas de desarrollo abiertas para: explorar las diferentes alternativas de implementación; entender y aplicar la metodología de diseño adoptada internacionalmente; generación de conocimiento sobre el uso de las herramientas de desarrollo.

Los dispositivos comerciales con grandes niveles de producción son un buen punto de partida para el estudio de la arquitectura de sistemas digitales que utilizan SoCs, ya que están optimizadas

(en número de componentes) para ser utilizadas en una aplicación específica, gracias a esto, su costo es muy bajo (en comparación con plataformas de desarrollo). Por otro lado, es posible encontrar estos dispositivos en el mercado local; reduciendo de forma considerable la inversión necesaria para realizar las actividades de esta etapa.

Metodología

Se utilizó ingeniería inversa para determinar como están construidos, como funcionan y como son programados los productos adquiridos. Esta tarea no se ejecutó en un único instante de tiempo, ya que el estudio se realizó de forma gradual, de tal forma que cada vez que se abordaba un nuevo tópico se repetía el proceso con un producto que permitiera su estudio.

Como se puede ver en la figura 1.4 existen dos formas de realizar las aplicaciones de un sistema embebido: con sistema operativo (OS) y sin Operating System (OS) (*standalone*). Las aplicaciones standalone son muy eficientes debido a que son escritas teniendo en cuenta los recursos hardware (memoria, velocidad del procesador y periféricos) de la plataforma donde van a ser ejecutadas, sin embargo, requiere el desarrollo completo de toda la funcionalidad. Por otro lado, los sistemas operativos proporcionan servicios (manejo de periféricos, capacidad de ejecución multi-tarea, manejo de sistemas de archivos y librerías) que facilitan el desarrollo de aplicaciones; sin embargo, al ser diseñados para ser ejecutados en cualquier dispositivo, es necesario cumplir con unos requerimientos de recursos mínimos para que se puedan ejecutar las aplicaciones. En este estudio se trabajó con dos sistemas operativos abiertos: *eCos* desarrollado por Redhat y *Linux* desarrollado por Linus Torvalds. A continuación se muestran los temas de estudio que se abordaron en esta etapa.

- Arquitectura: Determinar los componentes más utilizados y las diferentes topologías.
- Programación: Mecanismos que permiten cambiar el *firmware* original del dispositivo.
- Aplicaciones sin sistema operativo: Estudio de herramientas que permiten generar nuevas aplicaciones en los dispositivos sin el uso de sistemas operativos.
- Aplicaciones con sistema operativo: Uso de sistemas operativos para acelerar el proceso de desarrollo de aplicaciones, utilizando las facilidades que ellos proporcionan.
 - *eCos*: Sistema operativo de tiempo real altamente configurable que permite el uso eficiente de los recursos hardware.
 - *Linux*: Sistema operativo ampliamente utilizado que posee una gran cantidad de aplicaciones y un número considerable de desarrolladores.
 - Inicialización: Requerimientos mínimos para la ejecución de *Linux*.
 - Imagen del kernel: Adaptación del kernel de *Linux* a una determinada plataforma.
 - Sistema de archivos: Distribuciones y aplicaciones mínimas necesarias para la ejecución de *Linux*.
 - Drivers de dispositivos: Como dar soporte a nuevos periféricos.
 - Comunicación con periféricos dedicados implementados en FPGAs: Comunicación entre FPGAs y SoCs.
 - Aplicaciones gráficas: Uso de librerías gráficas para el desarrollo de aplicaciones.

Arquitectura: SoC, memorias, periféricos

Los SoCs comerciales se pueden dividir en dos grandes grupos dependiendo de la existencia o no de memoria no volátil para el almacenamiento del programa (memoria de instrucciones) dentro del SoC. Los que poseen memoria no volátil (hasta 512 Kbytes) normalmente incorporan una memoria Random Access Memory (RAM) (hasta 32 kbytes) junto con una serie de periféricos (timers, Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Universal asynchronous receiver/transmitter (USART)s, Analog-to-digital converter (ADC)s, Watchdog, USB device, canales para acceso directo a memoria - Direct memory access (DMA)); están diseñados para no utilizar componentes externos; normalmente este tipo de dispositivos utilizan procesadores que no tienen unidad de manejo de memoria³ (MMU) como la familia ARM7, cuyas velocidades de ejecución varían entre los 50 y 70MHz. En la figura 1.5 se muestra la arquitectura típica de un sistema basado en estos dispositivos.

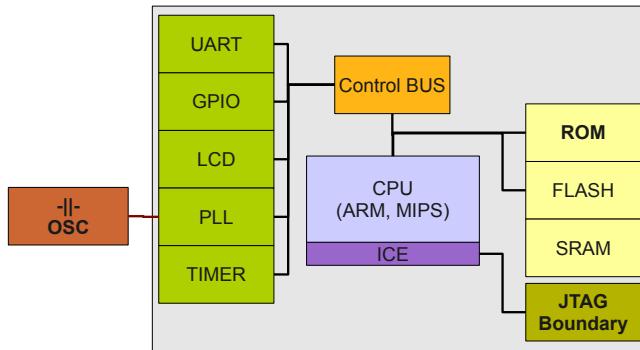


Figura 1.5: Arquitectura típica de un sistema embebido que utiliza SoC con memoria volátil interna

Los procesadores que no poseen memoria no volátil interna se dividen en dos grupos: los que poseen o no unidad de manejo de memoria; en ambos casos, se cuenta con una memoria RAM interna (del orden de cientos de Kbytes) y adicionalmente a los periféricos mencionados anteriormente se suministran controladores para USB host, puertos Serial Synchronous Interface (SSI), controlador de LCD, codecs de audio, controlador de touch screen; debido a la ausencia de memoria no volátil interna, estos dispositivos poseen periféricos dedicados al manejo de memorias no volátiles NAND flash, NOR flash, SPI, I2C y Secure Digital (SD); y memorias volátiles Synchronous dynamic random-access memory (SDRAM) y Double Data Rate (DDR); su velocidad de operación varía entre los 75MHz y 800MHz. En la figura 1.6 se muestra la arquitectura típica de un sistema basado en estos procesadores.

Debido a la falta de memoria volátil, las aplicaciones de este tipo de dispositivos requieren una memoria externa para almacenar las aplicaciones básicas y datos, en la actualidad las más populares son las memorias NAND flash, NOR flash, SPI, Electrically-Erasable Programmable Read-Only Memory (EEPROM) y SD. Normalmente, este tipo de procesadores son utilizados en aplicaciones que utilizan sistemas operativos, como se verá más adelante. Para que ciertos sistemas operativos (Linux, Windows CE) puedan ejecutarse se requiere una mínima cantidad de memoria RAM (del

³La Memory management unit (MMU) permite el manejo de memoria, dentro de sus funciones se encuentra el traslado de la memoria física a virtual, protección de la memoria, control de cache, control de buses

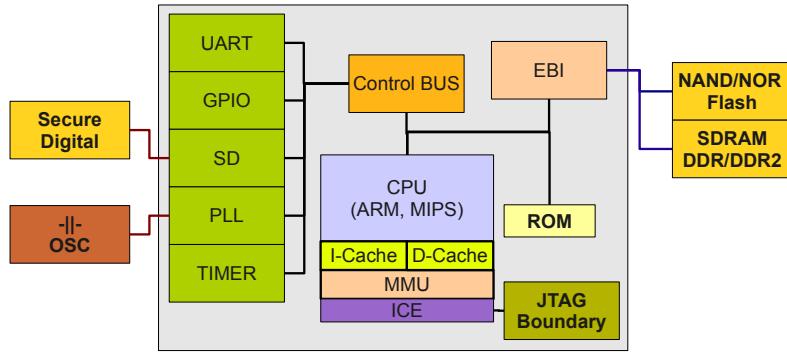


Figura 1.6: Arquitectura típica de un sistema embebido que utiliza SoC sin memoria volátil interna

orden de los Mbytes), por esta razón es necesario incluir una memoria RAM externa, en la actualidad las más utilizadas son las SDRAM, DDR y DDR2.

Como conclusión, se puede decir que en el mercado existen diferentes arquitecturas de SoCs que permiten realizar proyectos con diferentes grados de complejidad y que se ajustan a las opciones más utilizadas por los desarrolladores; la opción más económica es la utilización de un SoC que incluya las memorias no volátiles y RAM internamente; sin embargo, hasta el momento no existen dispositivos con grandes capacidades de memoria Flash y RAM internas, por lo que no es recomendado su uso en ciertas aplicaciones. Utilizar un SoC que no integre las memorias no volátiles proporciona una mayor flexibilidad, ya que estos dispositivos proporcionan periféricos que pueden controlar varios tipos de memorias, y se puede elegir la más económica, algo similar ocurre con la memoria RAM; sin embargo, el costo total de las memorias externas, SoC y área de circuito impreso es mayor que en el caso anterior. En la tabla 1.1 se resume la arquitectura de las plataformas utilizadas.

Tabla 1.1: Arquitectura de las plataformas comerciales utilizadas en la etapa de absorción

| Plataforma | CPU | Mem. volátil | Mem. no volátil | MMU | OS |
|-------------|--------------|--------------|-----------------|-----|-------------|
| Game Boy | ARM 7 | Externa NOR | Interna | NO | Propietario |
| Zaurus | Strong ARM | Externa NAND | Externa SDRAM | SI | Linux |
| iPAQ H3600 | Strong ARM | Externa NAND | Externa SDRAM | SI | Windows CE |
| Chumby | ARM MC94MX21 | Externa NAND | Externa SDRAM | SI | Linux |
| Ainol V2000 | MIPS JZ4740 | Externa NAND | Externa SDRAM | SI | Linux |
| SUNGALE DPF | MIPS JZ4740 | Externa NAND | Externa SDRAM | SI | Linux |
| B&N NOOK | ARM S3C6410 | Externa SD | Externa SDRAM | SI | Android |

Aunque estos procesadores operan a velocidades entre los 75 y 800 MHz, no todos los componentes del SoC operan a esta frecuencia, el componente externo que requiere la mayor velocidad de operación es la memoria RAM y puede estar entre los 50 y 130 MHz, los demás periféricos funcionan a frecuencias del orden de las decenas de MHz o KHz; por esta razón estos SoC suministran un circuito PLL que permite generar la frecuencia de operación a partir de cristales de frecuencias del orden de las decenas de MHz, lo que facilita el diseño de la placa de circuito impreso.

Cada periférico requiere una conexión específica con el dispositivo que controla, los SoC

modernos incluyen la mayor parte del circuito internamente con el objetivo de minimizar las conexiones y dispositivos adicionales. Existen tendencias de los fabricantes a agrupar periféricos teniendo en mente dos aplicaciones: multimedia, e industriales; para aplicaciones multimedia se proporcionan controladores de LCDs, ratón, teclado, pantalla táctil, codificador-decodificador (CODEC)s de audio, control de potencia, relojes de tiempo real, control de carga de baterías entre otros; para aplicaciones industriales se proporcionan controladores de red cableada, puertos Controller area network (CAN), I2C, y SPI.

Programación

Como se mencionó anteriormente, para este estudio se utilizaron herramientas abiertas para la creación de aplicaciones, en la figura 1.7 se muestra el flujo de creación de las tareas software usando la cadena de herramientas GNU is Not Unix (GNU) [13]. La ventaja de utilizar estas herramientas (adicional a la económica) es el soporte a diferentes procesadores (24 diferentes CPUs, incluyendo micro-controladores de 8 bits), lo que permite la fácil migración entre CPUs; adicionalmente su alto grado de configurabilidad permite el cambio de disposición de las memorias volátiles y no volátiles de forma fácil (a través del script de enlace). El proceso de generación del archivo binario que debe ser grabado en la memoria no volátil de la plataforma puede ser realizado en su totalidad por la cadena de herramientas GNU.

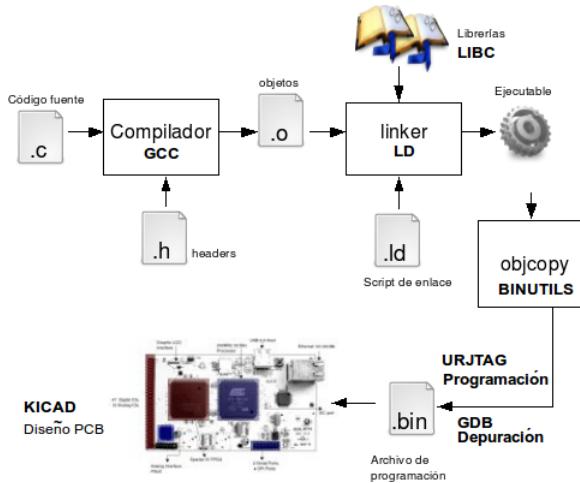


Figura 1.7: Flujo de diseño software para creación de aplicaciones.

Los SoC poseen la capacidad de *iniciar* desde diferentes dispositivos; cuando se activa la señal de *reset* a un SoC que no posee memoria volátil interna, el primer programa en ejecutarse es el que reside en una memoria Read Only Memory (ROM) interna, este programa revisa varios periféricos en búsqueda de un programa válido; los periféricos soportados varían según el fabricante, pero por lo general siempre soportan el uso de memorias NOR Flash (paralelas) y en SoCs más recientes memorias NAND Flash, SPI, o SD; sin embargo, la mayoría de SoC soportan memorias que se encuentran soldadas en la placa de circuito impreso, lo que hace necesario buscar métodos de programación de estas memorias que no implique desmontarlas o el uso de costosos conectores. En la mayoría de los SoC, cuando el programa residente en la ROM no encuentra ninguna aplicación válida

en los periféricos soportados, establece una comunicación por uno de sus puertos seriales o USB y queda en espera del envío de un programa válido, el programa enviado es almacenado en la memoria RAM interna, y una vez finaliza su descarga se ejecuta desde la RAM interna. La figura 1.8 muestra este proceso.

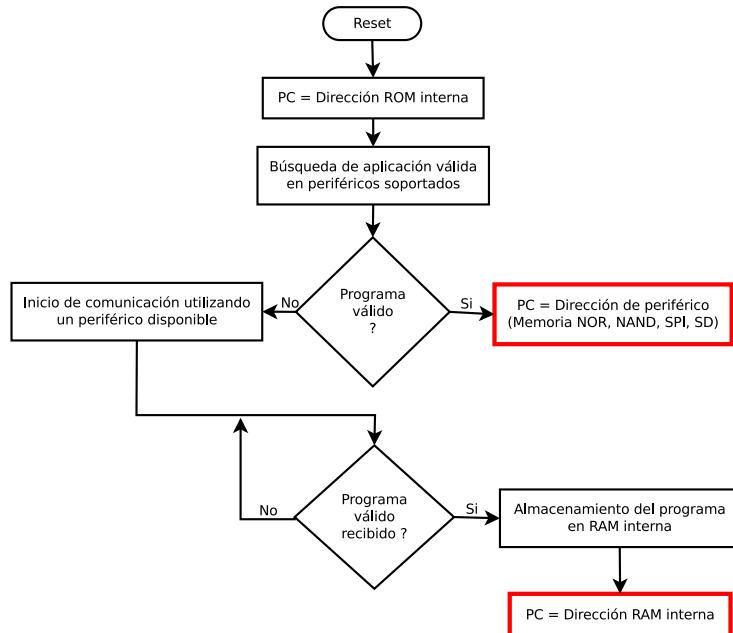


Figura 1.8: Inicialización de un SoC cuando las memorias no volátiles no están programadas.

Debido a que la RAM interna normalmente es pequeña (del orden de decenas de Kbytes), no es posible cargar aplicaciones muy grandes en ella, por lo que es necesario realizar el proceso de programación en varias etapas: en la primera etapa se carga una aplicación (*first - stage bootloader*) que se encarga de configurar el procesador (pila, frecuencia de operación), configurar la memoria RAM externa y habilitar un canal de comunicación para descarga de aplicaciones, de esta forma, es posible almacenar aplicaciones tan extensas como la capacidad de la memoria RAM externa (del orden de MBytes). En la segunda etapa se descarga una aplicación a la memoria externa que tiene la capacidad de programar las memorias no volátiles externas con la información proveniente de los diferentes periféricos de comunicación del SoC (como puerto serial, memoria SD, USB), este segundo programa recibe el nombre de *bootloader* y se auto-almacena en las primeras posiciones de la memoria no volátil, de tal forma que sea ejecutado después de la activación de la señal de *reset* y de la búsqueda que realiza el programa interno de la ROM.

Una vez programada la memoria no volátil con una aplicación válida, los SoCs realizan una serie de pasos para ejecutar las aplicaciones almacenadas en ella, esto debido a la poca capacidad de la memoria RAM interna. Como se dijo anteriormente, una vez se activa la señal de *reset* se ejecuta un programa contenido en la memoria ROM interna del SoC (figura 1.9 (a)), esta aplicación configura un periférico que permite la comunicación con los dispositivos de almacenamiento masivo externos, y además copia una determinada cantidad de información desde la memoria no volátil externa a la memoria RAM interna (figura 1.9 (b)), esto se hace porque el programa en la ROM no conoce la configuración de la plataforma y esta puede cambiar según la aplicación; después de esto ejecuta

la aplicación copiada a la memoria RAM interna colocando en el contador de programa (Program Counter (PC)) el valor correspondiente a la memoria RAM interna (figura 1.9 (c)).

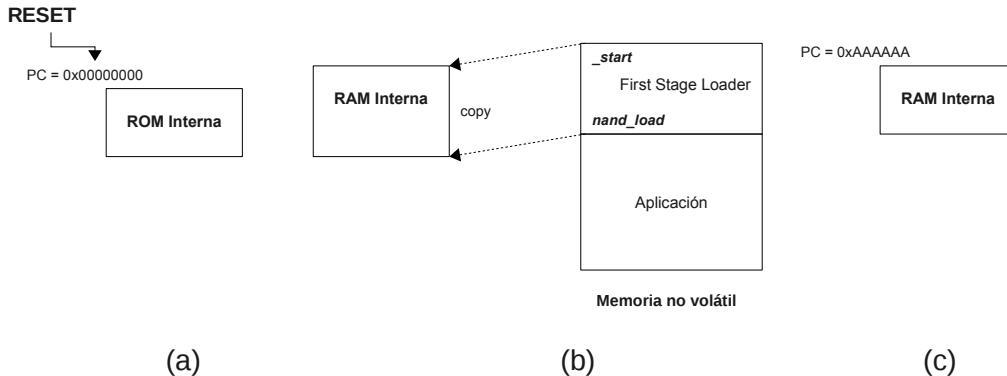


Figura 1.9: Inicialización de un SoC cuando la memoria no volátil está programada, parte 1.

Este programa (*loader*) está encargado de: configurar la memoria RAM externa (su capacidad varía dependiendo de la aplicación) y de copiar la aplicación propiamente dicha desde la memoria no volátil a la memoria RAM externa, (con lo que es posible cargar aplicaciones de mayor tamaño que la memoria RAM interna); finalmente, el *loader* ejecuta la aplicación almacenada en la memoria RAM haciendo que el contador de programa (PC) sea igual a la dirección donde se almacenó esta aplicación (ver figura 1.10)

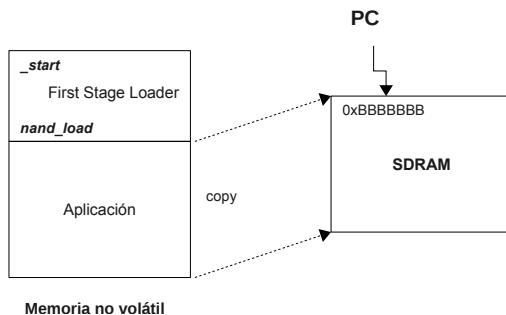


Figura 1.10: Inicialización de un SoC cuando la memoria no volátil está programada, parte 2.

Programación utilizando el puerto JTAG

Algunos SoCs no suministran un camino para la programación de la memoria RAM interna, para estos casos, se puede utilizar un periférico que la mayoría de los dispositivos proporciona: el puerto Joint Test Action Group (JTAG) (creado inicialmente como un mecanismo para realizar pruebas en las tarjetas de circuito impreso para verificar la correcta conexión entre componentes, y verificar el correcto funcionamiento de los circuitos integrados) el cual, está formado por un registro de desplazamiento (ver figura 1.11) que controla el paso de información desde y hacia cada uno de los pines del circuito integrado, permitiendo realizar varias operaciones. Con el paso del tiempo, se han

adiccionado funcionalidades a este protocolo y una de ellas es el control de circuitos especializados dentro de los SoCs para realizar emulación en circuito (In Circuit Emulation (ICE)), suministrando un canal para la programación de la memoria RAM interna.

Algunos SoCs antiguos no poseen una unidad de emulación en circuito por lo que no es posible acceder a la memoria RAM interna, en estos casos es posible utilizar el protocolo JTAG para controlar directamente los pines del SoC conectados a las memorias no volátiles y ejecutar los protocolos de programación de las mismas; debido a que es necesario programar todos los registros de la cadena Boundary Scan, el tiempo de programación suele ser más largo que cuando se utiliza el ICE.

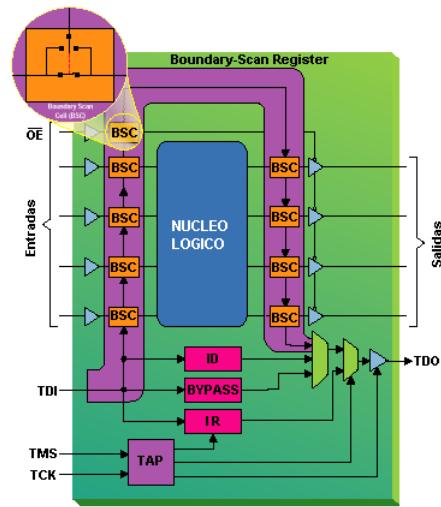


Figura 1.11: Cadena Boundary Scan fuente: Texas Instruments.

Aplicaciones *standalone* vs aplicaciones con sistema operativo

Los sistemas operativos proporcionan facilidades al programador que permiten acelerar el desarrollo de aplicaciones, suministrando una capa de abstracción de hardware que permite manejar los periféricos a alto nivel sin preocuparse por el manejo tedioso a nivel de registros; adicionalmente, proporciona soporte para aplicaciones en red, manejo de sistemas de archivos, multitarea, seguridad, entre otras (ver figura 1.12; adicionalmente, existen librerías especializadas que ayudan al desarrollo en diferentes áreas. Sin embargo, el uso de sistemas operativos como Linux, Android, Mac OS o Windows, exige el cumplimiento de condiciones mínimas para su uso; por ejemplo, Linux requiere 8 Mbytes de memoria RAM y 2 Mbytes de memoria no volátil, Android requiere 128 Mbytes de memoria RAM y 32 Mbyte de memoria no volátil; por esta razón es necesario agregar dos memorias externas, lo que aumenta la complejidad de la placa de circuito impreso y el costo del dispositivo. Por otro lado, los sistemas operativos tienen una particularidad en su funcionamiento que recibe el nombre de *latencia*; y se define como el tiempo que transcurre entre la generación de un evento (interrupciones hardware o software) y la respuesta ante este evento, este tiempo varía según el estado de carga del sistema; en un sistema operativo de tiempo real esta latencia es conocida y no depende de la carga de sistema. Esta latencia en algunas aplicaciones hace imposible el manejo de eventos ya que es posible que se pierdan algunos cuando el sistema se encuentre muy cargado.

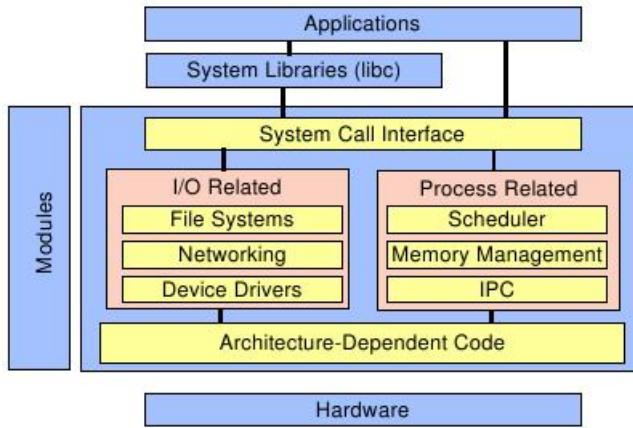


Figura 1.12: Estructura del kernel de Linux.

Las aplicaciones *standalone* utilizan los recursos necesarios y dependiendo de su complejidad pueden ajustarse a un SoC que incorpore la memoria RAM y no volátil internamente; adicionalmente, su programación puede reducir el problema que se presenta en la latencia a las interrupciones en los sistemas operativos. Sin embargo; es necesario dar soporte a todos los periféricos que se utilizarán y se deben escribir todas las rutinas, lo que puede aumentar el tiempo de desarrollo.

En conclusión, el uso de sistemas operativos o aplicaciones standalone depende de la complejidad de la aplicación, y de consideraciones económicas como el *time to market* y costo de los desarrolladores.

Conocimientos adquiridos

En la tabla 1.2 se muestran los conocimientos adquiridos al aplicar ingeniería inversa a las plataformas comerciales utilizadas en esta etapa.

Tabla 1.2: Conocimientos Adquiridos durante la etapa de adaptación

| Plataforma | Herramientas de desarrollo | Programación | Sistema Operativo y Aplicaciones |
|-------------|----------------------------|--------------------|---|
| Game Boy | ARM GNU toolchian | Puerto serie | eCos, implementación de periféricos en FPGAs. |
| Zaurus | ARM GNU toolchian | Puerto serie | Linux 2.4, sistema de archivos, QT. |
| iPAQ H3600 | ARM GNU toolchian | Puerto serie | Linux 2.4, Buildroot, QT. |
| Chumby | ARM GNU toolchian | Puerto serie | Linux 2.6, u-boot, Open-Embedded, QT, flash. |
| Ainol V2000 | MIPS - ELF GNU toolchian | Puerto serie | Linux 2.6, openwrt, QT. |
| SUNGALE DPF | MIPS - ELF GNU toolchian | Puerto serie | Linux 2.6, openwrt, QT. |
| B&N NOOK | ARM GNU toolchian | Puerto JTAG, serie | Linux 2.6, Android Dalvik (VM). |

Discusión

En la etapa de adopción se utilizaron herramientas comerciales para aprender: como están construidos; como funcionan y como se programan dispositivos comerciales; su utilización facilita el aprendizaje del funcionamiento de la tecnología, permitiendo el estudio de metodologías de diseño y la evaluación de herramientas de desarrollo, reduciendo el tiempo y el costo de desarrollo de plataformas propias diseñadas para dicho fin; adicionalmente, pueden ser utilizadas como base de nuevos productos comerciales. Sin embargo, se notaron los siguientes inconvenientes durante su utilización: al ser diseñadas para una aplicación específica son difíciles de adaptar a nuevas aplicaciones; los fabricantes no proporcionan documentación sobre su arquitectura; su uso crea dependencia hacia el fabricante y puede reducir la demanda de diseñadores y proveedores de bienes y servicios de manufactura; solo pueden utilizarse en aplicaciones similares para las que fueron creadas.

En esta etapa se utilizaron plataformas de desarrollo comerciales diseñadas para proporcionar herramientas en el estudio de un determinado procesador, en la mayoría de los casos utilizan la mayor parte de sus recursos (del procesador) y ponen a disposición del usuario una gran cantidad de periféricos, conectores y señales, lo que facilita el aprendizaje y su uso en aplicaciones comerciales. Sin embargo, su costo es relativamente alto (en comparación con los dispositivos comerciales de alto nivel de producción); debido a su gran oferta de conexiones son tarjetas muy grandes y difíciles de adaptar en algunas aplicaciones reales; por otro lado, es necesario fabricar tarjetas adicionales para que se adapten a la aplicación lo que incrementa el costo final.

1.2.4. Absorción y Asimilación

En esta etapa se realizan actividades que integran nuevo conocimiento para el país pero que no es nuevo para el mundo y están encaminadas al desarrollo o adaptación de metodologías de diseño y procesos de fabricación, desarrollo de productos tecnológicos propios, enseñanza de metodologías de diseño y procesos de fabricación.

La utilización de plataformas de desarrollo o dispositivos comerciales como punto de partida para nuevos productos es viable en casos donde solo se requiera un cambio de software, ya que adiciones en el hardware aumentarían el costo total o simplemente no se pueden hacer debido a que no se tiene acceso a los archivos de diseño. Para poder ser competitivos, es necesario reducir los costos de componentes y de fabricación, para lograr esto es necesario utilizar la mínima cantidad de componentes para la aplicación, lo que implica la producción de placas de circuito impreso (PCBs). El país no cuenta con una buena oferta para la producción de PCBs, solo existen tres empresas que pueden ofrecer productos que cumplen con los requerimientos: Microensamble en Bogotá, Colcircuitos en Medellín y Microcircuito en Cali, de los cuales, solo las dos primeras proporcionan el servicio de tarjetas de 1 a 10 capas.

El diseño de PCBs es una actividad que da empleo a profesionales de diferentes niveles de formación (técnicos, ingenieros, diseñadores) en los países asiáticos; desafortunadamente en Colombia existen muy pocas empresas que proporcionan este servicio, lo que evidencia la falta de demanda interna en las actividades relacionadas con la manufactura de dispositivos digitales; adicionalmente, en la mayoría de las universidades no se enseñan normas para el diseño de PCBs ni se somete a los estudiantes a este proceso de diseño, aceptando soluciones intermedias como el uso de placas de proyectos (project boards) o baquelitas universales o simplemente se ignora la parte de implementación física y el último paso es la simulación. Por estos motivos, una parte importante en esta etapa es el diseño y construcción de tarjetas de desarrollo que serán utilizadas para explorar

las técnicas de fabricación y montaje y posteriormente ser utilizadas en el desarrollo de aplicaciones académicas.

Plataformas de desarrollo

La primera plataforma desarrollada en este proyecto utiliza el SoC AT91R40008 de ATMEL y se le dió el nombre de ECB_ARM7, el diagrama de bloques de la plataforma y la foto del prototipo se muestra en la figura 1.13, ECB_ARM7 posee componentes de montaje superficial y fue el punto de partida para el estudio de técnicas de fabricación de placas de circuito impreso, con ella se realizaron aplicaciones *standalone* y con el sistema operativo eCos. Dentro del proceso de diseño de sistemas digitales, algunas de las tareas deben ser implementadas en hardware para cumplir con los requerimientos temporales; por este motivo es importante realizar implementaciones de tareas hardware en dispositivos lógicos programables (PLDs); para estudiar la forma de comunicar estas tareas hardware con SoC comerciales se diseñó un clon de la tarjeta comercial XPORT de la empresa norteamericana Charmedlabs, que permite conectarse con el procesador de la plataforma comercial ©GameBoy Advance de Nintendo [14] (ver figura 1.14).

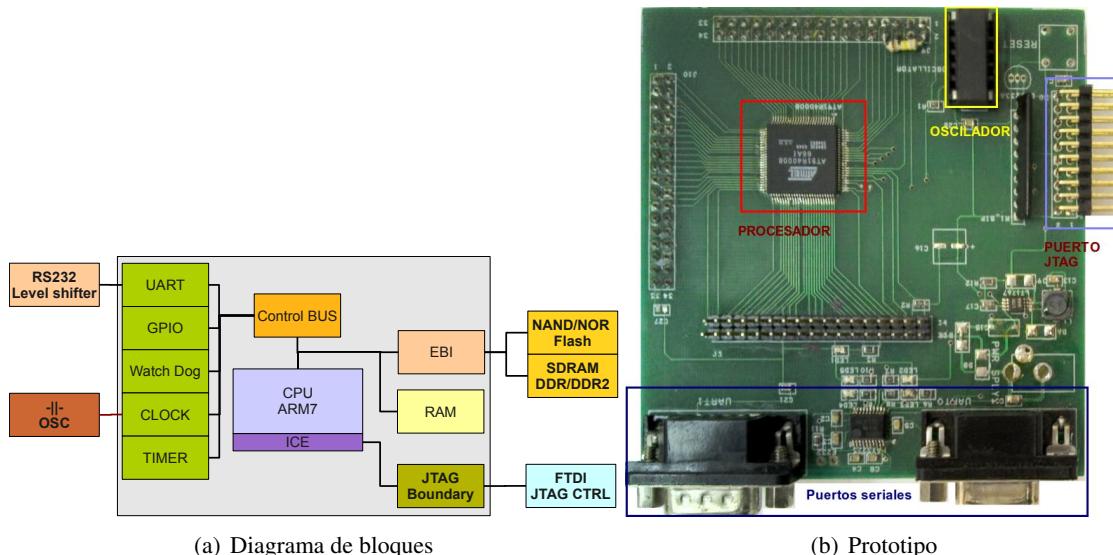


Figura 1.13: Plataforma de desarrollo ECB_ARM7

El uso de Linux como herramienta de desarrollo ha aumentado en los últimos años, llegando a ser el sistema operativo más utilizado por los desarrolladores (ver figura 1.4); empresas como Nokia, Motorola, Google, Dlink, Hewlett Packard utilizan Linux para desarrollar el firmware de sus dispositivos; por este motivo, se diseñaron plataformas que soportan la ejecución de este sistema operativo. Se diseñó una familia de plataformas denominadas ECB_AT91 V1 (ver figura 1.15 [15] [16]), ECB_AT91 V2 (ver figura 1.16) y ECBOT (ver figura 1.17 [17] [18] [19]); esta familia de plataformas se encuentran registradas en el kernel oficial de Linux, por lo tanto, su soporte queda garantizado en futuras versiones del kernel. ECB_AT91 V1 fue el punto de partida para el estudio de técnicas de fabricación de placas de circuito impreso para SoC que operan a velocidades mayores de 180MHz, y para aprender el proceso completo de adaptar el kernel de Linux a una plataforma nueva, el proceso detallado se muestra en el apéndice ???. En la figura 1.17 (a) se muestra el diagrama

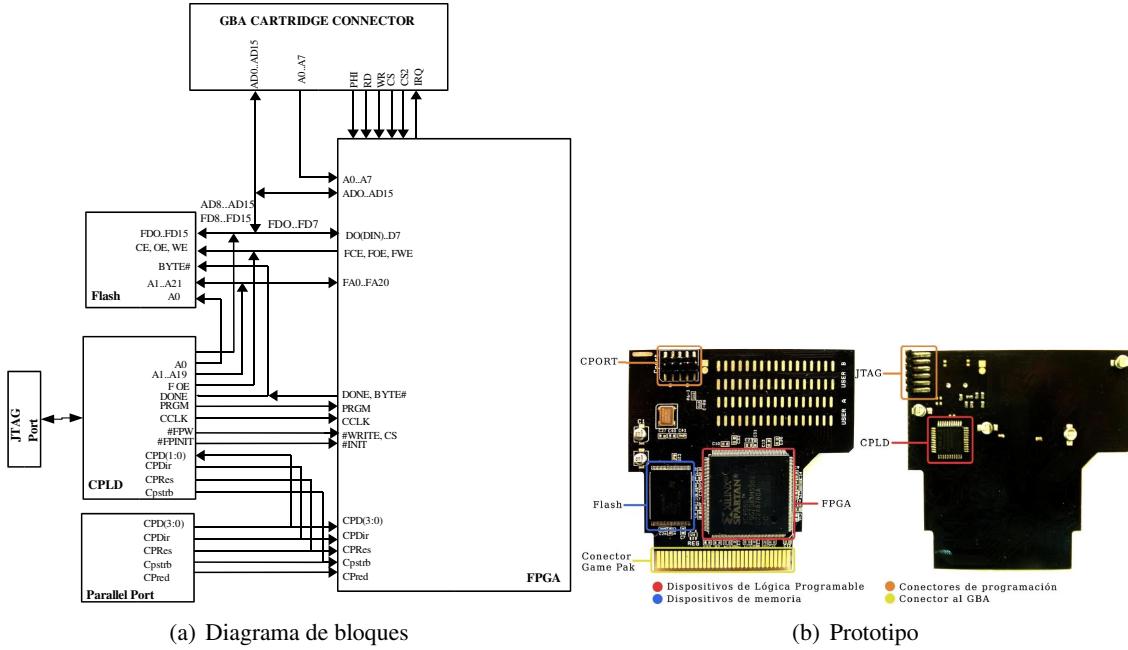


Figura 1.14: Plataforma de desarrollo UNAL_UIS_XPORT

de bloques de estas tres arquitecturas; todas poseen el SoC de 180MHz de Atmel AT91RM9200, debido a que este SoC no posee conversores análogos - digitales internos fue necesario incluir un micro-controlador de 8 bits que se comunica con el SoC vía I2C, el SoC realiza la programación del microcontrolador emulando un puerto paralelo en sus pines de entrada/salida de propósito general (General Purpose Input/Output (GPIO)s) que van conectados a los pines del puerto In System Programming (ISP) del microcontrolador.

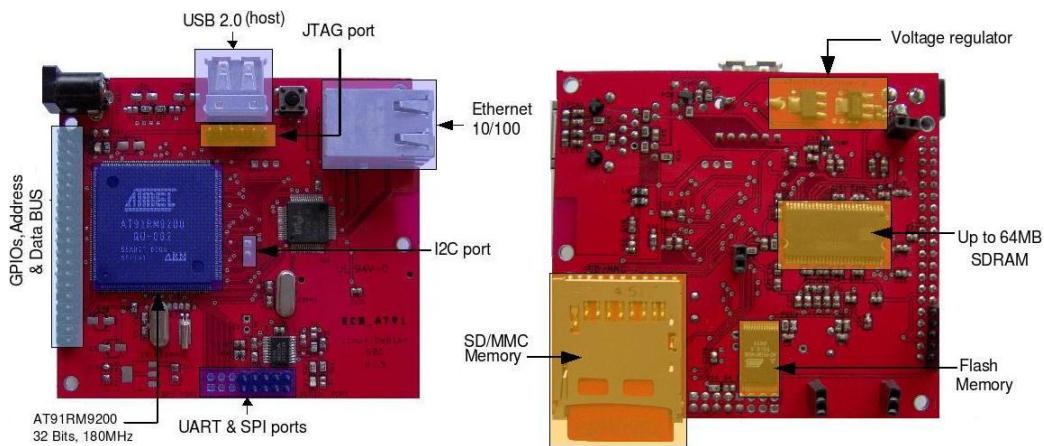


Figura 1.15: Plataforma de desarrollo ECB_AT91 V1

La plataforma ECB_AT91 V2 es el resultado del interés de utilizar dispositivos lógicos programables y SoC comerciales para crear periféricos dedicados y la creación de nuevos drivers de Linux que los controlaran, su arquitectura es idéntica a la de la plataforma ECB_AT91 V1 pero

el bus de datos, dirección y control del SoC se conectan a la FPGA para permitir la comunicación con los periféricos implementados en ella; la FPGA es configurada por el SoC utilizando unos GPIOs que van conectados al puerto JTAG de la FPGA, lo que elimina la necesidad de cables adicionales de conexión, permitiendo su re-programación total de forma remota.

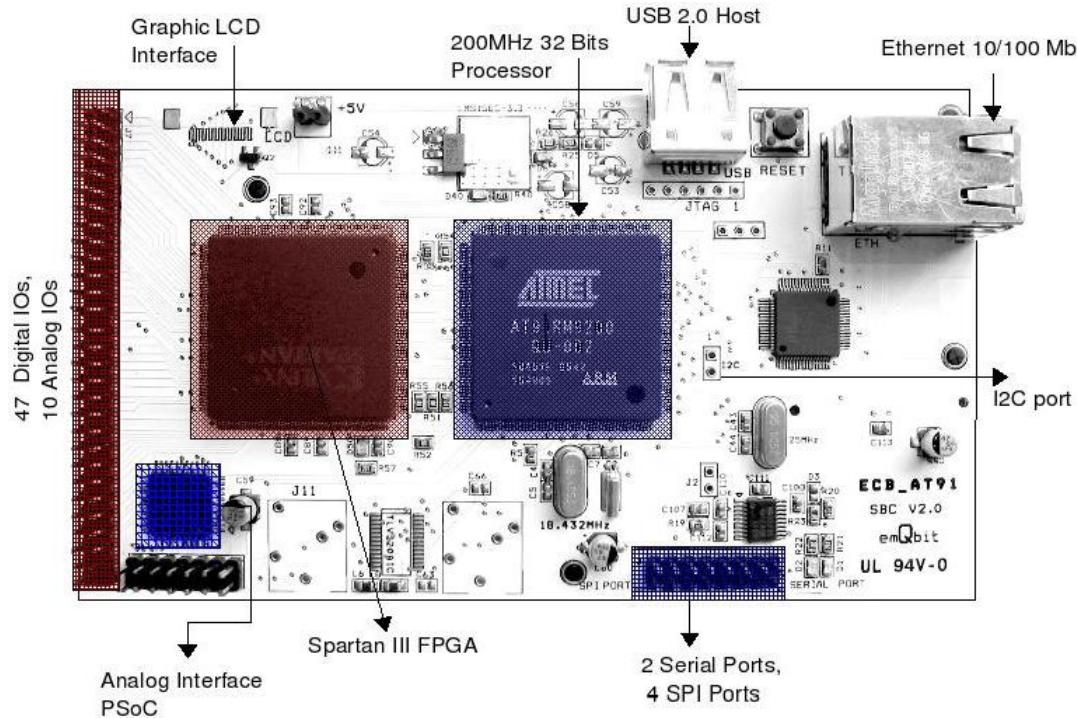


Figura 1.16: Plataforma de desarrollo ECB_AT91 V2

La plataforma ECBOT fue diseñada con el propósito de iniciar el estudio en robótica móvil y procesamiento de imágenes; para esto, se utilizó una arquitectura similar a la de la plataforma ECB_AT91_V2 adicionando una conexión dedicada a un sensor de imagen y circuitos especializados para el control de sensores (6 sensores de choque, 1 sensor de imagen, movimiento de los motores basado en BEMF⁴), actuadores (8 LEDs RGB, 2 motores DC) y un conversor DC/DC que garantiza el máximo uso de las baterías. Como parte de este trabajo, se realizó la adaptación de proyectos libres que facilitan la investigación en robótica como *player/stage*⁵ y *openblocks*⁶. En la actualidad ECBOT está siendo usado por los grupos de robótica de los departamentos de Sistemas y mecatrónica de la Universidad Nacional de Colombia.

Los SoC utilizados en las plataformas anteriores no pueden ser utilizados en aplicaciones de procesamiento de señales o de imágenes en tiempo real (a menos que se implemente algún periférico en la FPGA), ya que su arquitectura no está diseñada para este fin. Los DSPs son procesadores especializados para ejecutar las operaciones más comunes en procesamiento digital de señales; pensando en este tipo de aplicaciones se diseñó la plataforma ECB_BF532 [20] que utiliza el DSP BF532 de la familia Blackfin de Analog Devices. Esta plataforma será utilizada en los cursos

⁴Back Electro Magnetic Force

⁵<http://playerstage.sourceforge.net/>

⁶<http://dspace.mit.edu/handle/1721.1/41550>

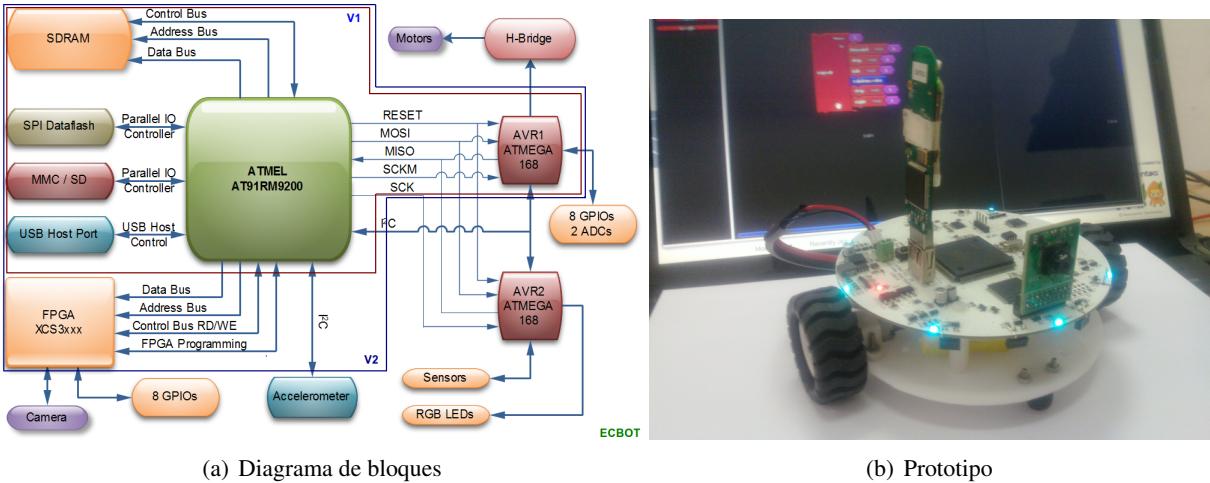


Figura 1.17: Plataforma de desarrollo ECBOT

de procesamiento digital de señales del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia.

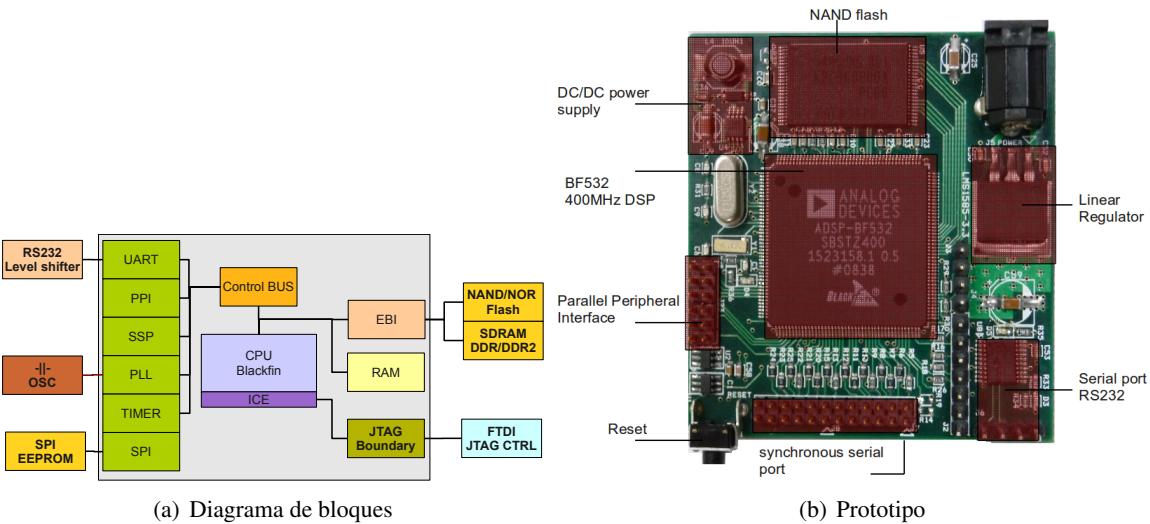


Figura 1.18: Plataforma de desarrollo ECB_BF532

La plataforma SIE[21], permite el desarrollo de aplicaciones gráficas; esta compuesta por un procesador MIPS fabricado por la compañía china Ingenic y por una FPGA (Spartan 3E de Xilinx) lo que posibilita la implementación de tareas hardware y la creación de controladores específicos en el sistema operativo Linux. En la actualidad esta pataforma esta siendo utilizada en tres de los cuatro cursos de la línea de electrónica digital en el Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia. En el capítulo ?? se realizará una detallada explicación del uso de SIE en la enseñanza de diseño de sistemas embebidos.

La plataforma STAMP (se encuentra en proceso de pruebas al momento de escribir este documento) es el resultado del interés por el estudio del sistema operativo Android; debido a los

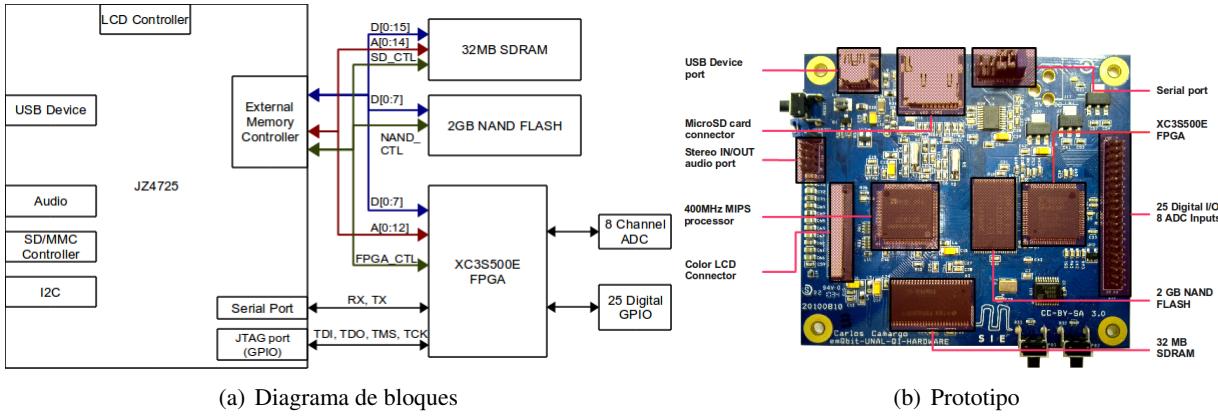


Figura 1.19: Plataforma de desarrollo SIE

requerimientos de memoria de este sistema operativo, las plataformas anteriormente diseñadas no pueden ejecutar aplicaciones Android. Por este motivo se utilizó el SoC de Freescale iMX233 que incluye el controlador de memorias DDR (las que son más económicas y de mayor densidad que las SDRAM utilizadas en las plataformas existentes); adicionalmente, este SoC permite inicializar desde una memoria SD, lo que elimina la necesidad de utilizar memorias NAND, reduciendo de forma considerable el costo. En la figura 1.20 se puede observar el diagrama de bloques y una foto del prototipo de esta plataforma.

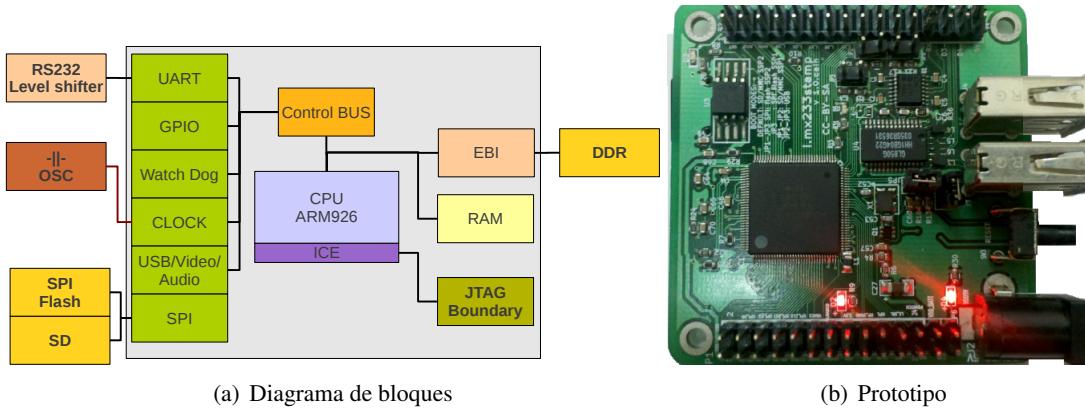


Figura 1.20: Plataforma de desarrollo STAMP

Técnicas de fabricación

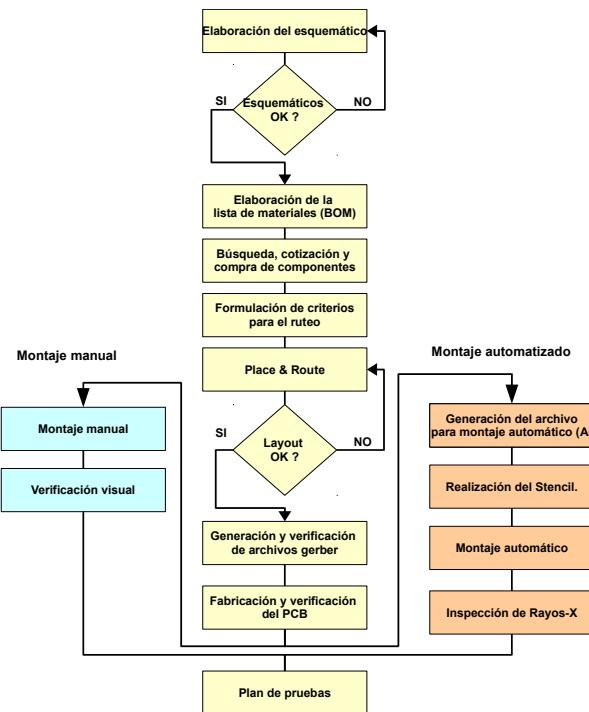
Durante el proceso de concepción, diseño, implementación y operación de estas plataformas se trabajó con diferentes proveedores de bienes y servicios nacionales y extranjeros, y se utilizaron diferentes técnicas para su construcción y montaje; en la tabla 1.3 se resumen las principales características de cada plataforma: CPU utilizada, número de capas del PCB, tipo de montaje: manual o automático, cantidad de unidades producidas, sistema operativo (OS) estudiado y adaptado totalmente a la plataforma y finalmente las universidades que utilizan estas plataformas.

Tabla 1.3: Características de las plataformas de desarrollo concebidas, diseñadas e implementadas

| Plataforma | CPU | Capas | Montaje | Cant. | OS | Usuario |
|--------------|---------------|-------|---------------------|-------|---------|---------------------------------|
| ECB_ARM7 | ARM7,33M | 2 | local Manual. | 2 | eCos | UN |
| UN_UIS_XPORT | ARM7,50M | 2 | local Manual. | 2 | eCos | UN, UIS |
| ECB_AT91_V1 | ARM920,180M | 2 | local Manual/Autom. | 100 | Linux | UN, UIS, ULA, ENAP, UDFJC, USTA |
| ECB_AT91_V2 | ARM920 180M | 4 | local Manual. | 30 | Linux | UN, UIS, ULA, ENAP, UDFJC |
| ECBOT | ARM920 180M | 4 | local Manual. | 20 | Linux | UN, UIS |
| ECB_BF532 | Blackfin 400M | 4 | local Manual. | 5 | uCLinux | UN |
| SIE | MIPS32 300M | 2 | externo Autom. | 80 | Linux | UN, UIS, ULA, ECI |
| STAMP | ARM926 454M | 2 | local Manual. | 2 | Android | UN |

Proceso de diseño y fabricación

La figura 1.21 muestra los pasos que se siguieron en la elaboración de las plataformas de desarrollo; esta metodología puede ser utilizada para la realización de cualquier nivel de producción; el montaje de prototipos puede realizarse de forma manual o automática, dependiendo de los recursos económicos, de la disponibilidad de equipo y de personal especializado.

**Figura 1.21:** Flujo de diseño para las placas de circuito impreso

En esta etapa se adquirieron las siguientes habilidades:

- Concepción, diseño, implementación y operación de tarjetas electrónicas que utilizan SoC de 32 bits.

Técnicas de diseño de placas de circuito impreso.

Técnicas de montaje manual y automático.

Metodología para el diseño e implementación de tarjetas electrónicas.

Diseño de protocolo de pruebas.

- Adaptación del sistema operativo Linux a una nueva plataforma.

Aplicaciones académicas realizadas

Durante esta etapa (que se realiza de forma continua cada vez que se desea incluir un nuevo tema de estudio) se desarrollaron una serie de trabajos de grado a nivel de pregrado en las universidades Nacional de Colombia, de los Andes e Industrial de Santander y se diseñó un nuevo curso llamado *Sistemas Embebidos* que fue dictado en varias ocasiones como electiva en la Universidad de los Andes y es un curso optativo en la Universidad Nacional de Colombia. Se ayudó a dos egresados de la carrera de Ingeniería Electrónica en la creación de la empresa *EM Electrónica*, proporcionando la información necesaria para la construcción y modificación de tarjetas de desarrollo de dispositivos lógicos programables (CPLDs y FPGAs), las que fueron utilizadas en los cursos de la línea de electrónica digital en la Universidad Nacional. Como resultado de la utilización de estas plataformas se eliminó el uso de componentes discretos (familias lógicas TTL y CMOS 40XX y 74XX) y se comenzó el uso de lenguajes de descripción de hardware desde el primer curso de la línea. A continuación se listan los trabajos realizados durante esta etapa:

- Trabajos de grado: UIS: [14], UNAL:[22] [23] [24] [25] [26]
- Tesis de Maestría UN: [27] [28]
- Artículos [29] [30] [31] [32] [33] [34] [18] [21] [35] [36] [37] [38] [39] [19]

Todos estos productos académicos tienen como característica el desarrollo de placas de circuito impreso y la utilización de los conocimientos adquiridos durante este estudio. Inicialmente, las placas de circuito impreso fueron diseñadas con herramientas comerciales debido a la falta de una herramienta abierta con las características necesarias para realizar el proceso completo. En los últimos dos años el desarrollo de la herramienta *Kicad* permitió la eliminación de estas herramientas propietarias; a partir de ese momento solo se utilizaron herramientas abiertas en todo el flujo de diseño.

Metodología para la apropiación de conocimiento y generación de nuevos productos

En la figura 1.22 se resumen los pasos que se siguieron con las diferentes arquitecturas (CPUs) estudiadas; estas actividades conforman una metodología que minimiza el costo de inversión al utilizar software libre y productos de consumo masivo; adicionalmente, puede ser aplicada a cualquier dispositivo comercial y la curva de aprendizaje es tal que los pasos en las últimas CPUs estudiadas se realizaron en menos de un mes, mientras que la primera tomó cerca de 1 año.

Discusión

La fabricación de placas de circuito impreso a la medida permite reducir el costo final de un dispositivo digital; en su proceso de diseño y fabricación intervienen diferentes personas con diversos niveles de formación, lo que la hace muy atractiva como actividad generadora de empleo. La situación actual de las empresas manufactureras del país (solo 2) no permite la producción a grandes escalas

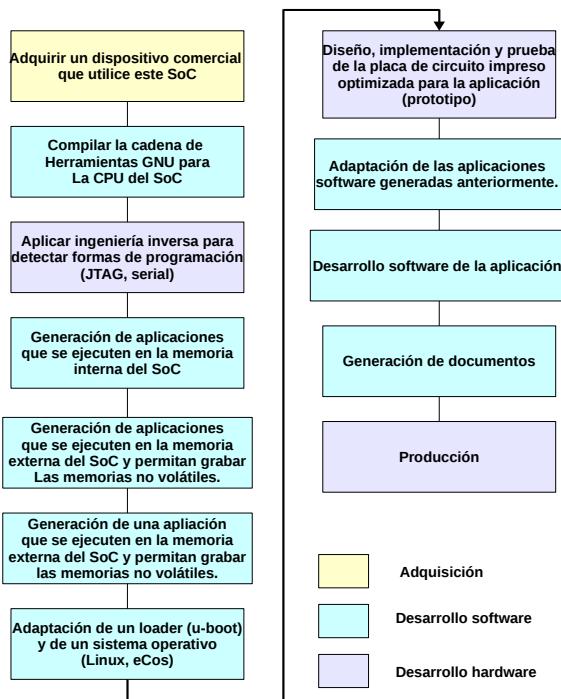


Figura 1.22: Flujo de actividades para la apropiación de conocimiento

ya que en el país no se cuenta con empresas que suministren componentes electrónicos a precios competitivos (los impuestos locales producen un sobre-costo de cerca del 26 %). En el país se pueden construir prototipos de muy buena calidad, por lo que la mejor opción es la fabricación local del prototipo y la producción en masa en el exterior. Esto mientras se crea en el país la demanda suficiente para poder realizar el proceso completo de forma local.

1.2.5. Aplicación

En esta etapa se aplican los conocimientos adquiridos previamente a la academia y a la industria, con el fin de evaluar su impacto. En la academia se utilizó la posición del investigador principal de este trabajo como docente del departamento de ingeniería eléctrica y electrónica de la Universidad Nacional de Colombia, para modificar las asignaturas del área de electrónica digital incorporando los temas recientemente absorbidos y asimilados. En el sector industrial , se formó una empresa de base tecnológica y se brindó asesoría en la adecuación de las plataformas *hardware copyleft* para convertirlos en productos comerciales.

En la academia

Se diseñó un nuevo programa académico que sigue los lineamientos de la iniciativa Concebir, Diseñar, Implementar, Operar (CDIO); en el capítulo ?? se detallan las características de este programa, el cual fue implementado en el Departamento de ingeniería eléctrica y electrónica de la Universidad Nacional de Colombia en las asignaturas del área de electrónica digital con el fin de formar profesionales con las habilidades necesarias para desarrollar productos comercializables basados en esta tecnología, el proceso de aplicación duró dos años y actualmente es el programa

oficial en dicha institución; los cambios generados debido a su aplicación son sorprendentes y se evidencian en la calidad de los trabajos de grado donde se pasó del uso de tabletas experimentales o proto-boards, placas de prototipado o stripboards y micro-controladores de 8 bits al uso de placas de circuito impreso diseñadas específicamente para la aplicación y el uso de SoCs de 32 bits y sistemas operativos. Estudiantes encuestados ⁷ sobre la pertinencia de esta nueva metodología de enseñanza manifiestan que concuerda con sus expectativas y se les proporcionan herramientas que pueden utilizar en el mundo real, motivándolos a crear nuevos productos que den solución a problemas reales ya sea como empleados en empresas ya constituidas o en nuevas empresas creadas por ellos.

En la industria

En el campo comercial, se trabajó con la empresa de base tecnológica emQbit, uno de los objetivos de trabajar con esta empresa era evaluar el impacto del uso de esta tecnología en la industria Colombiana; adicionalmente, emQbit proporcionó información sobre el estado de la industria electrónica en el país, ayudando a detectar los obstáculos que enfrentan las nuevas empresas en su ejercicio de suministrar soluciones a problemas locales. La mayor parte de sus productos utilizan como base las plataformas *hardware copyleft* diseñadas y distribuidas bajo el esquema de licencias CC BY-SA; el proceso de modificación se facilitó gracias a la disponibilidad de los archivos de diseño y al código fuente de las aplicaciones necesarias para la operación básica de las mismas. Estos productos están siendo documentados y podrán ser estudiados/utilizados/modificados por quien esté interesado. Es muy importante mencionar que (hasta donde llegan los conocimientos del autor) emQbit es la única empresa colombiana que realiza el proceso completo: diseño, construcción y programación de dispositivos electrónicos que utilizan tecnologías modernas (procesadores de 32 bits, sistema operativo Linux). En el desarrollo de sus actividades emQbit encontró las siguientes dificultades:

- Falta de personal con experiencia en desarrollo hardware y en creación de aplicaciones: Es notoria la carencia de habilidades relacionadas con el diseño y montaje de sistemas digitales en la mayoría de los profesionales entrevistados por esta empresa para ser parte de su grupo de desarrolladores; en general, desconocen el proceso de fabricación y las herramientas necesarias para diseñar una placa de circuito impreso, no son capaces de leer esquemáticos ni de buscar componentes adecuados para realizar una función determinada (lo que refleja la dependencia de la academia hacia las plataformas de desarrollo comerciales). Por otro lado, se detectan fallas de concepto o desconocimiento de partes del proceso de diseño, originado en algunos casos por el uso de herramientas propietarias que automatizan etapas de configuración de las plataformas.
- Escasa oferta de productos, bienes y servicios relacionados con la manufactura de sistemas digitales: Como se mencionó anteriormente, en Colombia solo existen 2 empresas con la capacidad de proporcionar los servicios necesarios para la producción, montaje automático y pruebas de placas de circuito impreso utilizando componentes modernos y con la calidad de las empresas asiáticas; desafortunadamente, sus costos son muy elevados en comparación con los de las industrias manufactureras asiáticas; esto se debe a la baja demanda de la industria colombiana y a los altos impuestos que deben pagar los dispositivos semiconductores y demás insumos del proceso (26 % o más). A pesar de estos inconvenientes, la fabricación de prototipos

⁷Encuestas realizadas por este estudio y la información suministrada por el sistema de evaluación de cursos de la Universidad Nacional

a nivel local es una buena opción desde el punto de vista económico, ya que la diferencia en costos no es muy grande, y es posible resolver dudas o problemas de forma ágil gracias a que se trabaja en la misma zona horaria.

- Desconfianza de la industria local por los productos del país: Los problemas descritos anteriormente generan un clima de desconfianza hacia los productos nacionales; ya que gran parte de ellos utilizan tecnologías obsoletas con procesos de fabricación de mala calidad, no cumplen con normas internacionales y no fueron sometidos a rigurosos procesos de pruebas. La desconfianza se presenta por la ausencia de productos locales, por el desconocimiento de los planes académicos de las universidades y por la falta de profesionales con la capacidad de diseñar y construir aplicaciones “a la medida” de las necesidades.
- Problemas relacionados con la financiación para desarrollo de nuevos productos: Es muy difícil encontrar entidades que financien el desarrollo de nuevos productos, la mayoría busca dispositivos terminados en fase de producción y no se encuentran interesados en invertir tiempo y dinero en nuevos desarrollos, esto es una consecuencia directa de la pérdida de capacidad de innovación en las empresas colombianas, donde no existen o fueron eliminados los departamentos de investigación y desarrollo (I+D).
- Competencia en desigualdad de condiciones con productos importados: Los impuestos a las importaciones de insumos para la fabricación de dispositivos digitales (más del 26 %), la poca oferta de bienes y servicios relacionados con la manufactura y la falta de políticas gubernamentales que protejan la industria electrónica nacional han permitido que los productos tecnológicos de los países asiáticos desplacen la producción local. Es importante que se formulen políticas que protejan los productos nacionales, sin provocar un desabastecimiento de productos que aún no se generan localmente, para esto es muy importante identificar los productos que son fabricados localmente y agrupar de forma adecuada a las empresas en la clasificación CODIGO INDUSTRIAL INTERNACIONAL UNIFORME (CIIU) en el registro de cámara y comercio.

Discusión

La adaptación de esta metodología al entorno académico reveló la deficiencia de los programas académicos del área de electrónica digital en la Universidad Nacional de Colombia, vale la pena mencionar que esta universidad ha ocupado los primeros puestos en los exámenes de estado *SABER PRO* (antes ECAES), por lo que es de esperarse que algunos programas académicos de otras universidades presenten problemas similares. La falla más notoria que se encontró en este proceso fue la ausencia de una metodología de diseño unificada durante la enseñanza de las asignaturas que componen el área; cada curso tenía su metodología y no existía una relación entre los contenidos de las asignaturas, lo que dejaba en los estudiantes la sensación de que se trataban de temas diferentes. Por otro lado, el uso de tecnologías obsoletas desviaba el enfoque del estudio a tediosas tareas manuales de minimización e implementación, restando importancia a la concepción y validación del diseño. Errores conceptuales en el diseño de los cursos hicieron que un lenguaje de descripción de hardware como el Verilog o VHDL fuera el centro de atención ignorando por completo temas relacionados con la arquitectura de computadores. Al final de estos dos cursos, los estudiantes no veían la utilidad práctica de los conocimientos adquiridos ya que nunca fueron enfrentados a la solución de problemas reales; esto, unido a la falta de herramientas de desarrollo hicieron que muchos

de ellos no consideraran la electrónica digital para dar solución de problemas. El último curso del área (micro-controladores) proporcionaba herramientas que permitían dar solución a problemas reales, por este motivo, los estudiantes opinaban que era la única útil de la línea; sin embargo, la falta de bases en arquitectura de computadores hacía que se pensara en el micro-controlador como una caja negra sin entender completamente su arquitectura, lo que limitaba el entendimiento del proceso de diseño.

Los cambios introducidos en estas asignaturas utilizan una única metodología de diseño, los problemas son atacados de la misma forma pero con herramientas cada vez más complejas pero más sencillas de utilizar. El uso de plataformas de desarrollo abiertas permite estudiar su arquitectura y entender completamente los flujos de diseño; adicionalmente, pueden ser utilizadas como punto de partida de desarrollos comerciales. Esto unido a la experiencia de diseño de circuitos impresos en 3 asignaturas, hizo cambiar completamente la mentalidad de los estudiantes sobre el uso de estas tecnologías, haciendo que realizaran búsquedas de problemas que puedan ser solucionados con estas herramientas, lo que constituye la base de la innovación. Todo esto representa una transferencia exitosa de esta investigación hacia la academia ya que se modificaron los hábitos existentes y se utilizaron los conocimientos generados por esta investigación; adicionalmente, se demostró la importancia del uso de esta tecnología.

Por otro lado, desde el punto de vista comercial, se comprobó que la universidad puede incentivar la creación de empresa, siempre y cuando se proporcionen herramientas para que puedan crear productos novedosos, y se brinde soporte técnico para su utilización. Sin embargo, los problemas mencionados anteriormente dificultan su operación, lo que permite concluir que es necesario una alianza entre este tipo de empresas, las universidades, los generadores de políticas gubernamentales y los *ángeles inversores* para financiar el desarrollo de productos comercializables. Así mismo, se comprobó que es posible que empresas de base tecnológica colombianas utilicen tecnologías modernas para competir con productos provenientes del exterior. Desde el punto de vista de la transferencia tecnológica se puede afirmar que en este caso, se realizó una transferencia exitosa hacia la empresa, ya que los conocimientos fueron absorbidos, asimilados y se generaron nuevos productos que fueron utilizados para dar solución a problemas locales.

1.2.6. Difusión y Desarrollo

La tarea más importante de esta etapa, es la creación de una comunidad que sea consciente de la importancia de la tecnología y utilice los conocimientos generados en el proceso (recurso, bien común), proporcione nuevo conocimiento que haga parte de este recurso; depure las herramientas de difusión y el contenido del mismo; y ayude a crear conciencia en la sociedad de la importancia de esta tecnología en el desarrollo tecnológico del país. Para esto, se deben proporcionar mecanismos que permitan la realización de esas tareas así como estrategias para la vinculación de nuevos miembros. En el capítulo ?? se muestra como está conformada esta comunidad, sus reglas y recursos.

Herramientas para el manejo, almacenamiento, depuración y aumento del recurso

El conocimiento absorbido y generado durante este estudio se encuentra representado en una metodología de diseño e implementación de sistemas digitales que utiliza herramientas abiertas y en una metodología para la transferencia tecnológica y de conocimiento informal basada en la libertad del conocimiento y el uso de este como un bien público. Las plataformas hardware desarrolladas en este estudio y que se encuentran disponibles bajo el esquema de licencias Creative Commons CC-BY-SA representan la aplicación del conocimiento acumulado en el diseño y fabricación de

sistemas embebidos y pueden ser utilizadas por cualquier interesado para mejorar sus habilidades en el diseño, implementación, fabricación o programación de este tipo de dispositivos. El recurso inicial esta compuesto por la información necesaria para entender, programar, reproducir, y modificar estas plataformas; este será la base de nuevos productos tecnológicos que den solución a problemas del país y ayuden a difundir el uso de esta tecnología en centros de formación y empresas de base tecnológica y de esta forma aumentar la oferta de bienes y servicios relacionados con el diseño de aplicaciones personalizadas que se ajusten exactamente a los requerimientos de la sociedad. Sin embargo, para que esto sea posible, es necesario proporcionar herramientas que permitan la administración y difusión del recurso; por este motivo, se creó el portal abierto *linuxencaja* donde se proporcionan los siguientes servicios:

- Sistema distribuido de control de versiones ⁸: El intercambio de información entre programadores usuarios que se encuentran separados geográficamente puede generar una sobrecarga de mensajes que dificultan el proceso de desarrollo; si no se cuenta con una infraestructura adecuada, se corre el riesgo de perder valiosa información por falta de sincronización entre los desarrolladores. De aquí la importancia de contar con un sistema que permita seguir el avance de un proyecto y los cambios que han sido realizados por otros miembros del equipo, los cuales, pueden estar en cualquier lugar del mundo. Para facilitar esta tarea, se cuenta con los sistemas distribuidos de control de versiones abiertos; siendo los más populares *subversion* (svn) y *git*, en este estudio se utilizó *git* ya que permite el seguimiento total del proyecto indicando el autor y un comentario descriptivo con el aporte. En la figura 1.23 se muestran las funcionalidades de esta aplicación.

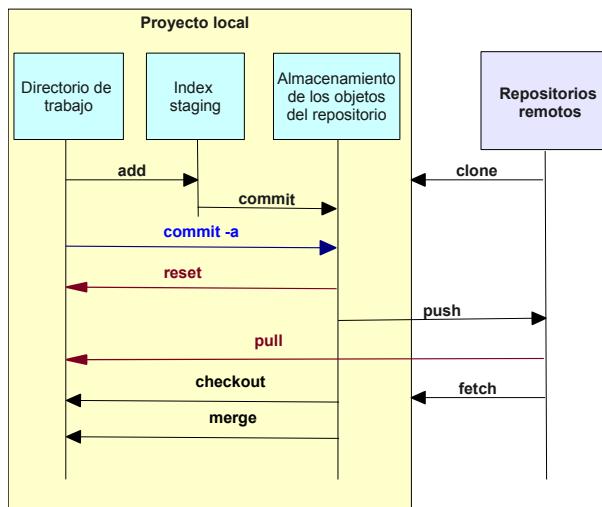


Figura 1.23: Funcionalidad del sistema distribuido de control de versiones **git**

El *sitio de trabajo* es donde se realiza el proyecto y contiene archivos que serán o no tenidos en cuenta para el control de revisiones. El índice (*staging area*, se utiliza para preparar los cambios que serán publicados (*commits*) se pueden agregar (*add*) archivos o partes de archivos para al siguiente *commit*. Todos los cambios que se deseen publicar van primero al índice; la

⁸<http://projects.linuxencaja.net/>

acción *commit* aplica los cambios en el índice en el sitio de almacenamiento de los objetos. Este repositorio de objetos puede ser creado localmente y puede ser copiado a (*push*) o desde (*pull*) otro repositorio remoto. De esta forma es posible sincronizarse y realizar modificaciones sobre un repositorio remoto. Si por algún motivo, accidentalmente se borra o se daña el contenido de algunos archivos del repositorio, es posible restaurar el estado del mismo a un punto donde se reviertan los cambios que ocasionaron los daños (*reset*). Si se piensa introducir un cambio que requiera la modificación de un extenso número de objetos es posible crear y trabajar con una nueva rama del proyecto (*checkout* y conservar la rama original (*merge*)). En la actualidad *linuxencaja* cuenta con 5 repositorios: SIE, ECB_AT91, ECBOT, ECB_BF532 y AndroidStamp; cada uno de ellos reúne la información necesaria para entender y reproducir plataformas hardware que utilizan diferentes SoC. En ellos se pueden encontrar las siguientes aplicaciones:

- Archivos de diseño: Archivos necesarios para reproducir y modificar el PCB.
- Hello World: Esta aplicación implementa un parpadeo (blink) en un LED conectado a un pin de entrada/salida de propósito general, con este sencillo ejemplo se indica como se puede programar la memoria RAM interna del SoC, se proporciona un ejemplo funcional para el desarrollo de aplicaciones sin sistema operativo.
- Loader: Aplicación que permite escribir en las memorias no volátiles de la plataforma.
- Kernel: Parches que dan soporte a la plataforma en el kernel de Linux.
- File system: Sistema de archivos con las utilidades necesarias para la ejecución de aplicaciones.
- Drivers: Módulos del kernel que muestran como implementar drivers de Linux para control de periféricos

blink: La misma funcionalidad que *Hello World* pero como un driver de Linux.

irq: Ejemplo de un módulo del kernel con interrupciones y con un programa en espacio de usuario que controla este módulo.

- Examples: Proyectos académicos destacados que utilizan la plataforma y que pueden ser utilizados como referencia para desarrollo de aplicaciones.

- Wiki ⁹: Las wikis son sitios web con facilidades de edición de páginas web utilizando un lenguaje propio (wikitexto). En este estudio se utilizó el software *Mediawiki* que permite la creación instantánea de páginas web sin la necesidad de permisos especiales por parte del administrador de la máquina; Mediawiki (y demás aplicaciones similares) almacena en una base de datos el contenido de las páginas en texto plano, (este formato se suministra al usuario en el momento de la edición) y genera en tiempo real el código HyperText Markup Language (HTML) a partir de este texto plano cada vez que se solicita una página; una característica especial es que las páginas pueden ser editadas por múltiples voluntarios; adicionalmente, se dispone de un historial de cambios que le permite al administrador restaurar el contenido en un eventual daño voluntario o involuntario.

Con esta herramienta es posible que usuarios del recurso documenten nuevos proyectos o mejoren la documentación de los ya existentes; en la actualidad, esta wiki está siendo utilizada por los estudiantes del área de electrónica digital del Departamento de Ingeniería Eléctrica y

⁹<http://wiki.linuxencaja.net/>

Electrónica de la Universidad Nacional de Colombia para documentar el proceso de diseño de los proyectos de estas asignaturas; los mejores proyectos son conservados y etiquetados como proyectos de referencia y se encuentran disponibles por futuros usuarios que pueden ser industriales o estudiantes de próximos semestres; generando de esta forma un banco de proyectos que puede ser utilizado como fuente de información para mejorar habilidades o para ayuda en el proceso de diseño de nuevos proyectos/productos.

- Lista de correo¹⁰: Contar con un canal de comunicación directo entre los diseñadores y usuarios con más experiencia, permite resolver dudas en corto tiempo, estas respuestas constituyen una fuente de documentación muy importante y debe ser almacenadas para crear un banco de preguntas/respuestas que ayuden a usuarios con las mismas inquietudes. Por esta razón, se creó una cuenta de correo que permite este intercambio de información y puede ser utilizado por los miembros de un determinado proyecto para discutir temas propios del desarrollo del mismo. La metodología utilizada por un determinado grupo quedará registrada en esta lista y puede ser utilizada como referencia de ese y otros proyectos; adicionalmente, estas listas de correo permiten realizar contactos académicos y laborales entre sus miembros.

Estrategia de difusión

Esta metodología de transferencia tecnológica y de conocimientos no cuenta con el apoyo financiero necesario para realizar cursos de capacitación a empresas de base tecnológica destinados a transferir las metodologías de diseño y fabricación propuestas, por este motivo se utilizará una difusión basada en relaciones locales entre centros de formación, empresas, y entidades del estado que financien proyectos de transferencia tecnológica.

- Relaciones con grupos de investigación de la misma universidad: En la Universidad Nacional (sede Bogotá) se importaban todas las placas electrónicas necesarias para el desarrollo de proyectos de investigación en diferentes departamentos de la facultad de ingeniería, muchas de ellas con grandes sobre-costos originados por la importación. Existen áreas de trabajo que pueden ser estudiadas por grupos interdisciplinarios como robótica, telemedicina, telemetría o control. Bajo esta investigación se desarrollaron aplicaciones con los departamentos de sistemas y mecatrónica para desarrollar plataformas robóticas [20] [20] que pudieran ser utilizadas para validar los modelos computacionales y algoritmos propuestos y con el centro de telemedicina de la Universidad Nacional, con quien se diseñó y construyó un monitor remoto de signos vitales [27].
- Relaciones con centros de formación: Aprovechando la posición y reconocimiento de la Universidad Nacional de Colombia y los contactos existentes con los centros de formación que participaron en este estudio. Se difundieron los conocimientos obtenidos y generados en este estudio y se proporcionó la información necesaria para que sean incorporados en los programas académicos de las asignaturas relacionadas con el diseño digital en estos centros de formación. Con esto se busca aumentar el número de personas que utilicen y contribuyan al crecimiento del recurso público. Adicionalmente, el programa académico (ver capítulo ch:education) creado e implementado en la Universidad Nacional de Colombia se encuentra disponible para cualquier centro de educación que quiera implementarlo parcial o totalmente.

¹⁰<http://groups.google.com/group/linuxencaja>

- Relaciones con la industria: Para trabajar con empresas de base tecnológica se creó un proyecto¹¹ que busca difundir el uso de esta tecnología en la creación de nuevos productos que den solución a un problema local; este proyecto realizará un proceso para seleccionar 2 empresas de base tecnológica en ciudades donde existan centros de formación que utilicen este recurso; dichas empresas formularán un proyecto que satisfaga una necesidad local; estudiantes de los centros de formación (como trabajo de grado) ayudarán a estas empresas en el diseño y construcción de un dispositivo que implemente la solución propuesta por ellas. Todos los proyectos deben ser de tal naturaleza que puedan ser implementados en un dispositivo electrónico que utilice la tecnología utilizada en este estudio; esto con el fin de implementar todos los proyectos con una plataforma electrónica base que permita adaptarse a las diferentes aplicaciones; esta *plataforma base* será diseñada por todos los participantes y se buscará reunir la máxima funcionalidad con el menor precio posible; utilizando los recursos suministrados por el portal *linuxencaja* se coordinará el trabajo entre los diferentes grupos de trabajo; una vez diseñada esta plataforma base se fabricaran tantos prototipos como grupos de trabajo para determinar su correcto funcionamiento, una vez corregidos los posibles errores y comprobado el correcto funcionamiento de todos los componentes; se realizará la producción de 100 unidades con el fin de adquirir experiencia en la producción de estos sistemas utilizando la industria manufacturera asiática; después de realizar la pruebas necesarias, cada grupo de trabajo diseñará y fabricará una tarjeta hija que proporcione la funcionalidad requerida por cada proyecto. Todo el proceso de diseño, los criterios de desarrollo, las discusiones para determinar las especificaciones tanto de la plataforma base como de las plataformas hijas se documentará en el portal *linuxencaja* para permitir la reproducción de este curso.

¹¹En la actualidad se están buscando fuentes de financiación para su ejecución

Tabla 1.4: Comparación de precios entre la transferencia ofrecida por Corea y el costo utilizando los conocimientos generados en este trabajo

| Item | KETI-ETRI | Este trabajo |
|---|-----------|--|
| Equipos de cómputo (25) con licencias de: <ul style="list-style-type: none"> ■ Windows, VMWARE,¹² ■ Android SDK, Apache | 115.000 | 50.000 ¹² |
| Plataforma de desarrollo (15) <ul style="list-style-type: none"> ■ Procesador ARM, 512MB DDR, MicroSD ■ LCD 800 x 480, touch screen ■ USB 2.0, WiFi, Camera. ■ Bluetooth. Android OS. | 60.000 | 4.500 ¹³ |
| Software para simulación: <ul style="list-style-type: none"> ■ Qemu con soporte para OpenMoko | 132.000 | 0 ¹⁴ |
| Depuración hardware: <ul style="list-style-type: none"> ■ Trace 32 de Laterbach (15) | 180.000 | 2.000 ¹⁵ |
| Herramientas CAD: <ul style="list-style-type: none"> ■ PADS de Cadence (15) | 180.000 | 0 ¹⁶ |
| Plataforma hardware (15): <ul style="list-style-type: none"> ■ Especificaciones de la plataforma de desarrollo ■ Archivos para reproducir el hardware: <ul style="list-style-type: none"> ● Esquemáticos. ● Archivos Gerber. ■ Código fuente: <ul style="list-style-type: none"> ● bootloader: u-boot. ● Kernel de Linux. ● Android OS. ■ documentación <ul style="list-style-type: none"> ● Guía de usuario. ● Port de Android. | 120.000 | 50.000 <ul style="list-style-type: none"> ■ Archivos para Modificar la plataforma¹⁷ ■ Código fuente: <ul style="list-style-type: none"> ● u-boot ● Kernel de Linux. ● Android OS. ■ Documentación: <ul style="list-style-type: none"> ● Tutoriales. ● Proceso completo de diseño. ● Cursos en Línea. ● Listas de discusión. |

En la actualidad se está evaluando la posibilidad de realizar una transferencia tecnológica entre el estado Colombiano y dos centros de desarrollo coreanos Electronics and Telecommunications Research Institute (ETRI) y Korea Electronics Technology Institute (KETI) en el área de aplicación de televisión digital. Como parte de la transferencia, se contempla transferir conocimientos necesarios para realizar la fabricación de una plataforma de desarrollo que pueda ser utilizada en este tipo de aplicaciones. La tabla 1.4 muestra los costos de esta transferencia (en miles de dólares) y el ahorro

que se puede realizar gracias a los resultados de este trabajo.

Como podemos observar en la tabla 1.4, los costos de esta parte de la transferencia¹⁹ son muy elevados (787.000 USD), frente al costo que tendría realizar las mismas actividades con los conocimientos adquiridos en este trabajo (106.500).

La transferencia que ofrece Corea no tiene en cuenta la capacitación necesaria para que el país pueda absorber y asimilar estos conocimientos de tal forma que pueda generar sus propios productos lo que constituye el éxito de la transferencia. Si este proyecto se llegara a formalizar se crearía una dependencia del país hacia a tecnología ofrecida por Corea. El uso de herramientas comerciales en la etapa de diseño hace que los costos de este tipo de productos aumente de forma considerable, en esta comparación las herramientas de diseño de circuitos impresos, el sistema operativo y un emulador de Linux cuestan 245.000 USD, y son reemplazadas sin costo alguno por las herramientas utilizadas en este trabajo.

¹²Se eliminan las licencias de Windows y VMWARE al utilizar Linux, Android SDK y Apache son gratuitos

¹³Existen dispositivos comerciales con estas características que pueden ser utilizados en el desarrollo utilizando ingeniería inversa, como se hizo en este trabajo

¹⁴Qemu es gratuito, el proyecto del celular abierto OpenMoko distribuye de forma gratuita su simulador

¹⁵El proyecto OpenOCD permite realizar este tipo de pruebas utilizando una interfaz con el puerto JTAG de los procesadores; en este trabajo se desarrolló una interfaz USB-JTAG que permite hacer esto de forma económica.

¹⁶Este trabajo proporciona una metodología para utilizar herramientas abiertas en todo el proceso de fabricación de PCBs

¹⁷Plataforma *hardware copyleft*

¹⁹El costo total de la transferencia cuesta 2 millones de USD

Bibliografía

- [1] Luis Alejandro Cortés. *Verification and Scheduling Techniques for Real-Time Embedded Systems*. PhD thesis, Linköpings universitet Institute of Technology, 2005.
- [2] Colciencias. Plan Estratégico del Programa Nacional de Desarrollo Tecnológico Industrial y Calidad 2005 - 2015. Technical report, Colciencias, 2005.
- [3] M. Smith. *Application specific integrated circuits*. Addison-Wesley, 199.
- [4] M. Odedra. *Information Technology Transfer to Developing Countries: Case studies from Kenya, Zambia and Zimbabwe*. PhD thesis, London School of Economics, 1990.
- [5] Innovation Associates Inc. Technology Transfer and Commercialization Partnerships Executive Summary.
- [6] M. Duque and A. Gauthier. Formación de Ingenieros para la Innovación y el Desarrollo Tecnológico en Colombia. *Revista de la Facultad de Minas - Universidad Nacional de Colombia*, December 1999.
- [7] D Zuluaga, S Campos, M Tovar, R Rodríguez, J Sánchez, A Aguilera, L Landínez, and J Medina. Informe de Vigilancia Tecnológica: Aplicaciones de la Electrónica en el Sector Agrícola. Technical report, COLCIENCIAS, 2007.
- [8] M. Tovar and R. Rodríguez. PROSPECTIVA Y VIGILANCIA TECNOLÓGICA DE LA ELECTRÓNICA EN COLOMBIA. Master's thesis, Universidad Nacional de Colombia, 2007.
- [9] Héctor Martínez. Apropiación de conocimiento en Colombia. El caso de los contratos de importación de tecnología. *Revista Cuadernos de Economía*, 2004.
- [10] Jon Hall. POR GRANDES QUE SEAN...: ASEGURE EL FUTURO DE SU NEGOCIO. *Linux magazine*, ISSN 1576-4079(58):92, 2009.
- [11] A. Grove. How America Can Create Jobs. http://www.businessweek.com/magazine/content/10_28/b4186048358596 May 2010.
- [12] Linux Foundation. Estimating the Total Development Cost of a Linux Distribution. URL: <http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html>, 2008.
- [13] R. M. Stallman. *The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.
- [14] S. Banguero and M. Erazo. *Plataforma de Desarrollo para Sistemas Embebidos Basada en el GameBoy Advance*. Universidad Industrial de Santander, 2007.
- [15] C. Camargo. ECB_AT91 y ECBOT Plataformas Abiertas para el desarrollo de Sistemas Embebidos. URL: <http://wiki.emqbit.com/free-ecb-at91>.

- [16] C. Camargo. First Colombian Linux SBC runs Debian. URL: <http://www.linuxfordevices.com/c/a/News/First-Colombian-Linux-SBC-runs-Debian/>, 2006.
- [17] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *VII conferencia Iberoamericana en Sistemas, Cibernetica e Informatica*, 2008.
- [18] C. Camargo. ECBOT y ECB_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones, Madrid España*, September 2008.
- [19] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *IEEE Colombian Workshop on Circuits and Systems*, 2007.
- [20] C. Camargo. PLATAFORMAS ABIERTAS HARDWARE/SOFTWARE PARA APLICACIONES EN ROBOTICA. *V Congreso Internacional de Ingeniería Mecánica y III de Ingeniería Mecatrónica (por publicar)*, 2011.
- [21] C. Camargo. SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales. *XVII Workshop de Iberchip, Bogotá Colombia*, February 2011.
- [22] D. Ovalle and D. Mendez. *Diseño e Implementación de equipo Básico de Laboratorio Utilizando Sistemas Embebidos y FPAAs*. Universidad Nacional de Colombia, 2004.
- [23] O. Orjuela. Sistema Embebido Para la Reproducción de Video en pantalla dde LEDs RGB. Master's thesis, Universidad Nacional de Colombia. Facultad de ingeniería, 2011.
- [24] C. Gavilan. Diseño Hardware-Software Para Seguimiento de un objeto Mediante Visión Artificial. Master's thesis, Universidad Nacional de Colombia. Facultad de Ingeniería, 2010.
- [25] J. Duque. Voz IP Sobre un Sistema Embebido Linux. Master's thesis, Universidad Nacional de Colombia. Facultad de Ingeniería, 2011.
- [26] D. Arango. Diseño e Implementación de una Plataforma de Desarrollo Para Aplicaciones que Requieran Co-Diseño Hardware-Software Utilizando un Procesador de 32 bits. Master's thesis, Universidad Nacional de Colombia. Facultad de Ingeniería, 2009.
- [27] N. Rosas. Diseño e Implementación de un Sistema Embebido para la Adquisición y Transmisión de Señales Biomédicas a Través de la Red Celular. Master's thesis, 2011.
- [28] D. Mendez. Desarrollo de una plataforma flexible para el prototipaje de aplicaciones WSN (Wireless Network Sensors). Master's thesis, Universidad de los Andes, 2007.
- [29] C. Camargo. Implementación de Sistemas Digitales Complejos Utilizando Sistemas Embebidos. *Memorias del XI Workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [30] C. Camargo. Control de Sistemas Paralelos Inspirado en la Naturaleza. *Colombian Workshop on Circuits and Systems*, 2009.
- [31] C. Camargo. Linux como herramienta de Desarrollo de Sistemas Embebidos. *XII Workshop de Iberchip, San Jose*, 2006.

- [32] C. Camargo and O. Sanchez. Linux embebido como herramienta para realizar reconfiguración parcial. *XII Workshop Iberchip*, 2006.
- [33] I. Castillo, C. Camargo, and C. Perez. Automatización de un puente grúa a escala, mediante una plataforma embebida la cual soporta multiprogramación. *XII Workshop Iberchip*, 2006.
- [34] J. Espinosa, F. Segura, and C. Camargo. Evolución de un Arreglo de Células Utilizando Algoritmos Genéticos. *Memorias del XI Workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [35] C. Camargo. ECBOT y ECB_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-diseño HW-SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones*, 2008.
- [36] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Simposio Argentino de Sistemas Embebidos*, 2011.
- [37] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Congreso Argentino de Sistemas Embebidos CASE 2011, Buenos Aires Argentina*, March 2011.
- [38] C. Camargo. Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft. *XVII Workshop de Iberchip, Bogotá Colombia*, February 2011.
- [39] C. Camargo. Low Cost Platform for Evolvable-Based Boolean Synthesis. *IEEE Latin American Symposium on Circuits and Systems (LASCAS 2011), Bogotá Colombia*, February 2011.