Módulo empotrado del sistema SMART para la gestión de servicios de redes inteligentes a través de ISUP del protocolo SS7

MAHDI SAFA DAOUD

RICARDO ANDRES VALLECILLA SIERRA

ANEXO A

TÉCNICAS DE CODISEÑO ELECTRÓNICO APLICADAS A SISTEMAS DE TELECOMUNICACIONES

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE TELEMATICA

POPAYÁN

2004

ANEXO A

TÉCNICAS DE CODISEÑO ELECTRÓNICO APLICADAS A SISTEMAS DE TELECOMUNICACIONES

CONTENIDO

LISTA DE FIGURAS

INTRODUCCION

El gran avance que ha tenido la microelectrónica y las altas escalas de integración que se han logrado alcanzar, han conseguido la construcción de sistemas muy complejos en un simple chip. Este hecho ha traído como resultado la introducción de nuevas herramientas enfocadas a la especificación de dichos sistemas que pretenden aumentar la productividad de las empresas y reducir los costos en desarrollo.

Uno de los elementos más versátiles que tiene hoy en día el mundo electrónico es el procesador, el cual trajo la introducción del software como parte fundamental del desarrollo de la gran mayoría de los sistemas, debido a la gran capacidad de abstracción, modelos de solución y pruebas con los que este puede ser descrito sin que pierda sus propiedades, además su rehusabilidad, portabilidad, falibilidad de adaptación y uso, ha logrado reducir considerablemente los procesos y los costos tanto en desarrollo como en producción. Tanto es así que la tendencia en las herramientas de síntesis HW es asemejarse en la mayor manera a los lenguajes de programación SW para concurrir en sistemas codiseñados (Sistemas mixtos Hw/Sw). Sin embargo la gran debilidad de los modelos de procesamiento de instrucciones esta en su capacidad de respuesta, la cual esta ligada por supuesto a la velocidad de procesamiento de la CPU (solamente procesa unas cuantas instrucciones simultaneas) y a la complejidad del sistema siendo esto es un punto critico, sobretodo en los sistemas de comunicaciones, en donde sistemas altamente complejos deben responder a severos y estrictos requerimientos de velocidad.

Es así como se impone la tendencia de los sistemas HW/SW que descargan las labores críticas a elementos HW y entregan la responsabilidad de procesamiento lógico más liviano pero no por ello menos complejo, al SW del procesador, esta tendencia ha logrado amentar la velocidad de los procesos de desarrollo y disminuir los costos y ha traído con ello la introducción herramientas CAD para simulación y síntesis de sistemas Hw que se asemejan a los ambientes de desarrollo Sw y que interactúan con estas ultimas para lograr una completa simulación de los sistemas antes de su implementación física, lo que

permite en su gran mayoría detectar fallos y ahorrar esfuerzos antes de plasmar los sistemas sobre el silicio.

1 REQUERIMIENTOS DE ESPECIFICACIÓN *[López-Hermida-Gesselhardt 98]*

Aunque el desarrollo hardware y software por separado, son disciplinas relativamente evolucionadas, el diseño de sistemas heterogéneos hardware/software, todavía es un campo abierto a la exploración, en donde uno de los principales problemas a resolver es el de especificación de el sistema completo que permita después separar sus funcionalidades para tomar la decisión de implementar unas en Hw y otras en Sw.

Uno de los aspectos determinantes en el proceso de desarrollo de sistemas heterogéneos esta en la partición HW/SW que se hace antes de empezar el co-diseño pues el resultado de esto determinara la complejidad y rendimiento de los subsistemas resultantes y del sistema total, así como también repercutirá en la velocidad y economía del desarrollo.

Los siguientes requerimientos ayudaran a dar una idea de las características que debe que debe tener el modelo de especificación del sistema co-diseñado, antes de determinar que partes se desarrollaran en HW y cuales en SW:

- Programadores.

En la mayoría de los casos, el comportamiento de un sistema puede ser descrito desde el punto de vista de un sistema secuencial. Sin embargo cuando se trata de sistemas de comunicaciones los requerimientos de rendimiento son muy altos este hecho dificulta un poco la especificación de un sistema completo como un algoritmo secuencial. A pesar de ello estos sistemas pueden seguir siendo especificados secuencialmente sin que ello implique que su implementación deba ser hecha de la misma manera, por ello el algoritmo secuencia será un de las técnicas preferidas en la especificación de un sistema heterogéneo.

- Comunicación

Los mecanismos de comunicación entre tareas se pueden clasificar en dos grupos; los de memoria compartida y los de paso de mensajes. Aunque los mecanismos de memoria compartida son económicos y más fáciles de implementar que los de paso de mensajes, se debe tener especial cuidado Cuando se usan para intercambio de información entre tareas concurrente, pues puede dar lugar a complicaciones y a errores muy difíciles de detectar.

- Descripción de los estados del sistema.

La descripción a nivel de estados es una técnica que facilita la abstracción tanto de un sistema HW como de un sistema SW es por ello que viene a ser una herramienta muy potente a la hora de hacer el paso de un módulo entre su especificación y su implementación.

- Particionamiento Jerárquico.

La división que se haga de un sistema siempre ayudará a disminuir su complejidad y esto además presenta un mejor panorama a la hora de decidir que funciones serán implementadas en HW y cuales en SW. Es muy importante una buena división para realizar un buen co-diseño, la partición debe ser hecha como una partición esructrural de componentes y como una partición comportamental del sistema.

- Comportamiento completo.

Una visión completa del comportamiento del sistema ayuda a comprender mejor los comportamientos de los subsistemas, lo que facilita la división jerárquica y verificación de la especificación completa del sistema.

- Concurrencia de eventos

Cuando se diseñan sistemas de comunicaciones complejos la concurrencia de eventos siempre es uno de los ingredientes fundamentales en el desarrollo y es un punto determinante a la hora de particionar el sistema ya sea arquitectural o comportamentalmente justo en el momento que se toma la decisión de decidir que elementos van a ser HW y que elementos se implementaran en SW, pues como ya sabemos las características secuénciales de los procesadores (elementos encargados de ejecutar el SW) limitan sus capacidades de respuesta y obligan en algunos casos a utilizar sistemas multiprocesador y/o también a sintetizar elementos HW que desarrollen efectivamente las labores de concurrencia de tareas.

- Sincronización.

Los mecanismos de sincronización entre tareas son muy útiles a la hora de solucionar problemas de comunicación y permitir la integración de múltiples sistemas pues ayudan a que los sistemas se estabilicen antes de tomar decisiones críticas que pueden afectar a otros procesos o tareas.

- Manejo de excepciones.

A menudo la ocurrencia de eventos no deterministicos que pueden afectar el correcto funcionamiento de los procesos y tareas hacen que los sistemas se vuelvan inestables y producen fallas fatales que deben ser capturadas como fallas eventuales y llevar al reestablecimiento del sistema para que este se haga estable de nuevo.

- Respuestas no determinísticas.

La posibilidad de no poder siempre predecir exactamente los resultados de algunos sistemas ante algún determinado evento debe ser tenida en cuenta siempre que se diseña un sistema complejo de comunicaciones, por varios motivos los sistemas deben ser diseñados bajo estos parámetros y a pesar de ello no deben perder su estabilidad. Existen dos clases de respuestas indeterministicas; funcionales y temporales. Las funcionales son aquellas que para un determinado evento pueden responder de distintas formas de acuerdo al estado del sistema y las temporales son aquellas cuyo tiempo de respuesta para un mismo evento puede variar severamente dependiendo de muchos factores, este comportamiento esta muy ligado con la manera en que se hace el particionamiento del sistema HW/SW.

- Temporización.

Como se dijo anteriormente, la respuesta temporal de algunos sistemas esta ligada con la partición HW/SW que se haga en su diseño. Las respuestas temporales de un sistema son a menudo las más críticas y pueden llegar a determinar la imposibilidad de realización de alguno, pues son ellas las que son determinadas por el ambiente externo al sistema.

- Orientación a objetos.

La técnica de orientación a objetos es una de las más poderosas porque además de permitir una aproximación al mundo real para reducir la complejidad del sistema, también presenta características muy útiles como la rehusabilidad el manejo distribuido de elementos. Además de que existen muchas herramientas para describir sistemas a partir de objetos.

2 DISEÑO ESTRUCTURADO DE SISTEMAS HETEROGÉNEOS (CODISEÑO)

El diseño estructurado es una herramienta clave para el desarrollo de sistemas digitales, la figura 1 muestra una arquitectura de diseño top-down basada en mantener una independencia mínima de la tecnología en las etapas tempranas del diseño que aumenta a medida que se avanza en al proceso manera incremental, lo que facilita tomar decisiones a nivel de partición HW/SW en etapas tempranas del diseño, a continuación analizaremos las fases de esta arquitectura haciendo énfasis en al análisis del sistema. [5]

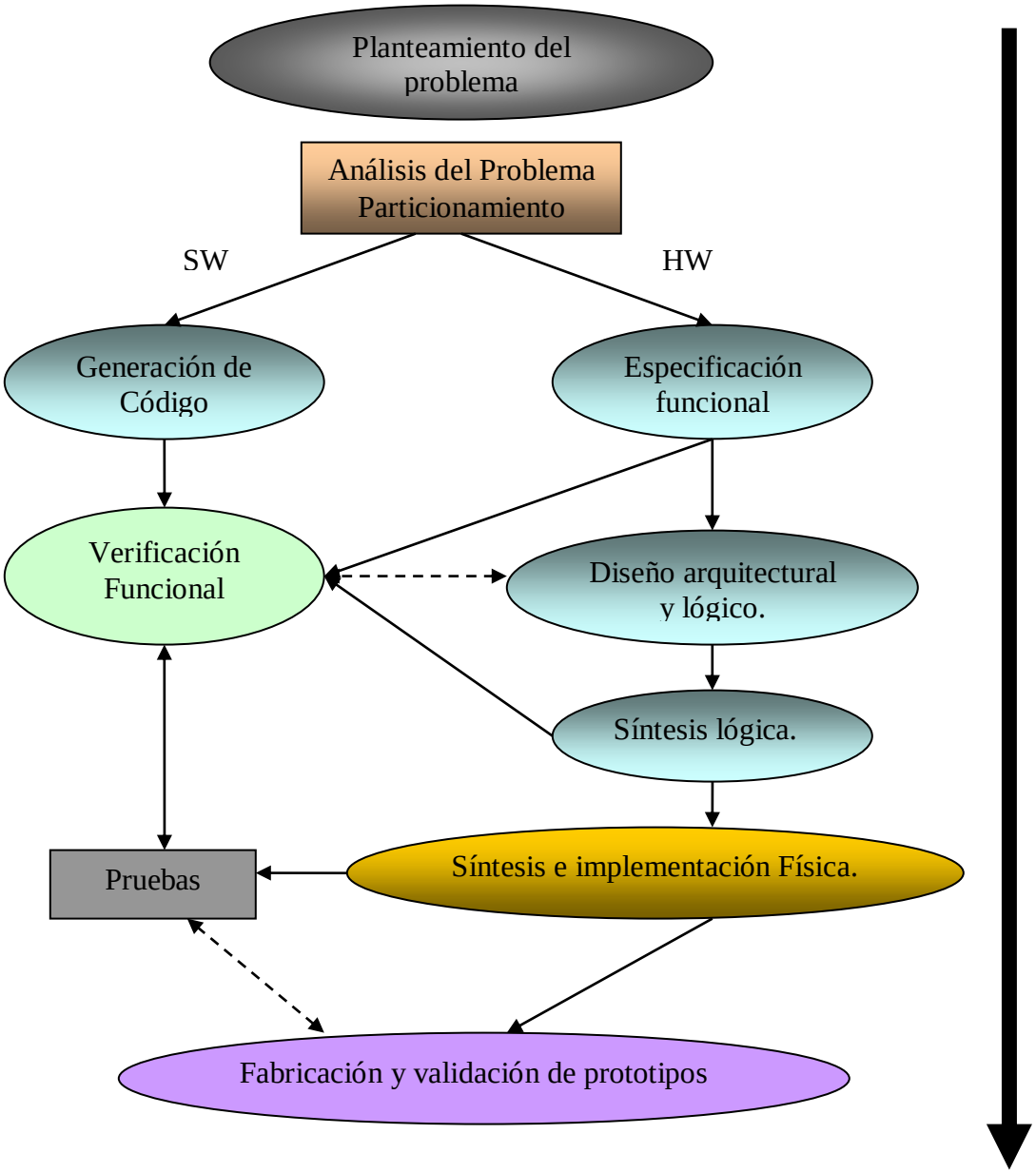

Figura 1 Fases del diseño estructurado Top-Down

2.1 ESPECIFICACIÓN FUNCIONAL

En esta etapa se hace un planteamiento detallado de la funcionalidad del sistema; el plan de pruebas previsto en su desarrollo, las herramientas para su diseño, sin olvidar que para todo esto se debe tener en cuenta la tecnología elegida. Posteriormente se hace una partición del sistema en bloques funcionales de entrada/salida, se planifica y se modela mediante lenguajes de especificación de alto nivel para descripción de software y hardware, para con esto para a hacer una primera verificación funcional a alto nivel.

2.2 VERIFICACIÓN FUNCIONAL

De acuerdo al esquema planteado en la figura 2, La especificación funcional planteada de acuerdo a una o varias alternativas, es enfrentada a través de un simulador a un banco de pruebas realizadas por el diseñador, llevadas a cabo sobre un conjunto de librerías qué reúnen las características reales de los componentes a nivel de simulación (dependencia de la tecnología) con el fin de validar los resultados y realizar los ajustes pertinentes a este nivel.

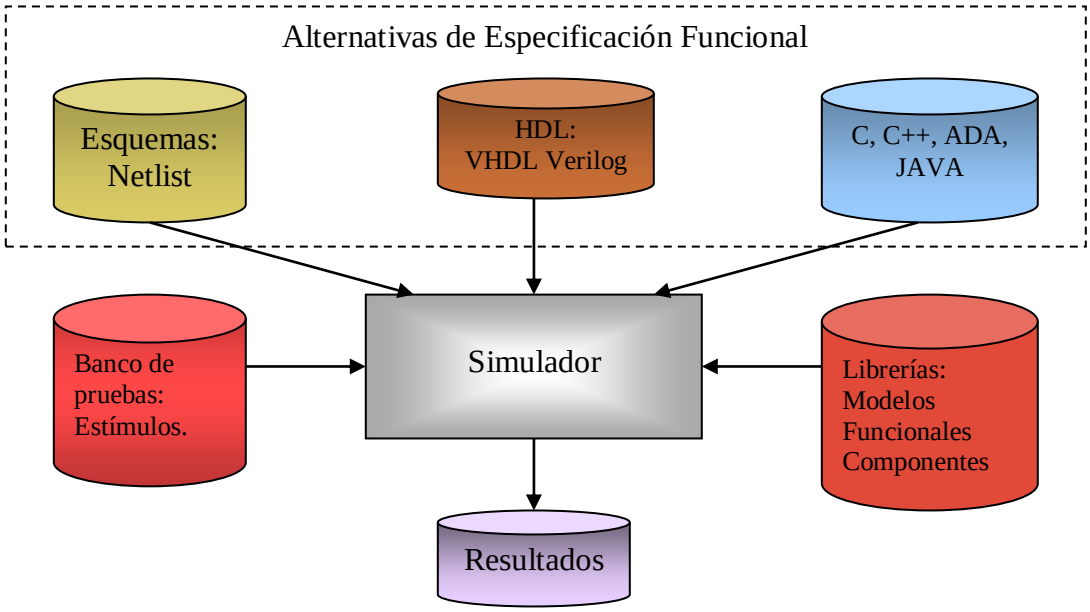

Figura 2 Esquema de verificación funcional

2.3 DISEÑO DETALLADO

2.3.1 Arquitectural

El diseño arquitectural busca dividir el sistema total en distintos bloques funcionales con el fin de reducir la complejidad. Para cada bloque funcional se especifica un banco de pruebas propio con la condición que las interfaces entre bloques funcionales deben quedar definidas en forma muy precisa y clara.

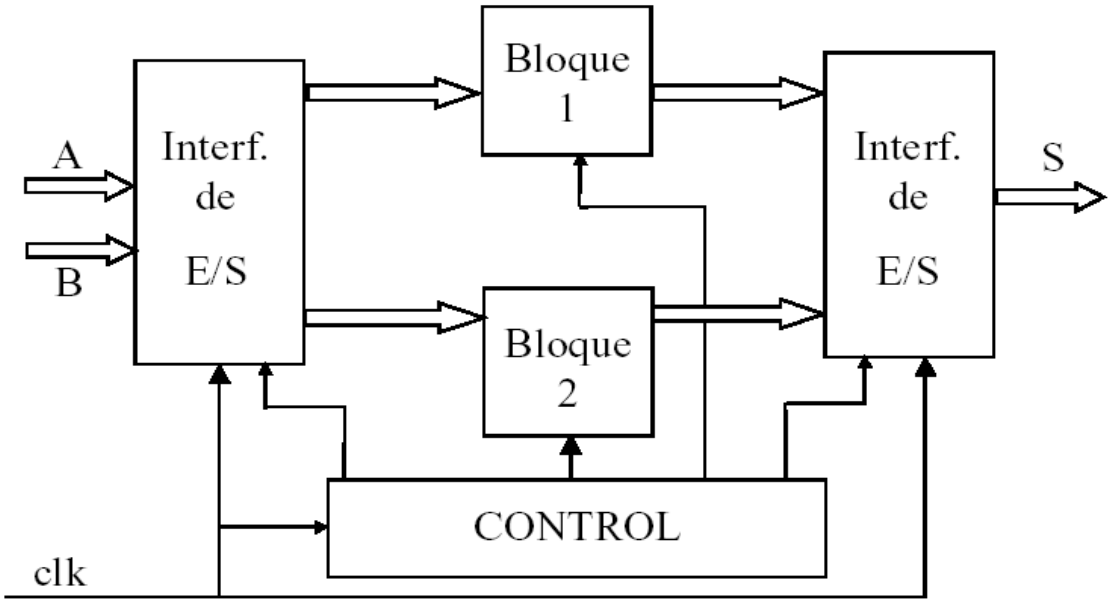

Figura 3 División Funcional En Bloques

2.3.2 Lógico

Busca obtener un diseño basado en compuertas y lógicas (Puertas lógicas, flip-flops, memorias, mux, etc.) para cada bloque arquitectural. Existen herramientas automáticas que transforman una especificación funcional en un diseño lógico, estas pueden estar orientadas a algún tipo de algoritmo o pueden ser dependientes de algún fabricante o determinada tecnología.

Esta fase consume mucho tiempo, sobretodo en el refinamiento del diseño que busca obtener resultados óptimos que coincidan con los obtenidos en la verificación funcional. Es muy importante no desligarse de la tecnología de implementación, pues decisiones de diseño adecuadas para algunas tecnologías pueden no resultar muy convenientes para otras.

2.4 PRUEBAS

Las pruebas de un circuito no solo consisten en comprobar la funcionalidad del mismo, sino que trata de detectar eventuales fallos en el proceso de fabricación, en teoría se busca comprobar que el circuito este libre 100% de fallos.

Existe la posibilidad de añadir un hardware específico que permita realizar una cobertura de fallos, este procedimiento tiene una efectividad estimada del 93% y no busca una comprobación funcional sino una comprobación de accesibilidad y controlabilidad de los nodos. Su uso tiene sentido par ASICs, PCBs y Gatearrays en donde existe un proceso tecnológico de fabricación y no en los FPGAs donde el diseñador, sólo configura la unión a implementar sobre bloques configurables y a programar las interconexiones de pistas que ya existen.

Los fallos de fabricación se producen por fallos al realizar el “placement” de las celulas o componentes y su interconexión o “routing” y su separación, ya que esto se hace sobre un sustrato de silicio que generalmente puede tener impurezas

2.4.1 Objetivos De Las Pruebas

- Las pruebas buscan detectar piezas defectuosas en el proceso de fabricación.
- El diseño debe ir ligado a un diseño de las pruebas, es decir hay que ir diseñando, pensando en facilitar las pruebas.
- Hay que conseguir que el diseño sea controlable y observable:

    - ✓ Accesibilidad a los nodos desde el exterior
    - ✓ Particionamiento y paralelización de los diseños.
    - ✓ Scan-Path permitir caminos de rastreo.
    - ✓ BIST (Built-In Self Test), construcción paralela de un HW especifico de pruebas.

2.5 ALTERNATIVAS TECNOLÓGICAS DE FABRICACIÓN

2.5.1 Placa de Circuito Impreso (PCB)

Esta técnica es muy usada en nuestro país, pues existen multitud de herramientas para su diseño y da la posibilidad que el diseñador tenga control de las dimensiones físicas, colocación de los componentes y su interconexión (placement/routing), su proceso de desarrollo no es en principio muy costoso y permite adaptarse cambios tecnológicos en forma no tan drástica, permite fácilmente la utilización de distintas técnicas de pruebas.

Entre sus inconvenientes están que el tamaño físico de los prototipos puede llegar a ser exagerado y a disipar demasiada potencia entre conexiones muy largas, que también pueden dar lugar a efectos de ruido capacitivo e inductivo. En producción a grandes escalas puede llegar a ser muy costoso.

Se pueden clasificar según distintos características [6]:

- ✓ Bicapa/Multicapa
- ✓ Con componentes estándar y/o montaje superficial
- ✓ Taladros metalizados/sin metalizar
- ✓ Para diseño analógico, digital o mixto
- ✓ etc.

El “routing” y el “placement” se pueden hacer en forma manual, automática o semiautomática, la experiencia aconseja una colocación manual de las señales más críticas (reloj, reset, señales de control, buses de comunicación, etc), minimizando su longitud y evitando ángulos y recodos pronunciados, el resto de las señales se puede rutear en forma automática, delimitando áreas críticas no ruteables para proteger señales críticas. También es aconsejable, en diseños mixtos (Análogico/digital) separar bien las zonas en la placa y tratar con cuidado las masas para evitar el filtrado de ruidos entre zonas. Para diseños digitales no se debe perder de vista la tecnología utilizada (TTL, CMOS, etc), pues unas son más ruidosas y consumen distinta potencia necesitando más grosor y separación entre las pistas, con el fin de evitar ruidos por acoplamiento.

2.5.2 Circuitos de Aplicación Específica (ASIC)

2.5.2.1 Diseño Full-Custom.

No es una alternativa muy recomendable, pues son difíciles y costosos de verificar, su proceso de desarrollo es poco productivo, difíciles de adaptar a cambios tecnológicos y las herramientas de diseño no son muy conocidas, además que se necesita mucha experiencia del diseñador en el tema y precisa un contacto muy cercano con el fabricante. Sin embargo, cuando las características de especificación del circuito a diseñar (velocidad, consumo, disponibilidad de células estándar, etc) lo aconsejan entonces no hay más remedio.

2.5.2.2 Diseño Semi-Custom.

Es cuando el diseño esta ligado a células estándar prediseñadas. Es un diseño más tratable basado en multitud de herramientas, librerías de células y macros (HW y SW) garantizadas por los fabricantes, es más económico y fácil de probar, mediante el uso de herramientas especificas, generadas para esto.

Con la tecnología actual, CMOS por ejemplo, se pueden conseguir chips de varios millones de puertas y una frecuencia de funcionamiento por encima de los 500MHz. Las dimensiones del chip dependen de la tecnología. Un circuito de 100mm2 en CMOS de 0,5 micras de canal, es GIGANTESCO y de una complejidad enorme. Entre las clases de células y macros que se pueden encontrar están:

- ✓ Macro células (macros): RAM, ROM,
- ✓ Multiplicadores, etc.
- ✓ Bloques funcionales: CPU, RISC.
- ✓ Microcontroladores, UARTS, etc.
- ✓ Células analógicas: AOs, Convertidores A/D y D/A
- ✓ etc.

2.5.2.3 Gate Array.

El diseñador solo se preocupa por especificar las funciones que ha de realizar el diseño, las herramientas de diseño físico únicamente proporcionan la interconexión entre células y al igual que los ASICs es imprescindible las técnicas y la circuitería para pruebas.

2.5.2.4 Lógica Programable.

Son muy versátiles y existen gran variedad de dispositivos y herramientas de lógica programable, el diseñador solo los configura, por lo tanto no hay un proceso tecnológico y no tiene sentido el test. Y se pueden clasificar como muestra la figura 4. Los tipos más conocidos de PLDs son:

- ✓ PLD: Programmable Logic Device
- ✓ PLA: Programmable Logic Array
- ✓ PAL: Programmable Array Logic
- ✓ GAL: Generic Array Logic
- ✓ CPLD: Complex PLD
- ✓ EPLD: Erasable PLD
- ✓ HCPLD: High Complexity PLD
- ✓ FPGA: Field Programmable Gate Array

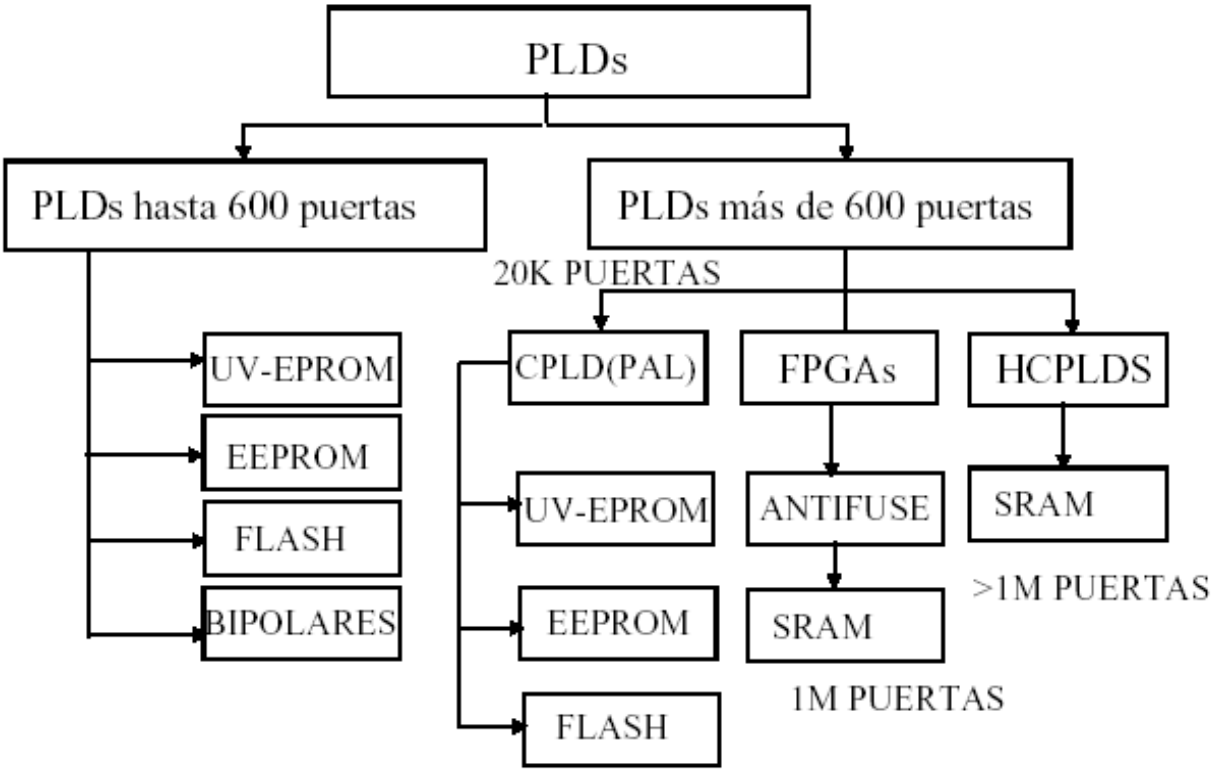

Figura 4 Clasificación de los PLDs

2.6 FABRICACIÓN Y VALIDACIÓN DE PROTOTIPOS

De acuerdo a la tecnología de implementación escogida, se deben tener diferentes consideraciones las cuales generalmente se salen de las manos del diseñador y vienen a depender del fabricante y de que también especificadas estén los vectores de pruebas y estímulos (los mismos utilizados para el diseño) para la validación de los prototipos fabricados, así como la tecnología a utilizar, tamaño, descripción funcional.

En el caso de los dispositivos programables se deben tener las herramientas correctas de programación, para los dispositivos con los cuales sea posible la implementación del prototipo diseñado, en este caso no tiene sentido el test, pues no se realiza ningún proceso tecnológico, solo se configuran los dispositivos.

2.7 ANÁLISIS DEL SISTEMA Y PARTICIÓN

En el codiseño, uno de los aspectos claves esta en el particionamento, en el cual no solo se determinan cuales bloques van a ser implementadas en software y cuales en hardware, sino también se determina que tipos de componentes se usaran (ASICs, ASIPs, Memorias, PLDs, etc.), y los escenarios temporales de asignación de tareas a los bloques funcionales. Es por ello que el particionamiento afectara directamente al resto del desarrollo en cuanto al desempeño y costo del sistema.

Mediante un análisis del problema el diseñador deberá decidir que implementar en SW y que implementar en HW, para ello el diseñador puede utilizar algunas herramientas metodologicas que le permitirán tomar las decisiones, sin embargo, se piensa que las mejores herramientas son:

- ✓ La experiencia
- ✓ El conocimiento de los entornos de diseño
- ✓ El conocimiento de tecnologías disponibles
- ✓ El Gran esfuerzo de investigación en Codiseño

Sin embargo, inclusive en este punto es necesario no olvidar las posibles tecnologías de implementación.

Para hacer el particionamiento del sistema, el diseñador puede guiarse por distintos clases de algoritmos planteados para ello, que conservan algunos puntos en común como:

- ✓ Todos ellos son procedimientos de optimización guiados por una función global de costo.
- ✓ Hay siempre un soporte estructural basado en gráficas.
- ✓ Existen siempre unas métricas definidas en cuanto al tiempo de ejecución del software y capacidad (Memoria de datos y programa).

A continuación se darán unos parámetros que podrán caracterizar el proceso de particionamiento [1, López-Hermida-Gesselhardt 98]:

- **Modelo del sistema para particionamiento.**

Antes de empezar el proceso de particionamiento, la descripción inicial del sistema es trasladada en una representación intermedia, más conveniente para su manipulación. Esta representación es un mapeo del modelo del sistema y mantiene una estructura de datos, una estructura comportamental y la información requerida para llevar a cabo el proceso de particionamiento. Este modelo no corresponde necesariamente al modelo de especificación, dado que este debe ser más adecuado para los algoritmos de particionamiento, los modelos más conocidos son

- ✓ Control/Flujo de datos
- ✓ Representaciones basadas en FSM.
- ✓ Redes petri.
- ✓ Gráficas estructurales.

El modelo interno debe definir el tipo de granularidad con la que será particionado el sistema, la cual describe como puede ser dividido el sistema en un grupo de objetos distribuidos en varios sistemas de componentes; Existen dos tipos de Granularidad; Tosca y Fina.

- **Costo por función.**

La evaluación de la conveniencia de una partición en objetos hardware software, no es directa, existen muchos factores que afectan estas decisiones, el costo por función esta ligado a factores como; los costos de fabricación, el desempeño del sistema, violación de restricciones. En general se considera que la implementación hardware eleva los costos pero mejora el desempeño mientras que la implementación software es más simple, más barata y más rápida pero su desempeño es más bajo. De acuerdo a esto se asumen algoritmos en los cuales se busca cumplir con las restricciones más severas usando hardware y para el resto se busca la solución más rápida y económicamente más conveniente

- **Arquitectura objetivo.**

La arquitectura objetivo es la descripción de los componentes que interactúan entre si por medio de distintas clases de conectores. Las arquitecturas codiseñadas se caracterizan por tener al menos un componente programable (Procesadores o microcontroladores estándar) y un circuito dedicado con funcionalidad especifica (Generalmente un ASIC) usando un mecanismo de comunicación de memoria compartida entre ellos. Es muy difícil clasificar los tipos de tendencias en cuanto a arquitecturas se refiere [2], pero se pueden nombrar cuatro principales:

- ✓ La clásica, basada en un microprocesador +ASIC + Memoria + interfase de bus.
- ✓ La orientada a datos, Es igual a la clásica pero basada en el uso de DSPs.
- ✓ La de sistemas on-chip que usa un procesador integrado en un chip con ASICs.
- ✓ La de sistemas de tiempo real distribuidos.

- **Dominio de la aplicación.**

El dominio de la aplicación es indispensable, pues es ilógico pensar en un ambiente general de codiseño, pues cada dominio tiene distintas tendencias, la mejor división en cuanto a esto esta en clasificarlo en dos grandes grupos, sistemas de control y sistemas de procesamiento de datos (Telecomunicaciones). Mientras los de control se basan en una granularidad tosca, pues sus restricciones básicamente están en sus entradas/salidas y no tienen requerimientos severos de tiempo, los de procesamiento de datos se adecuan a la granularidad fina pues tienen muchos caminos de procesamiento y grandes restricciones temporales.

Existen gran número de algoritmos en el área del codiseño para particionamiento en distintos marcos de referencia, de los cuales es muy difícil hacer un estudio comparativo, los más conocidos se muestran en la tabla 1.

Procedimiento	Granularidad	Algoritmo	Grupo
Automático	Tosca	Basado en lista de cronológica	Ptolemy (Berkley)
		Particionamiento Hardware	Fvahid (Riverside)
		Heurística (SA, TS)	P. Eles (Linköping)
	Fina	Enfriamiento simulado	Cosyma (Braunshweig)
		Grupo migración	Vulcan (Standfor Univ)
		Programación dinámico	Lycos (Tech.Denamarc)
Manual	Tosca		Polis, Chinook, CoWare

Tabla1 Algoritmos conocidos para particionamiento

3 JAVA COMO LENGUAJE DE ESPECIFICACIÓN PARA SISTEMAS CTI

Como un lenguaje derivado del C, JAVA se presenta una alternativa muy atractiva para especificación de sistemas de telecomunicaciones heterogéneos ya que gracias a su concepción para sistemas distribuidos, permitiría dar flexibilidad a los procesos de especificación para granularidad fina. Además JAVA cumple con todas las requerimientos de especificación presentados, exceptuando los requerimientos para especificación temporal, pues sus características de manejo de procesos en tiempo real son restringidas. Sin embargo, la introducción del RT-JAVA como lenguaje de programación para sistemas operativos de tiempo real puede facilitar el manejo de las especificaciones temporales. Entre las características más representativas de JAVA para el caso están:

- ✓ Lenguaje de programación orientado a objetos.
- ✓ Manejo de herencia simple.
- ✓ Manejo de hilos o tareas concurrentes.
- ✓ Manejo de excepciones
- ✓ Manejo de interrupciones y eventos.
- ✓ Capacidad de sincronización para acceso a procedimientos compartidos.
- ✓ Comunicación distribuida entre procesos.
- ✓ Acceso a puertos.

Para algunos sistemas CTI, en donde el objetivo es llevar la computación hasta un hardware especifico de telefonía, resulta conveniente pensar en una arquitectura objetivo basada en un sistema coprocesador comunicado por un puerto estándar de comunicación figura, teniendo en cuenta las altas velocidades que estos pueden alcanzar hoy en día. El objetivo es manejar eficientemente un Hardware de comunicaciones orientado a telefonía por medio de un procesador con periféricos de aplicación específica y controlar este desde una plataforma de cómputo convencional con un sistema operativo de alto nivel.

Se propone hacer la especificación del sistema completo en RT-JAVA, que permita descomponerlo en un gran numero de funciones (granularidad fina) de procesamiento de información. Posteriormente las funciones se deben agrupar en tres grupos:

- ✓ Funciones de alto nivel.
- ✓ Funciones especializadas de alto rendimiento.
- ✓ Funciones críticas en el tiempo y/o con interfaces no digitales.

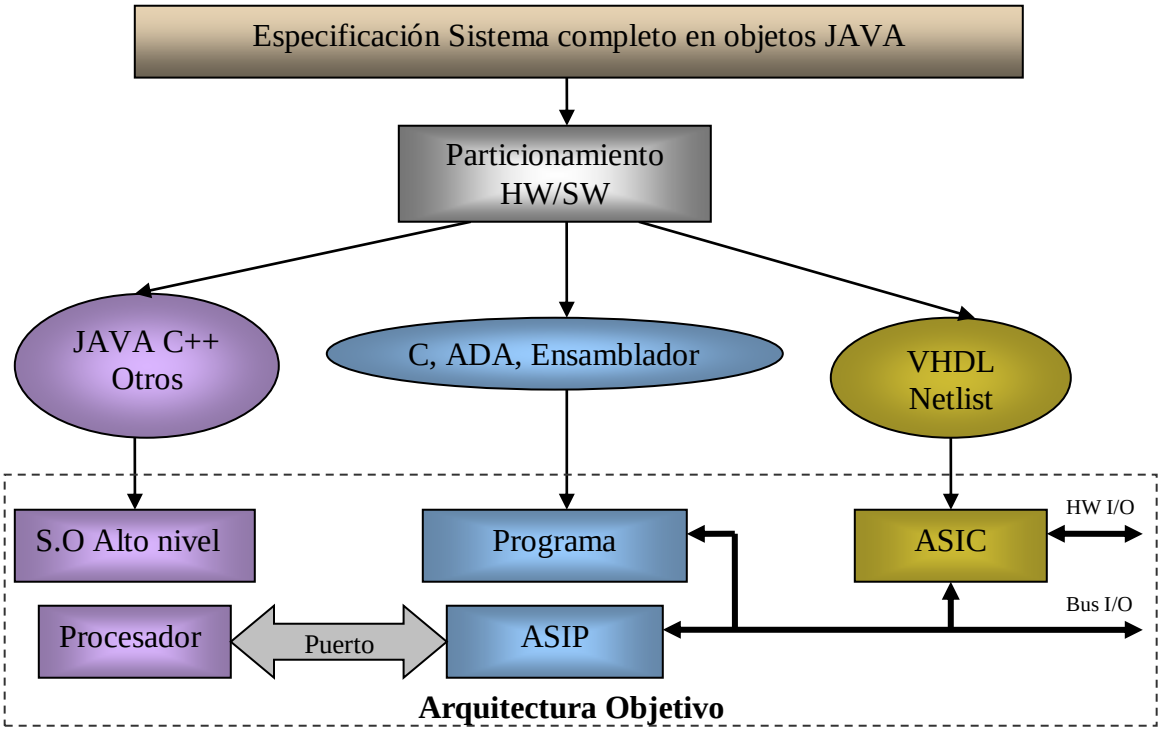

Figura 5 Metodología de codiseño propuesta para sistemas CTI

De acuerdo a esto las funciones de alto nivel serán implementadas sobre un sistema operativo de alto nivel, en general estas funciones se pueden identificar por sus grandes necesidades de interacción hombre-máquina, manejo de protocolos de capas altas, grandes necesidades de almacenamiento de información o por tener restricciones temporales poco exigentes. Esas funciones pueden ser dejadas en java o trasladadas a otro lenguaje de alto nivel, adaptando sus mecanismos de comunicación a los puertos de comunicación disponibles.

Las funciones especializadas de alto rendimiento (Tiempo real), de manejo de información a alta velocidad con bajos requerimientos de almacenamiento, concebidas observando las funcionalidades especiales del ASIP para conservar correspondencia en la granularidad funcional. Estas funciones pueden ser trasladadas a C, ADA o

ensamblador según lo requiera, para su compilación y ejecución desde el núcleo de un procesador especializado o ASIP que puede por si mismo suplir algunas funcionalidades planteadas en JAVA, mejorando así el desempeño del sistema.

Las funciones críticas en el tiempo corresponden a funciones de alto desempeño en tiempo real o con interfaces de línea analógicas. Estas pueden ser trasladadas a un lenguaje de especificación HW como VHDL o a esquemas Netlist para la configuración o fabricación del ASIC.

4 CONCLUSIONES

Se ha presentado una breve visión sobre lo que representa el diseño de sistemas heterogéneos, basados en una metodología para diseño estructurado. Como se pudo notar, el tema del codiseño es todavía un campo poco explorado y su principal problema reside en la especificación del sistema completo a diseñar y la partición de este en entidades funcionales hardware y software.

En general las decisiones claves del codiseño siempre están relacionadas con dos aspectos básicos, desempeño y economía. El hecho que los puntos críticos de decisión, sean aspectos muy difíciles de ponderar de una manera precisa, es una de las principales razones por las cuales no se avanzado mucho en la especificación de metodologías automáticas, que permitan tomar las decisiones del particionamiento del sistema de una manera más práctica. Hasta ahora la experiencia del diseñador sigue siendo la herramienta más fuerte en cuanto a la toma de decisiones en el particionamieto Hw/Sw.

El particionamiento y especificación del sistema, siempre es más fácil cuando se tiene planteada de antemano la arquitectura objetivo, como es el caso de la metodología para sistemas CTI planteada en el presente texto, por eso siempre es recomendable tomar la experiencia de sistemas similares ya desarrollados con éxito, para adoptar una arquitectura objetivo similar que facilite la toma de decisiones en el codiseño.

REFERENCIAS

[1] Lopez-Hermida-Gesselhardt, Advanced Techniques for Embedded Systems Design and Test, Klwer Academic Publisher 1998.

[2] HILL-PETERSON Sistemas Digitales, Organización y Diseño de hardware, LIMUSA 1994.

[3]http://www.upv.es/die/sci/ficheros_html/proy_c.htm

[4] Torrubia Sáez, Codiseño Hardware/Software y Driver bajo Windows NT para el Bridge PCI-SCI (1997).

[5] DIE-UPM, Programa de Diseño de Circuitos y Sistemas Electrónicos - 2002-2003.htm

[6] Montolli, Técnicas de fabricación de circuitos impresos 1984]