METODOLOGÍA PARA LA TRANSFERENCIA TECNOLÓGICA Y DE CONOCIMIENTOS EN EL DISEÑO DE SISTEMAS EMBEBIDOS

UNIVERSIDAD NACIONAL DE COLOMBIA

Carlos Iván Camargo Bareño

9 de marzo de 2011

METODOLOGÍA PARA LA TRANSFERENCIA TECNOLÓGICA Y DE CONOCIMIENTOS EN EL DISEÑO DE SISTEMAS EMBEBIDOS

AUTHOR: C. Camargo
E-MAIL: cicamargoba@unal.edu.co

Published by the Universidad Nacional de Colombia

Índice general

Capítulo 1

Sistemas Embebidos

1.1. Introducción

1.1.1. Qué es un sistema Embebido

Un 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, un Sistema Embebido realiza tareas pre-definidas, lo cual permite su optimización, reduciendo el tamaño y costo del producto [1]

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 su funcionalidad (por ejemplo, el control de un asensor siempre realizará las mismas acciones durante su vida útil); debido a su interacción con el entorno deben cumplir esctríctamente restricciones temporales, el término *sistemas de tiempo real* es utilizado para enfatizar este aspecto; son heterogéneos, es decir, están compuestos por componentes hardware (PLDs, ASICs) y software (μ-controladores, μ-procesadores, DSPs); tienen grandes requerimientos en términos de confiabilidad, errores en aplicaciones para aviación o automovilismo, pueden tener consecuencias desastrosas.

1.1.2. Arquitectura

En la Figura 1.1 se muestra la arquitectura típica de un Sistema Embebido. La cual integra un componente hardware, implementado ya sea en un (CPLD, FPGA) o en un ASIC, (conocido con el nombre de periféricos) y un componente software capáz de ejecutar software, la parte del procesador está dividida en la CPU (En algunos casos posee una caché) y las unidades de Memoria.

Al momento de diseñar un Sistema Embebido encontramos 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:

- Componente HW y SW Integrado en un dispositivo semiconductor (SoC, ASIC): En la actualidad existen muchas compañías que fabrican procesadores de 32 bits integrados a una gran variedad de periféricos, lo cual simplifica el diseño y reduce costos; este tipo de implementación es muy popular

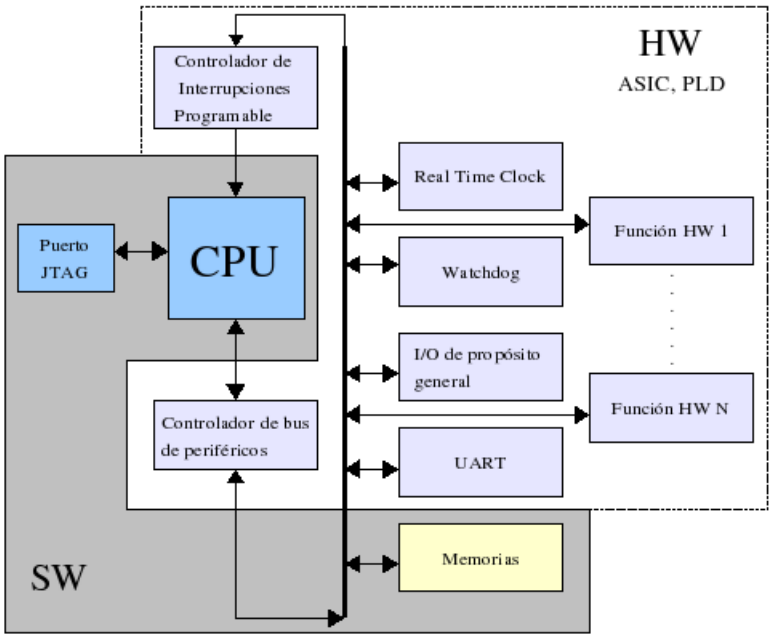

Figura 1.1: Arquitectura de un Sistema Embebido

en los dispositivos de consumo masivo (Reproductores de MP3, consolas de juego, 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, esto disminuye el costo de componentes y reduce el área de circuito impreso.

- Componente SW en un SoC y componente HW en una 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 ocaciones el periférico puede relizar funciones poco comúnes y no se proporciona comercialmente, la solución es entonces, implementar estas funcionalidades en una FPGA. También se recomienda la utilización de FPGAs en sistemas que requieren la utilización de la misma funcionalidad un gran número de veces (puertos seriales, pines de entrada/salida). Esta decisión esta atada al nivel de producción, ya que al incluir una FPGA aumenta el costo global del proyecto y al consumo de potencia, el consumo de las las FPGAs actuales las hace poca prácticas para aplicaciones móviles.

- Componente SW y HW en una FPGA: Esta es tal vez 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.

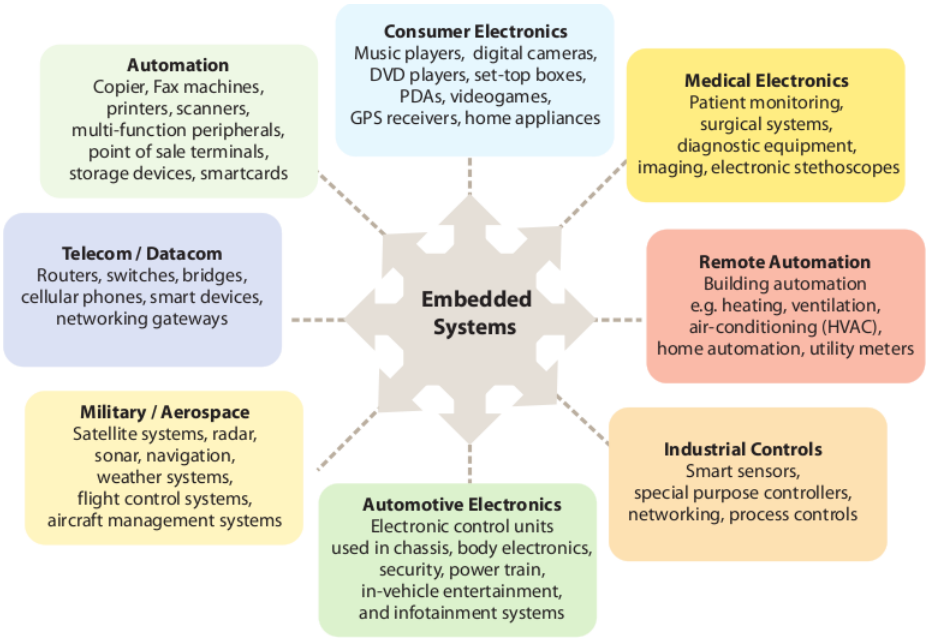

Figura 1.2: Aplicaciones de los Sistema Embebidos Fuente: TATA Consultancy services

1.1.3. Aplicaciones

Los sistemas embebidos se encuentran en casi todas las actividades humanas, a diario interactuamos con ellos aún sin darnos cuenta, ya sea porque son parte de nuestra vida diaria o porque hacen parte de aparatos que usamos a diario. La figura

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. Esta especificación debe ser muy general y no deben existir dependencias (tecnológicas, metodológicas) de ningún tipo, se suele utilizar lenguajes de alto nivel, como UML, 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 algorítmos seleccionados, por ejemplo, determinar si el algoritmo siempre termina o 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 algorítmos 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 existoso de la especificación a la implementación. Es importante definir que modelo matemático debe soportar el entorno de diseño; los modelos más utilizados son: Máquinas de estados algorítmicas, diagramas de flujos de datos, sistemas de eventos discretos y redes de petri; 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.

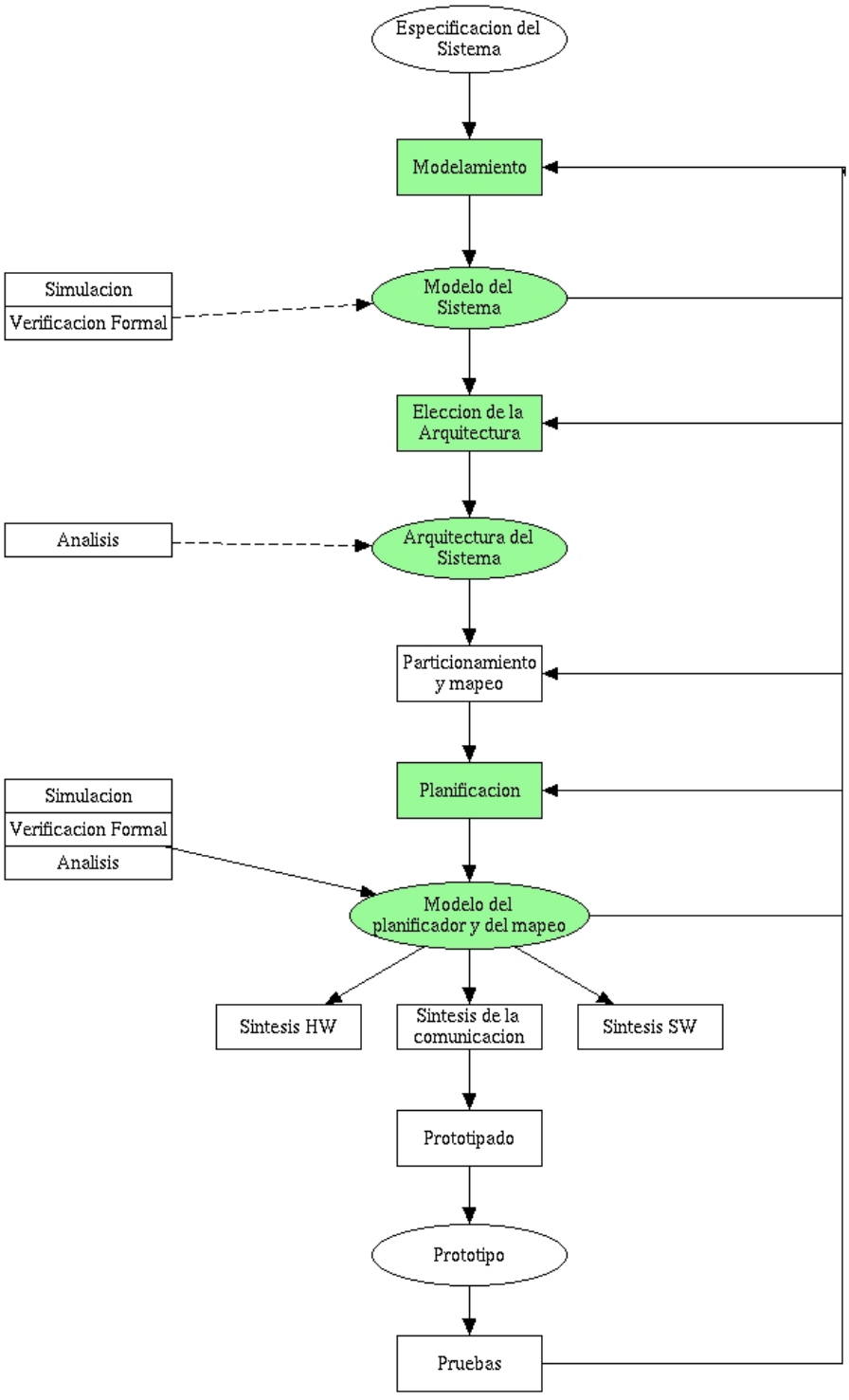

Figura 1.3: Flujo de Diseño de un Sistema Embebido [2]

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 más 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, viabilidad comercial.

Utilizando como base la arquitectura obtenida en el paso anterior las tareas del modelo del sistemas son mapeadas dentro de los componentes; esto es, asignación de funciones a los componentes 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 procesador.
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 fín de no ocupar al procesador en tareas cíclicas, un ejemplo típico de estas tareas es la generación de bases de tiempos. La decisión de que tareas se implementan en SW y que tareas se implementan en HW recibe el nombre de *particionamiento*, esta selección es fuertemente dependiente 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 hacer 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 o FPGA) y se debe obtener el ″ejecutable″ de las tareas software, 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 se cumplen 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. Elección

Niveles de complejidad de la tecnología

Existen varias alternativas para la implementación de un sistema embebido: FPGA, 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 ASIC (Circuito Integrado de Aplicación Específica) que contenga las tareas hardware y software en un circuito integrado; 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 el momento no se conoce el primer circuito integrado diseñado por una empresa colombiana o de existir, no es una práctica común en nuestro medio. Por otro

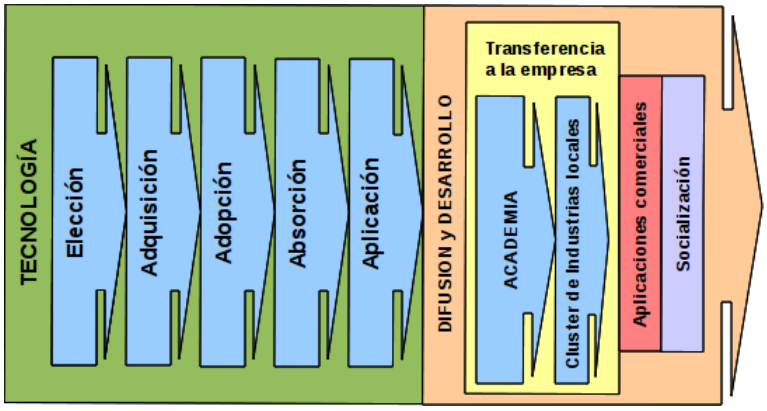

Figura 1.4: Etapas de la metodología propuesta para la transferencia tecnológica y de conocimientos en el área de diseño de sistemas embebidos

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 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 (HDL) como herramienta para la implementación de aplicaciones; es normal encontrar trabajos de pregrado y posgrado que utiizan 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 herramientas 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, 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 y 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 fín 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, ya que el proceso de fabricación de un ASIC toma varios meses; sin embargo, para que un producto sea viable económicamente es necesario 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 *standard cells* o diseños *full custom*; utilizando esta técnica, es posible reducir el número de unidades necesarias para encontrar un punto económicamente

viable de 5000 unidades; tal como se ilustra en la figura 1.5.

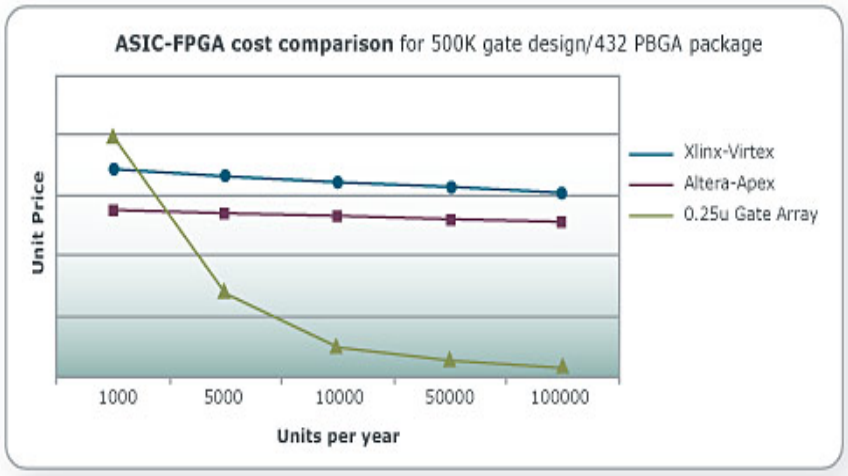

Figura 1.5: Comparación de costos entre FPGAs y Arreglos de Compuertas, Fuente: Silicon-Pro

El la figura 1.6 se muestra un comparativo de la utilización de tres tecnologías, las FPGAs, las celdas estandar y el arreglo de compuertas, en ella podemos observar que la opción más económica para bajos volúmenes de producción son las FPGA, 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 punto de ruptura se produce alrededor de las 50000 unidades, donde es más rentable la producción de un ASIC basado en celdas estándard. 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 apliacaciones móviles.

En Colombia es muy común el uso de micro-controladores y micro-procesadores de 8 bits como herramienta 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 actuales que requieren 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 que corre a frecuencias que van desde los 75MHz hasta los 800 MHz y poseen periféricos que permiten controlar directamente una gran capacidad de dispositivos, muchos de ellos están diseñados para apliacaciones que requien manejo de pantallas táctiles de cristal líquido, conexión a intenet, diferentes medios de almacenamiento, reproducción de audio, manejo de sensores de imágen, 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 diveros

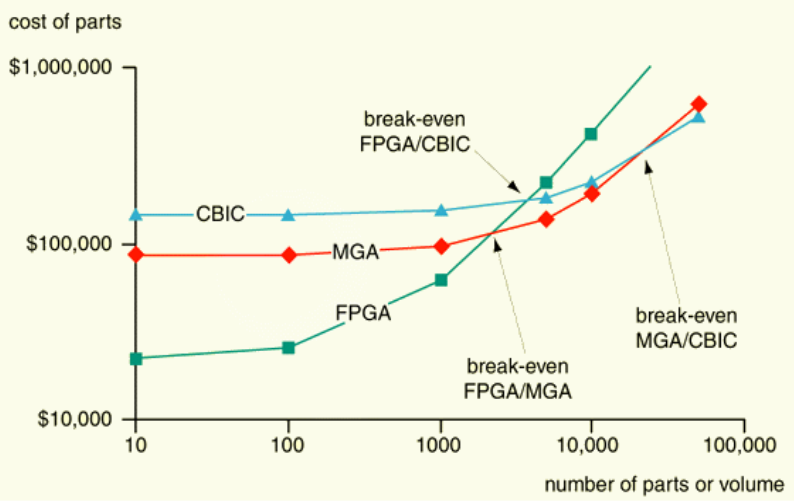

Figura 1.6: Comparación de costos entre FPGAs, Arreglos de Compuertas y ASIC basado en celdas Standard, Fuente: Application-Specific Integrated Circuits, MJS Smith

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 USD.

Diagnóstico de la Industria Local

Para determinar el estado de la industria electrónica en Colombia, se creó la empresa **emQbit LTDA.** en asociación con profesionales en ingeniería de sistemas, ingeniería eléctrica e ingeniería electrónica. Esta empresa 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. que coincide con los resultados de estudios consultados [3] [4] [5] [6] [7] [8].

Diagnóstico de la Academia

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 [9] ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y

operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [10] generando desempleo en las personas menos capacitadas. Según John Hall presidente y CEO de Linux International " algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico" [9]. 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), esto unido a programas académicos centrados en el análisis y no el diseño, donde el paso final es la simulación y el personal docente no tiene ninguna experiencia en 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.

Conclusión

Como vimos 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-controladores, o micro-procesadores 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 apliaciones móviles; por esta razón no es común encontrar FPGAs en dispositivos de consumo. Una revisión de los circuitos integrados utilizados en dispositivos como reproductores MP3, juegos, routers, y algunos teléfonos celulares; 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 lo podemos observar en las diferentes versiones de reproductores MP3 de la compañia *AINOL*, la primera versión evaluada contenía un DSP de Analog Devices, un codec de audio externo y un controlador para el puerto USB; en la siguiente versión evaluada solo se encontramos 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; adicionalmente, su costo era de 3.5 USD en grandes volúmenes. Esto refleja el estado de los SoC actuales, los grandes fabricantes como Atmel, Freescale, Marvell, NXP, Ingenic, etc, de foma 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 SoC 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.

Nuestra hipótesis 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, como se vió anteriormente el país no cuenta con una oferta considerable de bienes y servicios relacionados con la industria manufacturera de dispositivos digitales, es necesario, utilizar inicialmnete los servicios de la industria asiática para construir dispositivos diseñados en el país, y 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 estar preparados para una futura demanda de la industria local.

1.2.2. Adquisición

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; adicionalmente, es crucial contar con una adecuada documentación e información que ayude a resolver problemas que se presenten en el ciclo de diseño. En la actualidad podemos 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 formuadas 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 relizar 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 consultamos varias encuestas realizadas por compañias 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 publicamente, 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 1.7; el porcentaje de utilización de sistemas operativos basados en proyectos abiertos es del 35.6 %, lo que supera a los sistes operativos comerciales; es interesante observar que casi el 70 % de los encuestados utilizan algún tipo de sistema operativo, lo que nos da un claro indicio de la necesidad de este en el ciclo de diseño.

De la anterior podemos obervar 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 celulares realizada por Admob indica que android superó a los sistemas operativos de Apple y a RIM utilizado en los blackberry y se proyecta que en el 2014 igualará a Symbian de Nokia (aunque esto puede ocurrir antes, debido a la mala acogida que ha tenido la unión Nokia - Microsoft). Android utiliza el kernel de Linux como base de su desarrollo y utiliza herramientas abiertas para su desarrollo; otras empresas como Motorola y Nokia utilizan Linux como plataforma de varias de sus aplicaciones; asi 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 recientemente un estudio 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 basa de datos de poryectos abiertos, representados en 200.000 proyectos, 4.9 billones de líneas de código, utilizando su detallado conocimiento de los proyectos abiertos y aplicando técnicas estándard de estimación de costos, estiman que el costo de desarrollo total del proyecto FOSS excede los USD$387000 millones y representa la inversión colectiva de mas de dos

[1] http://www.blackducksoftware.com Líder mundial en el suministro de productos y servicios que aceleran el desarrollo software utilizando software libre

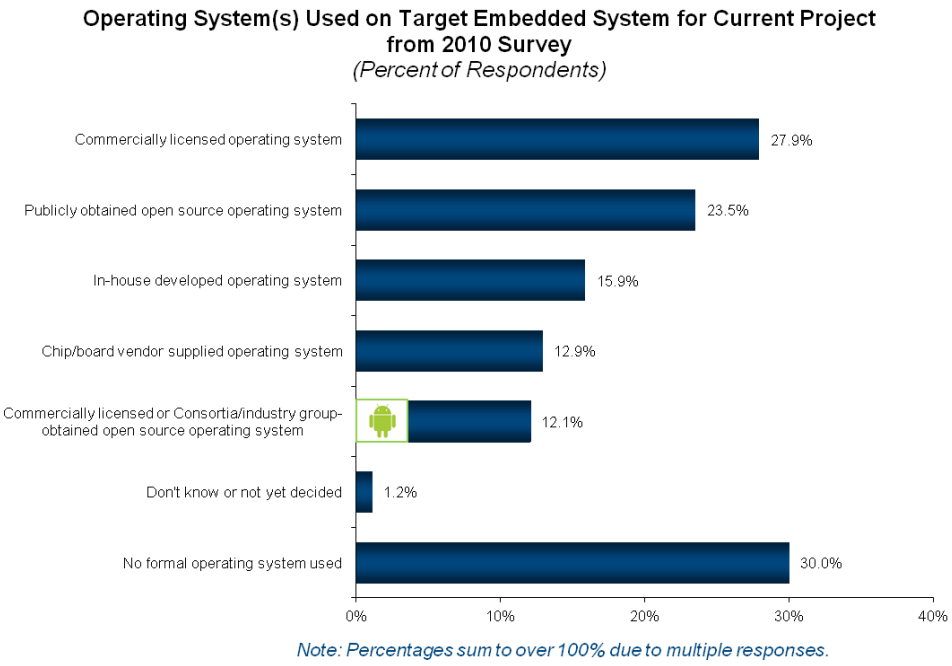

Figura 1.7: Comparación del uso de sistemas operativos Fuente: Venture Development Corp

millones de desarrolladores al año. Un análisis adicional, el cual estima que el 10 % de las aplicaciones utilizadas para el desarrollo de aplicaciones IT, se pueden reemplazar por proyectos abiertos , lo que ahorraría mas de USD$22 billones al año.

Los proyectos de código abiero 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 y aplicaciones; permite invertir tiempo y esfuerzo en proyectos que pueden ser comercializados rápidamente.

Dispositivos Semiconductores

Existe una gran oferta de SoC en la actualidad, grandes compañias 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 (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 IP (Intellectual Property); compañías como Atmel, Marvell, Freescale, NXP, Cirrus Logic, Samsung, Texas Instruments adquieren licencias que les pemiten utilizar estos núcleos lógicos en la fabricación de sus SoCs. La figura 1.18 muestra los resultados de una encuesta realizada por *linuxfordevices* a diseñadores 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 Ingenic) y el DSP de Analog

devices BF532.

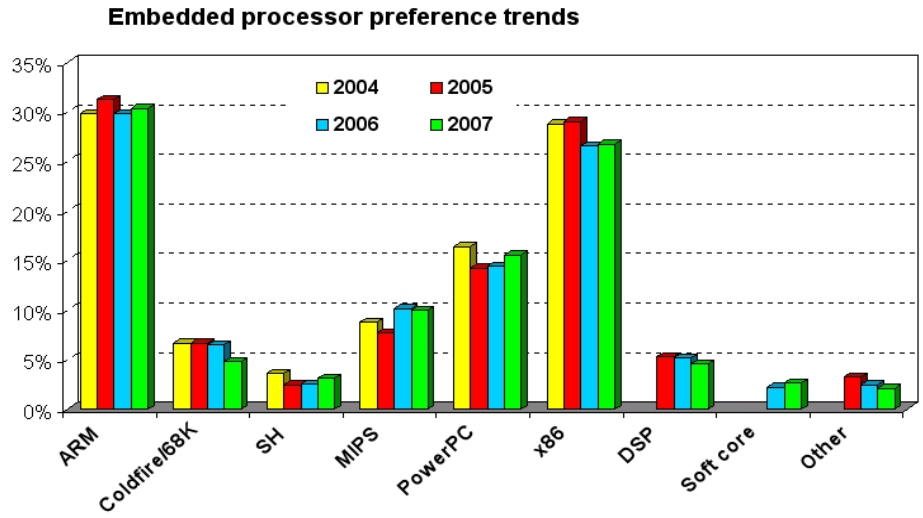

Figura 1.8: Comparación de los procesadores más utilizados para aplicaciones embebidas Fuente: Linux-fordevices

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; dispositivos *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, LEDs, Displays, LCDs, 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 en una determinada aplicación. El término 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 cual reduce los costos y tiempos de desarrollo; desafortunadamente en el país no existe una gran demanda, por lo que deben ser importados, lo que hace que su costo se eleve por lo menos en un 26 % (arancel e IVA). Las plataformas adquiridas para este estudio se muestran en la figura 1.9, la primera de izquierda a derecha 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 segunda es la plataforma de la compañía china *Embest* basada en el SoC AT91R40008 (ARM7), la tercera es la agenda electrónica Zaurus de Toshiba (basado en el StrongARM de Intel SA-1110), la cuarta es el chumby (basado en un ARM de freescale iMX233) y finalmente un portaretratos digital de la empresa china *SUNGALE*

Figura 1.9: Plataformas adquiridas para el estudio de los Sistemas Emebebidos

1.2.3. Adopción

En esta etapa se utilizarán 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; por lo anterior, este tipo de plataformas reduce de forma considerable la inversión necesaria para realizar las actividades de esta etapa.

Metodología

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

Como podemos ver en la figura 1.7 existen dos formas de realizar las aplicaciones de un sistema embebido: con sistema operativo (OS) y sin OS (*standalone*). Las aplicaciones standalone son muy eficientes debido a que son escritas teniendo en cuenta los recursos hardware (memoria, velocidad del procesador, 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, 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 el sistema operativo pueda ejecutarse. 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 gener 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 permte el uso eficiente de los recursos hardware.
    - Linux: Sistema operativo ampliamente utilizado que posee una gran cantidad de aplicaciones y un numero considerable de desarrolladores.
        - Inicialización: Requerimientos mínimos para la ejecución de Linux.
        - Imágen 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 SoC 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 SoC que poseen memoria no volátil (hasta 512 Kbytes) normalmente incorporan una memoria RAM (hasta 32 kbytes) junto con una serie de periféricos (timers, I2C, SPI, USARTs, ADCs, Watchdog, USB device, canales para acceso directo a memoria - DMA); están diseñados para no utilizar componentes externos; normalmente este tipo de dispositivos utilizan procesadores que no tienen unidad de manejo de memoria [2] (MMU) como la famila ARM7, cuyas velocidades de ejecución varían entre los 50 y 70MHz. En la figura 1.10 se muestra la arquitectura típica de un sistema basado en estos dispositivos.

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 (hasta 128 Kbytes) y adicionamente a los periféricos mencionados anteriormente se suministran controladores para USB host, puertos 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 SD; y memorias volátiles SDRAM y DDR; su velocidad de operación varía entre los 75MHz y 800MHz. En la figura 1.11 se muestra a arquitectura típica de un sistema basado en estos procesadores.

[2] La 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, arbitramento de buses

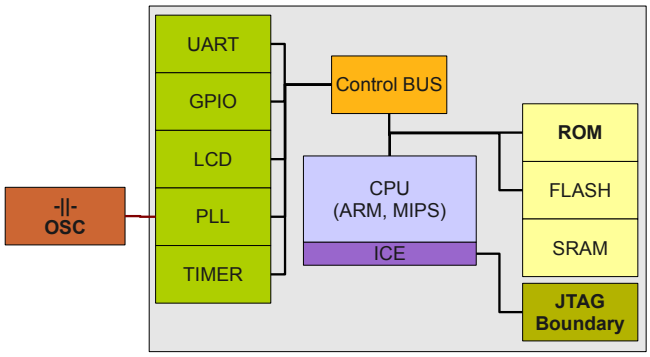

Figura 1.10: Arquitectura típica de un sistema embebido que utiliza SoC con memoria volátil interna

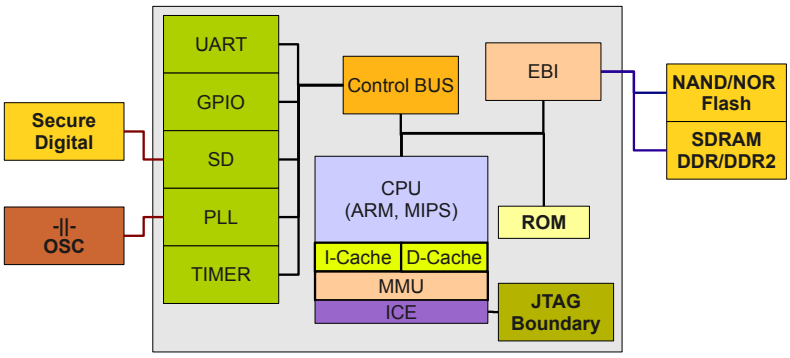

Figura 1.11: Arquitectura típica de un sistema embebido que utiliza SoC sin memoria volátil interna

Debido a la falta de memoria volátil, las aplicaciones de este tipo de dispositivos requieren una memoria externa de este tipo, en la actualidad las más populares son las memorias NAND flash, NOR flash, SPI, EEPROM y SD. Normalmente, este tipo de procesadores son utilizados en aplicaciones que utilizan sistemas operativos, como veremos más adelente, para que ciertos sistemas operativos (Linux, Windows CE) puedan ejecutarse se requiere una mínima cantidad de memoria RAM (del 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, podemos decir que en el mercado existen opciones que nos 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 SoC que incluyan las memorias no volátiles y ram internamente; sin embargo, hasta el momento no existen dispositivos con grandes capacidades memoria Flash y RAM internas, por lo que no es posible su uso para ciertas aplicaciones que utilicen grandes cantidades de memoria; por ejemplo, aplicaciones gráficas con LCDs con resoluciones mayores a la VGA. Utilizar SoC que no integren 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.

Cuadro 1.1: Arquitectura de las plataformas comerciales utilizadas en la etapa de aborción

Plataforma	**CPU**	**Mem. volátil**	**Mem. no volátil**	**MMU**	**OS**
Game Boy	ARM	Externa NOR	Interna	NO	Propietario
Zaurus	ARM	Externa NAND	Externa SDRAM	SI	Linux
iPAQ H3600	ARM	Externa NAND	Externa SDRAM	SI	Windows CE
Chumby	ARM	Externa NAND	Externa SDRAM	SI	Linux
Ainol V2000	MIPS	Externa NAND	Externa SDRAM	SI	Linux
SUNGALE DPF	MIPS	Externa NAND	Externa SDRAM	SI	Linux

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 proporciona controladores de LCDs, mouse, teclado, touch screen, CODECs 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 CAN, I2C, SPI.

Programación

Como se mencionó anteriormente, para este estudio se utilizarán herramientas abiertas para la creación de aplicaciones, en la figura 1.12 se muestra el flujo de creación de las tareas software usando la cadena de herramientas GNU [11]. 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 enlazado).

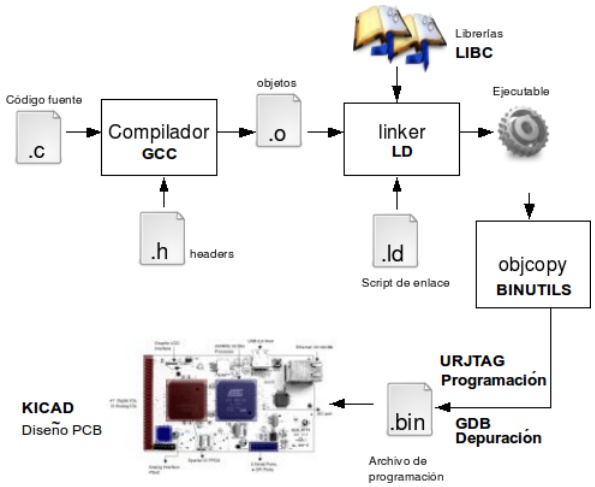

Figura 1.12: Flujo de diseño software para creación de aplicaciones.

El proceso de generación de el 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. Los SoC proporcionan 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 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 SoC 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 que no implique desoldar las memorias.

Existen dos formas de realizar esta programación: utilizando un canal de comunicación suministrado por el SoC; 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 1.13 muestra este proceso.

Debido a que la RAM interna normalmente es pequeña (del orden 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); la segunda aplicación se descarga a la memoria externa debe tener la capacidad de programar las memorias no volátiles externas con un programa que permita futuras actualizaciones, este segundo programa recibe el nombre de *booloader* y es almacenado en las primeras posiciones de la memoria no volátil, de tal forma que es el primero que se ejecuta cuando se activa la señal de reset.

Una vez programada la memoria no volátil con una aplicación válida los SoC realizan una serie de pasos para ejecutar las aplicaciones almacenadas en ella, esto debido a la poca capacidad de la memoria RAM

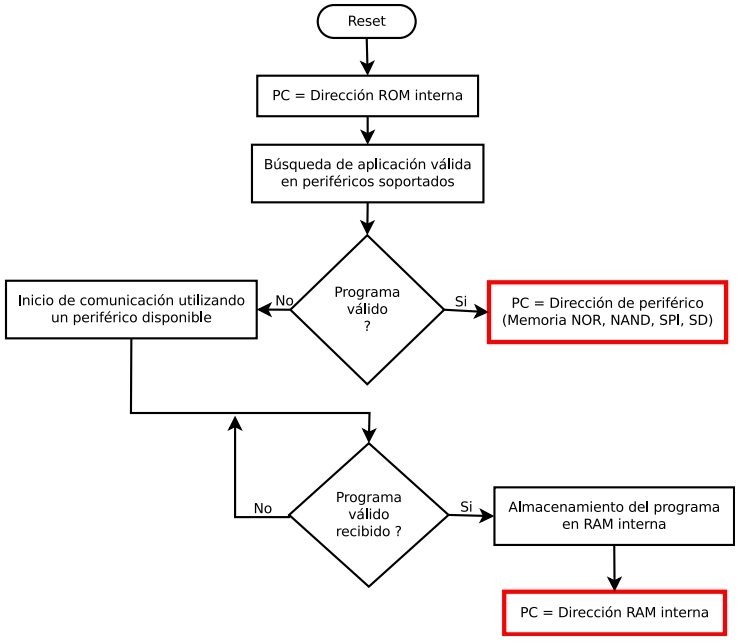

Figura 1.13: Inicialización de un SoC cuando las memorias no volátiles no están programadas.

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 1.14 (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 1.14 (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 (PC) el valor correspondiente a la memoria RAM externa (figura 1.14 (c)).

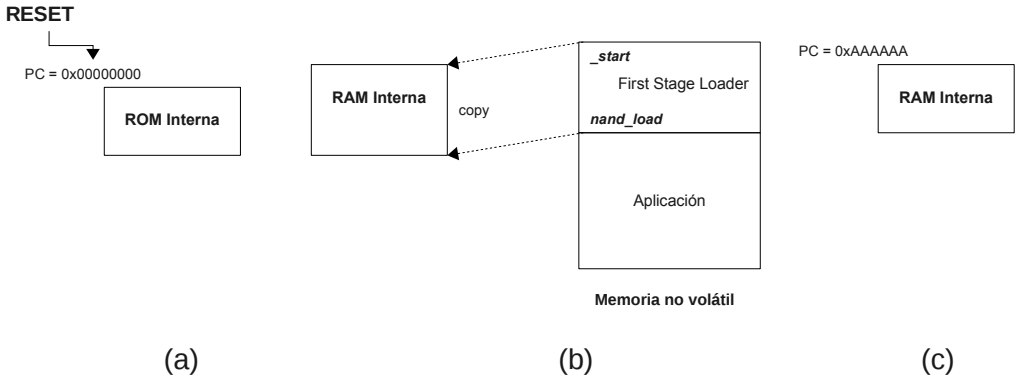

Figura 1.14: Inicialización de un SoC cuando la memoria no volátil está programada, parte 1.

Este programa (*loader*)está encargado de: configurar la memoria RAM externa (su capacidad varía dependi-

endo 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 almaceno esta aplicación (ver figura 1.15)

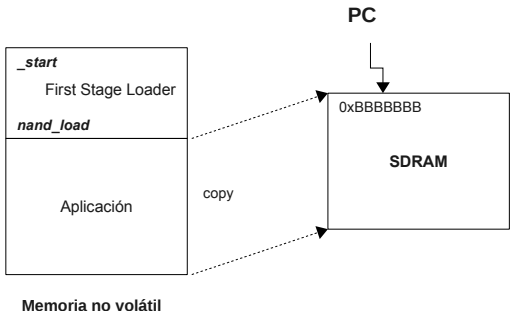

Figura 1.15: Inicialización de un SoC cuando la memoria no volátil está programada, parte 2.

Programación utilizando el puerto JTAG

Algunos SoC no suministran un camino para la programación de la memoria RAM interna, para estos casos, se puede utilizar una utilidad que la mayoría de los dispositivos proporciona: el puerto JTAG; creado inicialmente como un mecanismo para realizar pruebas en las tarjetas d ecircuito impreso para verificar la correcta conexión entre componentes, y verificar el correcto funcionamiento de los circuitos integrados; se basa en un registro de desplazamiento (ver figura 1.16 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 adicionado funcionalidades a este protocolo y una de ellas es el control de circuitos especializados dentro de los SoC para realizar emulación en circuito (ICE), suministrando un canal para la programación de la memoria RAM interna.

Algunos SoC 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 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 muy largo.

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 1.17; 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 32 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

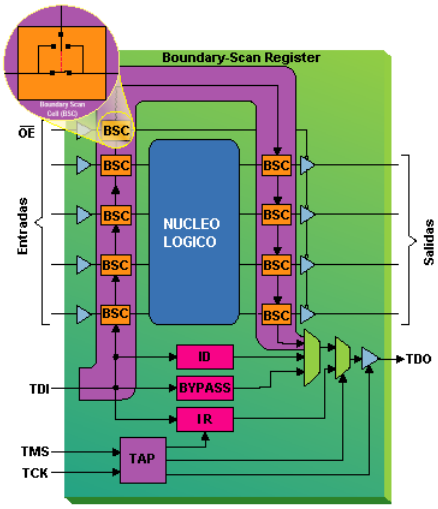

Figura 1.16: Cadena Boundary Scan fuente: Texas Instruments.

en su funcionamiento que recibe el nombre de *latenica*; y se definie 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.

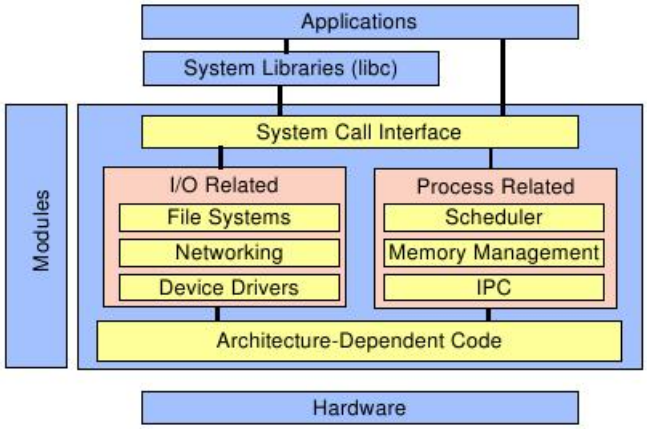

Figura 1.17: Estructura del kernel de Linux.

Por otro lado, las aplicaciones standalone utilizan los recursos necesarios y dependiendo de su complejidad pueden ajustarse a SoC que incorparan memoria RAM y no volátil internas; adicionalmente, su programación puede reducir el problema que se presenta con los sistemas operativos con la latencia a las interrupciones. 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

Cuadro 1.2: Conocimientos Adquiridos durante la etapa de absorción

Plataforma	**CPU**	**Mem. volátil**	**Mem. no volátil**	**MMU**	**OS**
Game Boy	ARM	Externa NOR	Interna	NO	Propietario
Zaurus	ARM	Externa NAND	Externa SDRAM	SI	Linux
iPAQ H3600	ARM	Externa NAND	Externa SDRAM	SI	Windows CE
Chumby	ARM	Externa NAND	Externa SDRAM	SI	Linux
Ainol V2000	MIPS	Externa NAND	Externa SDRAM	SI	Linux
SUNGALE DPF	MIPS	Externa NAND	Externa SDRAM	SI	Linux

Conocimientos adquiridos Metodología de diseño usando software libre Sistema Operativo eCos, Linux Diseño de periféricos en PLDs y drivers

Plataformas Utilizadas y conocimientos adquiridos

Game Boy de Nintendo + Chamedlabs Xport Programación de procesadores de 32 bits. Cadena de Herramientas GNU Sistema operativo eCos. Programación de memorias no volátiles. Interfaz entre procesadores y periféricos implementados en una FPGA. Diseño de periféricos en HDL. Zaurus SL-5500 Desarrollo de aplicaciones utilizando el sistema operativo Linux. Desarrollo de aplicaciones gráficas Requerimientos HW para ejecutar aplicaciones Linux.

Chumby Modificación del kernel de Linux. Creación de módulos del kernel. Desarrollo de aplicaciones gráficas usando QT

Arquitectura de sistemas digitales. Metodología de diseño de sistemas embebidos.

Aplicaciones Realizadas

Game Boy de Nintendo + Chamedlabs Xport Académicas Osciloscopio Digital utilizando FPAAs Automatización de un Puente grúa a escala. Control Adaptativo Embebido Control de un horno de reflujo para componentes de montaje superficial. Dispositivo de visualización de variables que suministra el computador de un automóvil. Comerciales Adquisición de datos para medición de calidad de energía. Plataforma robótica didáctica. Chumby Registro de aceleración de vehiculos para compañías de seguros. Sistema de seguimiento vehicular. Porta Retrato Digital Monitor de signos vitales. Diccionario basado en Wikipedia. Menú electrónico y consola de juegos.

Ventajas de utilizar plataformas comerciales

Facilita el aprendizaje del funcionamiento de la tecnología. Permite estudiar metodologías de diseño y evaluación de herramientas de desarrollo. Reduce el tiempo y el costo de desarrollo. Variedad y disponibilidad.

Desventajas No se puede o es muy costoso modificarlas. No son productos exclusivos y pueden ser duplicados facilmente. Dependencia del fabricante. Reducción de la demanda de diseñadores y proveedores de bienes y servicios de manufactura Diseñadas para una aplicación específica Dificiles de adaptar a nuevas aplicaciones Escasa documentación. No se puede utilizar en productos comerciales. Limita el aprendizaje. Diseñadas como plataformas de desarrollo En ocaciones son sobredimensionadas. Es necesario diseñar

tarjetas adicionales y carcasas para adaptarlas a una aplicación comercial.

1.2.4. Absorción

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 en centros de educación consolidados.

Integración de nuevo conocimiento para el país pero no es nuevo para el mundo. Adaptación de metodologías de diseño y procesos de fabricación al entorno local. Desarrollo de productos tecnológicos propios. Transmisión de conocimientos a la academia

Actividades Realizadas

Transferencia de conocimientos a la academia. Desarrollo de técnicas de fabricación de prototipos. Aplicación de metodologías de diseño en la solución de problemas. Consideraciones de diseño para cumplir normas internacionales. Diseño de aplicaciones comerciales y académicas. Contacto con empresas manufactureras locales y asiáticas para fabricación de PCBs.

Productos Comerciales

Control de tornos industriales. Plataforma robótica didáctica. Monitoreo de Temperatura. Sistema de seguimiento vehicular. Sistema de medición de la calidad del suministro de energía eléctrica. Monitor de signos vitales (UNAL). Sistema de comunicación encriptada utilizando el canal GSM (MICROENSAMBLE). Switch de 4 canales de radio frecuencia (TESAMERICA).

Productos Académicos

Plataformas de desarrollo para: CPLD FPGAs Procesadores ARM Robótica Linux Embebido Codiseño HW/SW y Linux. Programa académico para la enseñanza de sistemas digitales utilizando tecnología de punta que crea las habilidades necesarias para concebir, diseñar, implementar y operar dispositivos digitales modernos. Definición del concepto hardware copyleft y su utilización como herramienta en la enseñanza de diseño de sistemas embebidos.

Aplicación

Desarrollo de soluciones a problemas locales; uso de metodologías de diseño en la concepción, diseño e implementación de sistemas digitales utilizando la tecnología; utilización de procesos de fabricación adaptados al entorno local; desarrollo de proyectos académicos utilizando esta tecnología.

Uso de metodologías de diseño en la concepción, diseño e implementación de sistemas digitales utilizando la tecnología. Utilización de procesos de fabricación adaptados al entorno local. Desarrollo de soluciones a problemas locales. Control de tornos industriales; Plataforma robótica didáctica; Monitoreo de Temperatura; Sistema de seguimiento vehicular; Sistema de medición de la calidad del suministro de energía eléctrica; Monitor de signos vitales; sistema de comunicación encriptada utilizando el canal GSM; switch de 4 canales de radio frecuencia. Desarrollo de proyectos académicos utilizando esta tecnología. Plataformas de desarrollo para: FPGAs, proc. ARM; Linux Embebido; codiseño HW/SW. Programa académico para la enseñanza de sistemas digitales.

Difusión y Desarrollo

Vinculación de la academia para incluir los conocimientos generados en los programas académicos de las carreras relacionadas; capacitación a la industria local sobre el uso de la tecnología, las metodologías de diseño y procesos de producción; creación de una comunidad que utilice, mejore y aumente el conocimiento generado; hacer que el conocimiento generado en los pasos anteriores este disponible a todos los interesados; dar a conocer los procesos, productos y conocimientos creados a los generadores de políticas de estado.

Aumento de la demanda de productos, bienes y servicios relacionados; compra de maquinaria que permita la fabricación masiva de forma local; diseño de nuevos componentes (Circuitos Integrados, IPs, software CAD); creación de políticas gubernamentales que protejan la producción local; participación activa de la academia en la solución de problemas y en la formulación de políticas de gobierno relacionadas.

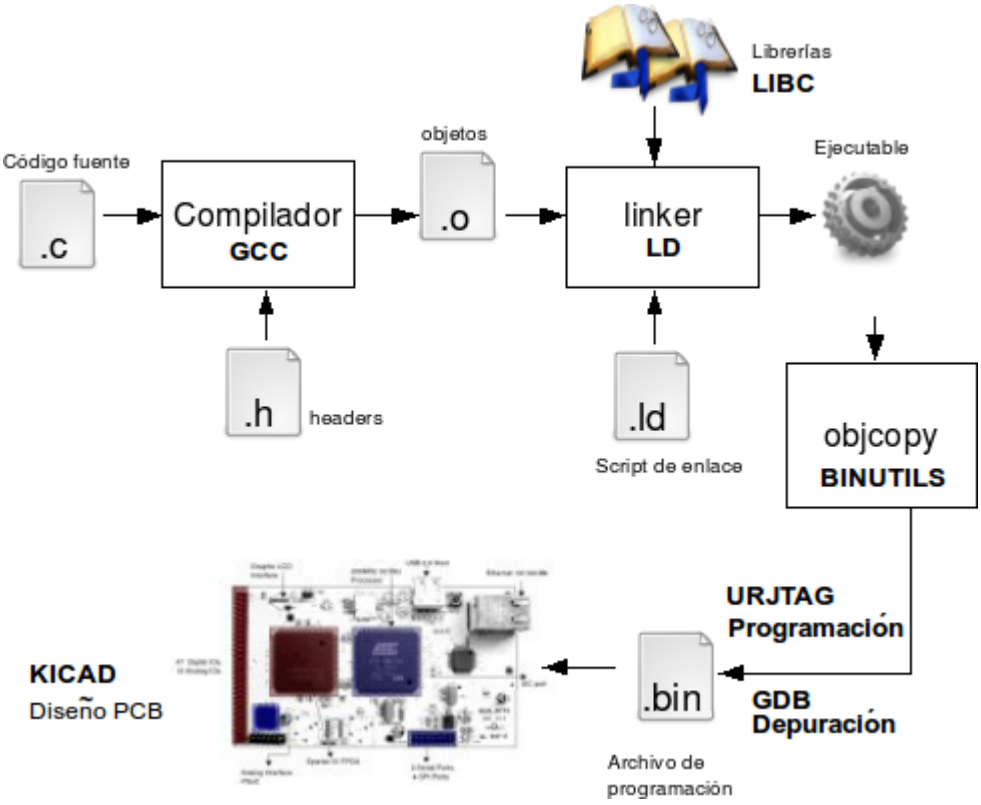

Figura 1.18: Flujo de Diseño SW

Bibliografía

[1] Wikipedia. Wikipedia, the free encyclopedia. http://en.wikipedia.org/.

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

[3] M. Odedra. *Information Technology Transfer to Developing Countries: Case studies from Kenya, Zambia and Zimbabwe*. PhD thesis, London School of Economics, 1990.

[4] Innovation Associates Inc. Technology Transfer and Commercialization Partnerships Executive Summary.

[5] M. Duque and A. Gauthier. Formación de Inegnieros para la Innovación y el Desarrollo Tecnológico en Colombia. *Revista de la Facultad de Minas - Universidad Nacional de Colombia*, December 1999.

[6] 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.

[7] M. Tovar and R. Rodríguez. PROSPECTIVA Y VIGILANCIA TECNOLÓGICA DE LA ELECTRÓNICA EN COLOMBIA. Master's thesis, Universidad Nacional de Colombia, 2007.

[8] Héctor Martínez. Apropiación de conocimiento en Colombia. El caso de los contratos de importación de tecnología. *Revista Cuadernos de Economía*, 2004.

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

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

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