UNIDAD 5
Arquitectura FPGA

5.1 Fundamentos Teóricos.

Un *FPGA* (*Field Programmable Gate Array - Arreglo de Compuertas Programable en Campo*) es, al igual que un PLD y un CPLD, un ASIC programable, pero que requiere estrategias de programación diferentes y que por ende su arquitectura no se basa en arreglos tipo *PLA*.

En otras palabras, un FPGA se compone de elementos con recursos no comprometidos que pueden ser seleccionados, configurados e interconectados por usuario. Recordemos que en un PLD las interconexiones entre los elementos ya están hechas, solamente podemos habilitar o deshabilitar la interconexión; en el caso de un FPGA no hay nada interconectado.

Estos dispositivos se componen de cierto número de *Módulos Lógicos*, que determinan la capacidad del dispositivo. Los módulos son independientes entre sí y pueden interconectarse para formar un modulo más complejo. Dependiendo del fabricante, estos módulos pueden ser *Bloques Configurables*, como en los FPGA's de *Xilinx*, o bien, *Elementos de Función Fija* formados por arreglos de compuertas, como en el caso de los dispositivos de *Actel*[1].

Los módulos en un FPGA, se interconectan por medio de *Canales Configurables* (ver **figura 5.1**). Al proceso de interconexión, se le conoce como *Enrutamiento* y consiste en determinar la mejor estrategia de interconectar los módulos, ya sea en forma manual o mediante alguna herramienta de *diseño electrónico (EDA)*.

Por la capacidad de un FPGA (el más sencillo contiene 1,200, hasta los más grandes que contienen 4,000,000[2] de compuertas lógicas), se dice que son dispositivos para diseños *LSI* y *VLSI*.

[1] *En el mercado actual, los dispositivos fabricados por* Xilinx *y por* Actel, *son los más utilizados comercialmente según un estudio realizado por la IEEE.*
[2] *Dato actualizado hasta el primer semestre del año 2000. El XCV3200 de Xilinx es el FPGA con mayor capacidad en el mercado.*

La gran ventaja de utilizar estos dispositivos radica en que todo el desarrollo se lleva a cabo en un solo ambiente de trabajo. El diseñador propone la función lógica a realizar y en base a métodos de descripción define los parámetros de su problema. Esto se hace por medio de código programable, que puede ser un *Lenguaje de Descripción de Hardware*, o bien, un *diagrama esquemático* de conexiones.

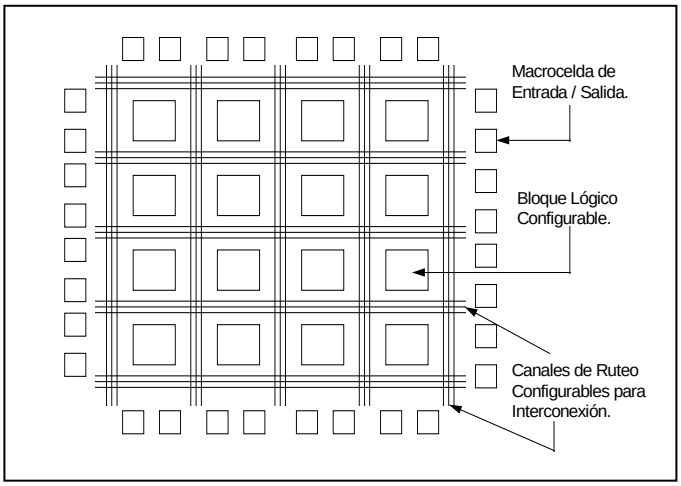

Figura 5.1. - Arquitectura demostrativa, no detallada, de un FPGA de Xilinx.

Una vez delimitado el problema, se optimiza su representación lógica mediante métodos de minimización (*Síntesis Lógica*); posteriormente se simula, lógica y eléctricamente (*Simulación*). Con la ayuda del software actual que es muy sofisticado, este último paso puede llegar a ser muy aproximado al comportamiento real del dispositivo.

Se selecciona, entonces, el dispositivo que mejor se adapte a las condiciones de nuestro problema según criterios de *capacidad, velocidad, consumo de energía, costo*, etc., y finalmente se programa en campo.

Después de la programación se puede ejercer una *optimización* si se cree necesario hacer modificaciones en el diseño. Sólo bastaría con hacerlas en el código y repetir los pasos anteriormente descritos. El dispositivo se podrá reconfigurar nuevamente hasta en unos 200 intentos, que según el fabricante, es el tiempo de vida promedio.

De acuerdo a varias bibliografías consultadas, las principales ventajas de diseñar sobre FPGA's se enlistan a continuación:

a) *Minimización del número de componentes en un diseño*. Con esto se reducen los gastos de inventario, inspección y prueba, así como el número de fallas a nivel circuito

impreso, propiciando un ahorro de espacio físico. Así, una medida de la eficiencia de un Dispositivo Programable se expresa mediante el número de dispositivos de función fija (Circuitos Integrados de Catálogo) que pueden remplazarse.

b) *Reducción en el tiempo de diseño*. Debido a su naturaleza programable, reducen el tiempo y los costos de desarrollo, no sólo de nuevos productos sino también de aquellos que requieren modificaciones (*reingeniería*), ya que son reutilizables tantas veces como sea necesario. Esto último se debe a que los cambios en el diseño son realizados mediante una nueva programación que se prueba inmediatamente si se está utilizando un dispositivo programable en el mismo circuito (IS).

c)Uso de una gran variedad de herramientas de *Diseño Asistido por Computadora (CAD*), disponibles actualmente en el mercado. Estas herramientas promueven y facilitan el diseño sobre este tipo de dispositivos. Así mismo, no se requiere de grandes recursos de cómputo

5.2 FPGAs Coarse Grained (Granularidad Gruesa) y Fine Grained (Granularidad Fina).

La complejidad de los elementos contenidos en los módulos lógicos, es factor determinante para medir el desempeño de un FPGA. Recordemos que independientemente del fabricante, los módulos lógicos realizan las operaciones básicas que en conjunto representan la función que operará el FPGA. Como ya lo habíamos mencionado con anterioridad en la **Unidad 3**, en el apartado referente a los ASICs Programables, un FPGA al igual que un CPLD, son dispositivos con una arquitectura avanzada.

La arquitectura avanzada, llamada así por la densidad de sus componentes y sus estrategias de interconexión entre módulos, tiene dos derivaciones estructurales de acuerdo al tipo de módulos lógicos que la conforman: *Granularidad Gruesa* y *Granularidad Fina*.

Los módulos lógicos en una arquitectura de *Granularidad Gruesa (Coarse Grained – CG)*, son módulos grandes generalmente consistentes de una o más *Tablas de Búsqueda* y dos o más flip – flop's. Si analizamos la configuración física de cada módulo lógico o *Grano*, podemos advertir que particularmente cada uno de ellos, puede ejecutar una función simple o una función compleja, que adicionada a otra función ejecutada por un módulo diferente conforman un sistema más complejo. La

Tabla de Búsqueda (*LookUp Table - LUT*) actúa como una memoria donde se encuentra almacenada la tabla de verdad que representa la función lógica del circuito, así en una LUT es posible implementar cualquier función deseable. Por lo anterior se dice que un módulo que contiene elementos como éstos, es un *Grano Grueso*.

Por otra parte, una arquitectura de Granularidad Fina (Fine Grained – FG), está estructurada por una gran cantidad de módulos lógicos pequeños que realizan funciones relativamente simples. Cada Grano o módulo en este tipo de arquitectura está compuesto de un circuito de dos entradas que realiza una función lógica determinada, o en algunos otros casos por un multiplexor 4 a 1. Adicionalmente contienen un solo flip – flop.

Como hemos de suponer, entre ambas granularidades existen diferencias. La CG permite implementaciones menos detalladas debido a que desde un nivel muy básico se tienen módulos complejos. Sin embargo, son dispositivos con una gran densidad de compuertas, ya que el hecho de utilizar LUT deja entrever que se pueden realizar diseños grandes. Los FPGAs con tecnología SRAM, como los de *Xilinx* o los de *Altera,* tienen arquitecturas CG, y son *ISP (Programables en Sistema)*.

La arquitectura FG, como la de los FPGAs de *Actel*, está relacionada con la tecnología de programación Antifuses. La simpleza de la constitución de cada módulo, permite implementaciones más detalladas y sobre todo más veloces. Debido a que para llegar a implementar una función compleja se requiere el uso de varios módulos, se trata de dispositivos de alta densidad de módulos (comparándolos con los anteriores), sien embargo, realmente son FPGAs con una gran cantidad de módulos (por su tamaño), pero cada módulo tiene un número mínimo de compuertas lógicas a diferencia de los FPGAs CG, que pueden tener menor número de módulos pero cada módulo tiene un número grande de compuertas lógicas. La apreciación puede ser engañosa. Los FPGAs con tecnología Antifuses como los de la arquitectura FG, son Programables Fuera del Sistema (OSP).

4.3 FPGA OTP (One Time Programmable – Programable Una Vez).

Anteriormente, en la **Unidad 3**, comentamos acerca de los *ASICs Programables Una Sola Vez (One Time Programmables - OTP*) y los *Programables Varias Veces (Many Times Programmables – MTP),* así como de la llamada *Lógica Configurable*.

En el contexto de desarrollo de sistemas empleando FPGAs resulta muy socorrido el uso de estos conceptos. No redundaremos en éstos, nos limitaremos a aplicarlos en las arquitecturas implicadas.

En el mercado del diseño en FPGAs es muy importante conocer las ventajas que nos ofrece un dispositivo en relación al otro. Como resulta obvio, un dispositivo OTP no es nada conveniente si estamos desarrollando prototipos experimentales, en este caso la mejor opción es emplear dispositivos MTPs. Un FPGA OTP, se dice que es un dispositivo con *Lógica No Reconfigurable*, debido a que su lógica solamente puede ser configurada en una sola ocasión. Por el contrario, un FPGA MTP es un dispositivo de *Lógica Reconfigurable*.

Debido a la tecnología de programación, un OTP es básicamente programado mediante una tecnología *Antifuses* que una vez establecidas las conexiones y desconexiones, ya no es posible reestablecer el circuito. Los FPGAs de *Actel*[3] utilizan esta tecnología, que a pesar de su desventaja en la reprogramación adiciona satisfactoriamente otras ventajas como lo son: la velocidad de desempeño (cuestión que analizaremos cuando tratemos el apartado **5.4**) y la prevención de fallas en el momento en que se descarga una configuración en un dispositivo *ISP (Programable en Sistema*)[4]. Además no requieren de un soporte en hardware adicional para mantener su configuración como en el caso de los SRAM, ya que no son volátiles.

Los FPGAs de *Xilinx* y los de *Altera* son MTP, programándose mediante tecnología SRAM, por lo que pueden ser Programados en Sistema, a diferencia de los OTP que se programan *Fuera de Sistema*. Debido a que los SRAM necesitan almacenar su configuración en RAM, son módulos volátiles y requieren en ocasiones una memoria exterior para hacerlo, lo que implica mayor hardware de soporte. Así mismo, se trata de dispositivos más lentos en comparación a los OTP, pero con la gran ventaja de que son completamente Reconfigurables y tienen una mayor capacidad de compuertas lógicas.

5.4 Algunas Familias de FPGAs Comerciales, Utilizadas en el Mercado Actual.

[3] *Posteriormente comentaremos que Actel fabrica otra clase de dispositivos a los que llama Pro ASICs, los cuales no utilizan una tecnología de programación Antifuses, sino FLASH EEPROM, por lo que resultan reprogramables.*

[4] *Esta ventaja es muy mencionada por Actel y la justifica exponiendo que el sistema donde está empotrado el FPGA puede llegar a afectar mediante ruido eléctrico o algún otro factor, la configuración del dispositivo; o debido a que el cable que sirve de interfase de descarga de la configuración, presente alguna anomalía. Sin embargo, de acuerdo a nuestra experiencia personal, no llega a pasar y son fallas que se pueden prevenir.*

Los diseñadores coinciden en afirmar que desde el punto de vista usuario, existen tres fabricantes mayoritarios en la distribución de FPGAs y software de soporte: *Xilinx, Altera* y *Actel*. En el mercado mundial podemos encontrar otros tantos con producciones menores pero que figuran también como FPGAs útiles: *Lucent, Texas Instruments, Philips, QuickLogic, Cypress, Atmel*, etc. En este apartado nos enfocaremos en los tres principales, dando una breve introducción a las familias lógicas y sus características principales.

Todo fabricante ofrece la información de sus productos por Internet, donde podemos encontrar hojas de especificaciones, notas de aplicación (proyectos realizados) y tutoriales para el manejo de dispositivos y software de diseño, entre otras opciones. Se recomienda visitar los sitios WEB para familiarizarse con los términos y facilitar la búsqueda de información. En cada sitio WEB, es posible solicitar una copia gratuita de los *Data Books (manuales*), así como una copia de evaluación del software de diseño (en la Unidad 5, trataremos más a detalle lo referente al software de propietario). Por lo general, en el sitio WEB aparece la opción *"PRODUCTS" (productos*), con la que accedemos a las *Digital Libreries (Bibliotecas Digitales*[5]*)* que son los manuales que contienen la información técnica de cada familia disponible.

Las direcciones de los sitios son:

- **Xilinx:** http://www.xilinx.com
- **Altera:** http://www.altera.com
- **Actel:** http://www.actel.com

El desarrollo de sistemas en FPGAs en tan amplio y necesario en la actualidad, que la evolución en las técnicas y metodologías en diseño digital electrónico, ha hecho que en la mayoría de las Universidades adopten la enseñanza de estos conceptos como parte de la formación académica del profesionista en áreas afines. Cada fabricante tiene instituido un *University Program (Programa Universitario)* en el que donan a las Universidades, software de diseño y hardware (tarjetas de programación y dispositivos) tras una solicitud formal que justifique el uso de los recursos con propósito docente. En los mismos sitios WEB está la información pertinente.

5.4.1 FPGAs de Xilinx.

[5] *Llamadas así, simplemente por estar en formato accesible electrónicamente, y no a través de un libro consultable físicamente, como en el caso de un manual tradicional TTL. Además la distribución gratuita contiene la misma información que Internet en un Disco Compacto Multimedia con archivos PDF (Acrobat Reader), que también incluye el software instalable para leerlos.*

Xilinx está considerado como uno de los fabricantes más fuertes a nivel mundial. Sus FPGAs (también fabrica PLDs y CPLDs) están basados en la tecnología SRAM y son dispositivos *MTP*, *programables en sistema (IS*).

Sus principales familias de FPGAs son: *XC3000, XC4000, XC Virtex*, *y XC Spartan*. La estructura de estos dispositivos está compuesta por módulos lógicos llamados por Xilinx, *CLBs (Configurable Logic Blocks – Bloques Lógicos Configurables*), basados en *Tablas de Búsqueda (LookUp Tables - LUTs)*. Cada CLB contiene circuitos que les permiten realizar operaciones aritméticas eficientes (como la de un algoritmo de Suma Paralela). También los usuarios pueden configurar las tablas de búsqueda como celdas "read/write" (lectura/escritura)de RAM. A la vez, a partir de la serie XC 4000, se incluye un generador interno de señal de reloj con 5 diferentes frecuencias.

Además de los CLBs, los FPGA de este fabricante incluyen otros bloques complejos que configuran la entrada de los pines físicos que conectan el interior del dispositivo con el exterior, a estos bloques se les llama *IOBs (Input/Output Blocks – Bloques de Entrada/Salida)*. Cada IOB contiene una lógica compleja que permite que un pin pueda actuar como entrada, salida o un tercer estado.

La **figura 5.2** muestra a detalle, la arquitectura del XC4003E de Xilinx. Nótese la complejidad de un CLB, así como la disposición de los IOBs alrededor del dispositivo, sirviendo como interfases entre el FPGA y el mundo exterior

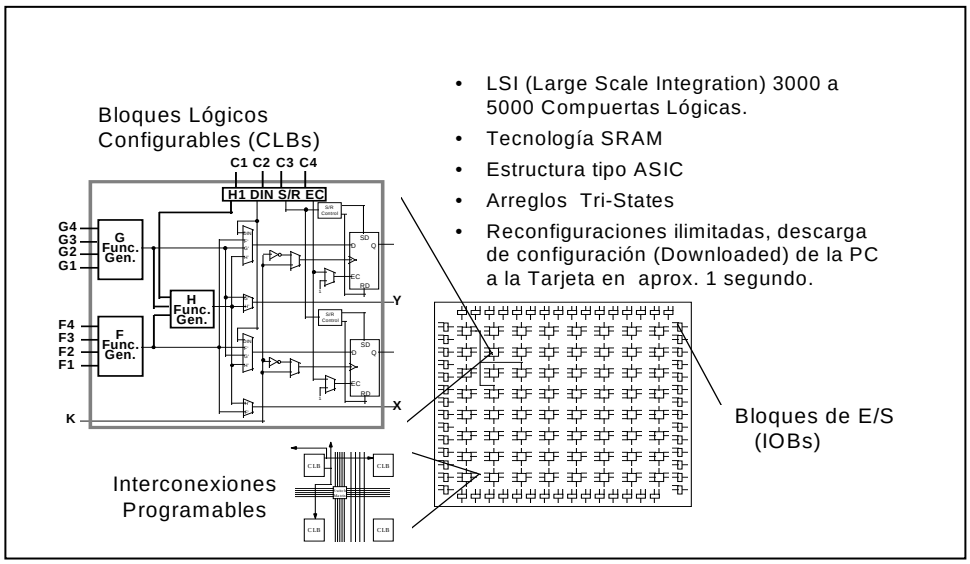

Figura 5.2. –Arquitectura de un FPGA XC4003E de Xilinx.

La serie *XC Virtex*, o tan solo *Virtex*, es la más nueva de todas las propias de Xilinx. Se dice que los dispositivos pertenecientes a esta familia son los más rápidos (velocidades de trabajo de hasta 250 Mhz), densos en compuertas y menor consumo de potencia, pero por lo mismo son los más costosos.

Apropiados para diseños muy grandes, complejos y de alto desempeño, el FPGA *XCV3200E*, de la familia Virtex (en específico, de la serie *Virtex E*) es el dispositivo más grande y poderoso de este fabricante, con cerca de 4,047,000 de compuertas lógicas (4 millones).

La serie *Spartan* surgió como una opción para sustituir diseños probados de menos de 15,000 compuertas por dispositivos de bajo costo y alto desempeño (además incluyen el soporte de los *CORES* prediseñados), pero sacrificando algunas características que manejan las series estándar de Xilinx, como la XC4000 tradicional o la serie Virtex. Las series estándar *XC3000* y su sucesora, la *4000*, son series que muestran la mayor parte de las características funcionales de los FPGAs de Xilinx, pero con la gran desventaja que son dispositivos de baja densidad de compuertas. Sin embargo, el uso de la serie XC4000 (en especial el XC40003E) es muy favorecido para el diseño e implementación de prototipos de bajo impacto[6].

La **tabla 5.1**, ilustra una breve comparación entre familias de Xilinx, tomando como parámetros de comparación a la densidad de compuertas lógicas y a los números de pines configurables como entradas o salidas.

Familia	Pines de E/S	Número de Compuertas Lógicas
Viirtex E	176 A 804	470,000 a 4,047,000
Virtex	180 a 512	34,000 a 1,124,000
Spartan y Spartan II	77 a 260	2,000 a 150,000
XC 4000	56 a 448	10,000 a 180,000

Tabla 5.1. – Comparación de densidad entre las diferentes Familias de FPGAs de Xilinx.

5.4.2 FPGAs de Altera.

Altera ofrece dos familias de FPGAs con características diferentes, pero conservando algunas básicas que representan las ventajas originales de las primeras

[6] *Un diseño de bajo impacto, es un circuito digital simple, que no llega a ser un sistema completo. Recordemos que un diseño "pequeño" en un FPGA será "grande" en comparación a uno diseñado sobre un PLD o algunos CPLDs de poca capacidad.*

familias estándar: *FLEX 6000, 8000*, y *10K*; así como la más novedosa, *APEX 20K*. Las primeras familias estándar, la FLEX 6000 y la 8000, aún se utilizan ampliamente. La serie FLEX estándar contiene un número considerado de compuertas en tecnología SRAM con tablas de búsqueda, agregando mayor flexibilidad a los diseños (*FLEX: Flexible Logic Element Matrix – Matriz Flexible de Elementos Lógicos*).

A continuación, en la **tabla 5.2**, se listan las diferencias en densidad de las diferentes familias. Nótese que la serie APEX 20K es la que contiene el mayor número de compuertas y es la que se usa para diseños más complejos y dedicados.

Familia	Pines de E/S	Número de Compuertas Lógicas
APEX 20K	250 a 780	263,000 a 2,670,000
FLEX 10K	59 a 470	10,000 a 250,000
FLEX 8000	71 a 218	16,000 a 24,000
FLEX 6000	68 a 208	2,500 a 16,000

Tabla 5.2. – Comparación de densidad entre las diferentes Familias de FPGAs de Altera.

La serie estándar FLEX combina la arquitectura de los CPLDs con los FPGAs. El dispositivo consiste de una arquitectura muy parecida a la de un CPLD, en la que el nivel más bajo de la jerarquía es un conjunto de *Tablas de Búsqueda*, en lugar de un bloque muy similar a un SPLD, por lo mismo se considera un FPGA. Por ejemplo, el FLEX 8000 está basado en tecnología de programación SRAM, teniendo una tabla de búsqueda de 4 entradas en su módulo lógico más básico. La **figura 5.3**, ilustra la arquitectura general del FLEX 8000. El módulo lógico básico, nombrado por Altera, *Elemento Lógico (Logic Element*), contiene la LUT de 4 entradas, un flip – flop y un elemento de acarreo (carry) de propósito especial para circuitos aritméticos (similar al XC4000 de Xilinx). El Elemento Lógico también incluye circuitos en cascada que permiten una implementación eficiente de funciones AND amplias.

Figura 5.3. –Arquitectura de un FPGA FLEX 8000 de Altera.

Esta arquitectura agrupa Elementos Lógicos en grupos de 8, y los llama *Bloques de Arreglos Lógicos (Arrays Logic Blocks – ALBs*). Cada ALB, contiene una interconexión local que le permite conectarse con otro ALB, a la vez, la misma interconexión sirve para conectarse a la interconexión global de la crossbar[7] (matriz de interconexiones), nombrada por Altera como *FastTrack*. Así, las interconexiones se hacen al estilo de los CPLDs, pero la configuración de los Bloques de Arreglos Lógicos utilizan tecnología SRAM propia de los FPGAs.

4.4.3 FPGAs de Actel.

Actel ofrece una serie de familias OTP que resultan ampliamente utilizadas después de haber probado satisfactoriamente un diseño (*emigrar* a otro FPGA). Las principales son: La serie estándar *ACT*, y las más nuevas por orden cronológico de aparición, *sX, sX – A*, *mX* y la más reciente, *eX*. Todas las anteriores son programables fuera del sistema (*OS*). También ofrece una familia reprogramable a la que llama *Pro ASIC* (es de alta densidad de componentes, y Actel no la considera parte de los FPGAs), basada en una tecnología *Flash EEPROM* programable en sistema (*IS*).

Los FPGAs de Actel, emplean como módulo o elemento básico una estructura tipo *Arreglo Fijo de Compuertas*. La **figura 5.4**, muestra un dispositivo *ACT 3*, donde podemos apreciar como la lógica del arreglo está dispuesta en renglones de módulos lógicos interconectables, rodeados hacia fuera de la ilustración por *Módulos de E/S*

[7] *La crossbar de un FPGA de Altera está conformada por varias otras, por lo que resulta muy avanzada, incluyendo mayores ventajas que la de un CPLD típico.*

(Input/Output Modules). La estructura de interconexiones consiste en pistas o líneas fijas de interconexión horizontales y verticales con los segmentos de alambrado. Hay muchas pistas en cada canal entre los renglones de la lógica. Las pistas verticales son menos y pasan sobre los canales horizontales y los módulos lógicos.

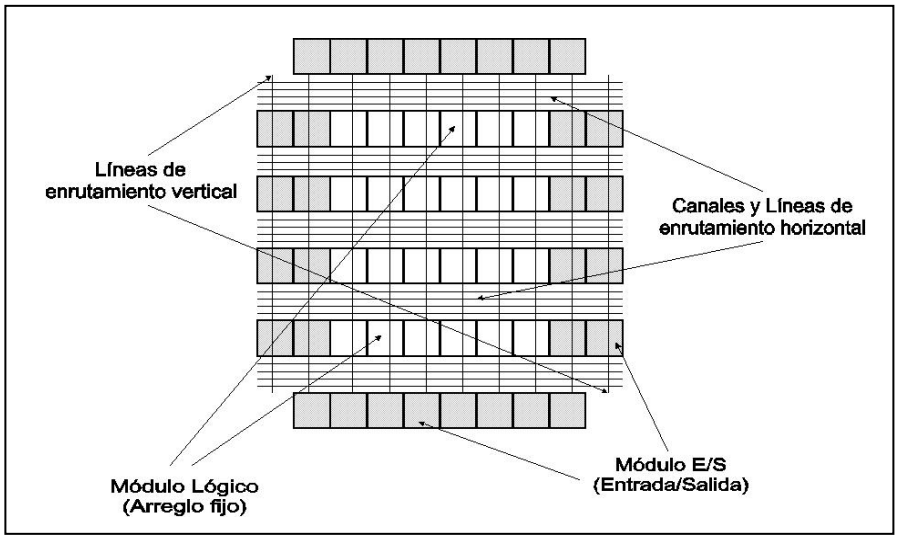

Figura 5.2. –Arquitectura de un FPGA ACT 3 de Actel.

El FPGA de Actel utiliza tecnología *antifusible* que proporciona una programación permanente y no volátil. El dispositivo tiene muchos antifusibles para conectar las entradas y salidas de los módulos de lógica y E/S a los segmentos de alambrado de los canales. También tiene antifusibles que interconectan los segmentos de alambrado entre las pistas para ofrecer conexiones de diferentes longitudes.

Una de las principales características de los módulos lógicos de los FPGAs de Actel, es que los módulos no se programan[8] para que efectúen una operación, sino que toda la programación se hace mediante antifusibles en las pistas de alambrado.

Dentro de las series antifusibles, la familia *eX* es la tiene el mejor desempeño, con las características más novedosas, pero en contraparte es la de menor densidad de compuertas. La serie *mX* es la que entrega el menor retardo de propagación entre todas las familias y fabricantes de FPGAs, combinando la tecnología antifusibles con las interconexiones de alta rapidez para alcanzar velocidades de trabajo de hasta 450 Mhz. La serie *Pro ASIC* de Actel con tecnología reprogramable EEPROM es la de mayor

[8] *Recordemos que en un CLB de Xilinx, la función lógica a implementar se programa configurando las tablas (LUTs). Mientras que en un Módulo de Actel, la función es fija, por lo que se dice que son arreglo de lógica o función fija.*

densidad de componentes, pero ostenta el menor desempeño en comparación a los dispositivos de las otras familias.

La **tabla 5.3** ilustra la comparación entre familias de Actel en relación al número de compuertas.

Familia	Pines de E/S	Número de Compuertas Lógicas
EX	84 a 132	3,000 a 12,000
sX y sX – A	130 a 360	12,000 a 108,000
MX	57 a 202	3,000 a 54,000
Pro ASIC	210 A 446	98,000 a 473,000
ACT 1, 2, y 3	56 a 244	2,000 a 25,000

Tabla 5.3. – Comparación de densidad entre las diferentes Familias de FPGAs de Actel.

Juan Carlos Gonzalez Robles jgrobles@ipn.mx

Metodología de Diseño VLSI con Herramientas CAD

6.1 Metodología de Diseño sobre ASICs Programables.

El elemento fundamental en cualquier metodología actual en el diseño de circuitos, tanto digitales como analógicos, es la especificación y forma en la que se describe el diseño. La captura clásica de diagramas esquemáticos resulta obsoleta si se pretende estimar desarrollos complejos con miles de elementos, haciéndose necesaria la participación de un lenguaje dedicado a la especificación del hardware, describiendo de manera abstracta el funcionamiento de cada elemento físico involucrado.

Como tal, el uso de un lenguaje formal implica conocer las diferentes apreciaciones en la programación lineal y paralela de procesos específicos, no sólo como parte del código de un programa, sino en el algoritmo mismo que por su naturaleza como pieza de hardware representa arquitecturas seriales o paralelas[9], con la intención de optimizar resultados aprovechando los recursos disponibles.

Sin la evolución en las metodologías de diseño, concernientes a las herramientas *VLSI CAD (Diseño Asistido por Computadora para dispositivos VLSI*), sería imposible integrar y atañer diseños con la complejidad que los dispositivos actuales lo permiten. La implementación de circuitos sobre *ASICs programables*, fundamenta el estudio de los dispositivos VLSI a nivel *sustrato* y como tecnología de aplicación para la construcción de diseños personalizados[10].

El diseño sobre dispositivos VLSI programables, particularmente digitales[11], tiene sus inicios desde hace más de una década. El conjunto *EDA (Electronic Desig Automation – Diseño Electrónico Automatizado*) son todas las herramientas, tanto hardware como software, que se utilizan para el diseño de sistemas electrónicos. Dentro de EDA, las herramientas *VLSI CAD* juegan un importante papel en el diseño de hardware a través de software. En virtud del inminente incremento en la complejidad de los circuitos VLSI, se hace indispensable un sofisticado aporte por parte de las herramientas CAD para automatizar el proceso de desarrollo, repercutiendo en una *disminución en el tiempo*

[9] *El estudio de las arquitecturas en hardware, inició tiempo atrás en comparación con el uso del software para modelar el funcionamiento de las mismas.*
[10] *El campo de desarrollo de dispositivos VLSI se divide en dos partes: la primera es el diseño de estos dispositivos a nivel sustrato, es decir, la construcción del circuito integrado como tal. La otra parte hace referencia a la utilización de los dispositivos comercialmente existentes para aplicaciones de usuario, que es el enfoque que se les da en este trabajo.*
[11] *Actualmente es posible diseñar sobre dispositivos digitales (FPGAs, CPLDs) o sobre analógicos (FPAAs,- Field Programmable Analog Array – Arreglo Analógico Programable en Campo); de cualquier forma, el flujo de diseño es muy similar respetando algunas restricciones en cuanto a aplicación y topología del dispositivo.*

de diseño, aumentando la calidad del producto y *reduciendo los costos de producción*.

Hoy en día, existen variados ambientes de desarrollo para ASICs programables (FPGAs y CPLDs, reconfigurables y no reconfigurables), algunos con herramientas de software completas para aceptar la *captura, simulación, síntesis y configuración física* del dispositivo; a la vez que otros son más modestos en sus capacidades; o en otros casos se adaptan a las herramientas de propietario disponibles en el mercado actual. Entre los diferentes fabricantes se respetan estándares de diseño con *Lenguajes de Descripción de Hardware* (*HDLs*, posteriormente se comentará formalmente al respecto, en el apartado **6.3.1.2**) pero no así de captura de esquemáticos y mucho menos de configuración física de dispositivos, debido a que los ambientes y arquitecturas de propietario, son distintas entre sí a causa de la manufactura y de la creciente evolución de las nuevas familias de ASICs programables que los diferencian.

Los ambientes completos *Foundation Series, MAX+PLUS II* y *Actel DeskTOP*, representan los más utilizados actualmente para el diseño e implementación de aplicaciones con lógica programable (**Unidad 7**). Para la elaboración de estos apuntes nos adecuamos al uso de estos tres ambientes y específicamente a la implementación sobre FPGAs como principal arquitectura programable. Lo concerniente a los CPLDs no es materia de discusión en este trabajo, aunque no se desconoce la familiaridad entre ambas tecnologías, por lo que resulta similar el tratamiento en el diseño respetando las restricciones consideradas para cada arquitectura.

La **figura 6.1** esquematiza de manera general, la secuencia de fases necesarias para diseñar sobre ASICs programables (en especial, sobre FPGAs). Obsérvese la prioridad que precede cada uno de los pasos dentro del flujo de diseño VLSI, tomando en consideración que dependiendo del fabricante, el nombre de la fase puede variar manteniendo su operatividad básica de acuerdo al caso de estudio presentado.

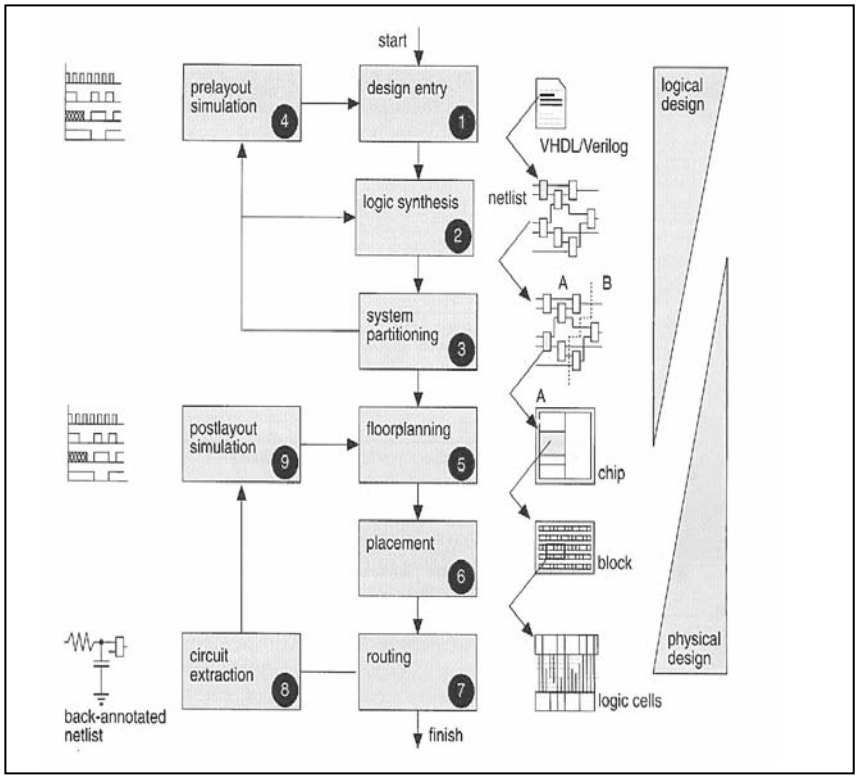

Figura 6.1. – Flujo de desarrollo para ASICs programables, utilizando herramientas VLSI CAD.

1. – *Captura del diseño (Design Entry).* En este paso se procede a la captura del modelo dentro del ambiente de desarrollo VLSI CAD elegido por el diseñador. La captura puede ser mediante uno (es válido el uso de varios) de los siguientes medios que describen el funcionamiento del circuito: *diagrama esquemático,* mediante un *Lenguaje de Descripción de Hardware (HDL – Hardware Description Language)*, o mediante un *Netlist (Archivo de Conexiones)*. En el caso de la captura de esquemático, el software de diseño cuenta con bibliotecas muy extensas de componentes de uso común, por lo que sólo basta con realizar interconexiones entre las *primitivas*[12] formando el sistema digital (referirse a la **figura 6.2**).

[12] *Una primitiva es una celda lógica básica, o bien un circuito muy simple formado con elementos muy básicos; por ejemplo, una compuerta lógica. Debido a su naturaleza, la primitiva es el elemento que se encuentra en el nivel más bajo de la jerarquía de diseño en el VLSI CAD.*

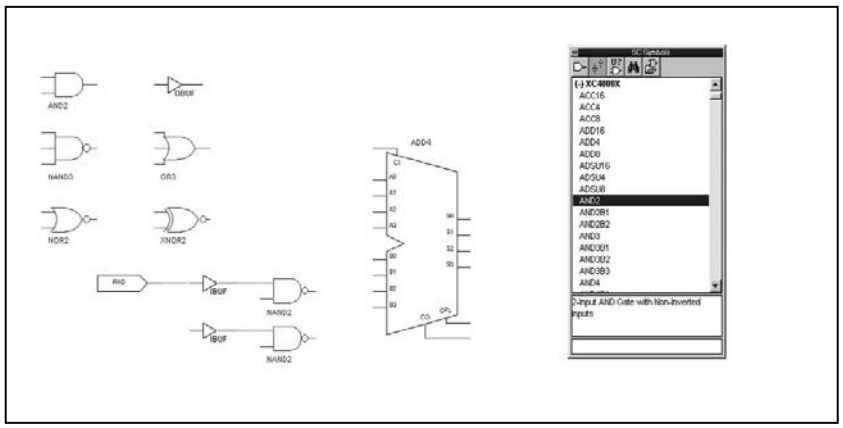

Figura 5.6. – Editor de diagramas esquemáticos.

El método clásico para la interconexión de los distintos símbolos de una hoja de diagrama son los *cables* o *wires*. Un cable tiene una correspondencia inmediata con el circuito real, se trata de una conexión física que une una terminal con otra, creando una correspondencia para la transmisión de la señal eléctrica. Debido a la cantidad de cables que puede presentar un diseño complejo, el editor permite la agrupación en *buses*. La conexión hacia el exterior del circuito integrado se conoce como *puerto*, no obstante la mayoría de los editores hace la referencia conforme a la dirección de su señal: *IPAD (puerto de entrada*), *OPAD (Puerto de Salida*), agregando el pin bidireccional *IOPAD (Puerto de Entrada/Salida*)

Una captura por medio de un HDL también culmina con *símbolos* que se integran es un diagrama esquemático. Algunos ambientes, en especial los más sofisticados, permiten editar código introduciendo la descripción del funcionamiento del circuito por medio de: Tablas de Estado, Diagramas de Estado (con herramienta gráfica o código directo), y Ecuaciones; siguiendo una sintaxis natural. Los editores que incorporan los diferentes ambientes VLSI CAD admiten código HDL y posteriormente crean una *Macro* reutilizable, que es un *módulo esquemático* (simbólico) que representa un componente cuya función es específica. La biblioteca de la macro se adiciona automáticamente a las bibliotecas del software para quedar disponible en el entorno de la captura esquemática, pudiéndose utilizar como elemento independiente o combinado con las primitivas originales, como se puede observar en la **figura 6.3**.

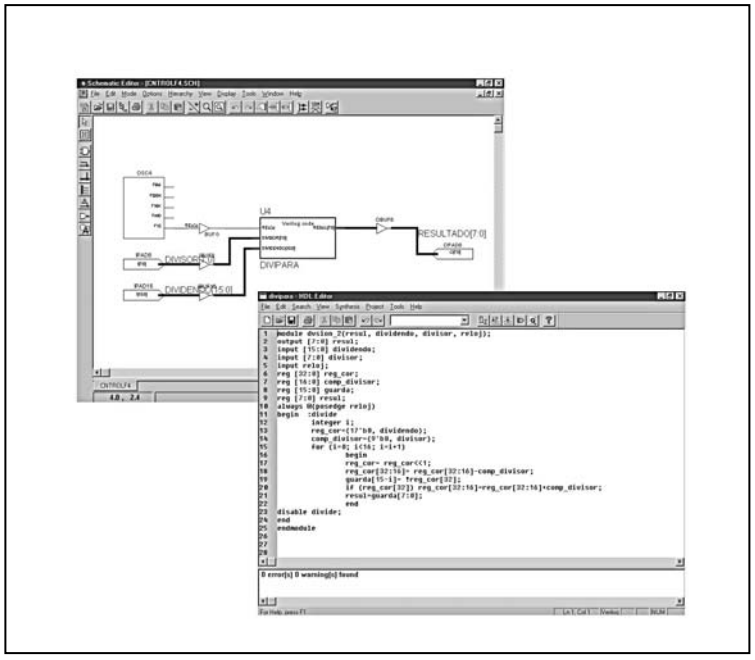

Figura 6.3. – Editor de HDL y diagrama de conexiones de una macro creada.

Un diseño mediante HDL, inicia con el planteamiento tradicional de método de descripción. El proceso de diseño VLSI puede verse como una secuencia de transformaciones sobre representaciones físicas del modelo a partir de una descripción del *comportamiento (behavioral*) o una descripción *estructural (structural*) del mismo.

La *descripción del comportamiento* representa el funcionamiento de un circuito en relación a sus entradas y salidas, esto es, se modela la operación integral del circuito desde un panorama muy exterior. En este sentido, la *representación estructural* describe la conformación del circuito en términos de los componentes y las interconexiones entre ellos. Cada una de las descripciones tiene connotaciones más formales, por ejemplo a la descripción de comportamiento se le llama también *RTL*[13] *(Register Transfer Level –Transferencia a Nivel de Registros*), mientras que a la estructural se le conoce como *GTL (Gate Transfer Level – Transferencia a Nivel de Compuertas Lógicas*).

[13] *La descripción de Comportamiento o RTL, tiene además, otra forma de nombrarse: High Level Modules (Módulos de Nivel Alto), y hacen referencia a que son módulos que describen su funcionamiento desde el nivel más alto de abstracción, tal y como lo comentaremos en el apartado* ***5.2****.*

La mayoría de los diseños complejos (por ejemplo, un microprocesador) se describen a nivel RTL, en consecuencia a que es más simple describir un comportamiento que realizar un diseño componente a componente. La desventaja es obvia, a nivel compuerta perdemos de cierta forma el control de las acciones, suponiendo solamente el funcionamiento sin reparar en cómo se está llevando a cabo y por quién. Hace algunos años era un verdadero inconveniente; sin embargo, la evolución y sofisticación de las herramientas VLSI CAD, soporta que ahora sean programadas con algoritmos inteligentes que compilan de la mejor manera un diseño, dejando a consideración del diseñador si se desea un circuito con restricciones de *velocidad de procesamiento* o con restricciones de *espacio físico*.

2. – *Síntesis Lógica (Logic Síntesis*). El software de desarrollo incluye la herramienta propia para realizar la síntesis lógica de un código en HDL o de un diagrama esquemático. A través de una compilación es posible producir un *Netlist* a partir de cualquiera de los métodos de captura. Un *Netlist* es un archivo que registra la descripción de las *celdas lógicas* y sus conexiones específicas, que conforman un circuito.

3. – *Partición del Sistema (System Partitioning*). Un sistema completo no necesariamente grande, se divide y distribuye en determinadas piezas dentro del ASIC programable. Lo anterior tiene la finalidad de realizar una partición física del dispositivo para facilitar la labor del mapeo colocando estratégicamente las celdas lógicas de acuerdo a su respectiva correspondencia en un diseño.

4. – *Simulación Pretrazado (Prelayout Simulation*). Hasta el tercer paso, aún no se ha comenzado con el *diseño físico*, solamente se ha realizado el *diseño lógico* del circuito modelado. La mayoría de los ambientes de desarrollo incluyen dos tipos de simulación, una llamada *lógica* y la otra *física*. La diferencia es común, la primera es antes de dirigir el diseño hacia una *tecnología particular (dispositivo físico*) y la otra es posterior a dirigirlo.

5. – *Planeación de la Superficie (Floorplanning*). Una vez que se simuló el diseño, se procede a mapear el netlist sobre el ASIC programable seleccionado. La configuración de las celdas lógicas y sus respectivas conexiones, descritas por el netlist, se distribuyen sobre la superficie del circuito integrado a manera de identificar recursos.

6. – *Colocación (Placement*)[14]. Estratégicamente, el software decide la colocación de las celdas sobre un bloque del dispositivo físico. Los algoritmos inteligentes con los que trabaja el sistema de compilación, deciden la mejor ubicación para cada celda lógica

[14] *Las tareas de Place y Rute, propias de los FPGAs, son conocidas como el Fitter (Ajuste) en el diseño sobre CPLDs.*

respetando ciertas consideraciones como las líneas de retardo o las *redundancias* en el diseño *(componentes o conexiones repetidas)*.

7. – Ruteo (Rutting). Realiza la conexión entre las celdas y los bloques que conforman los arreglos lógicos.

8. – *Extracción (Extraction*). Determina la *resistencia* y *capacitancia* eléctrica, entre las interconexiones para verificar un correcto ruteo de las líneas. El software de diseño se encarga automáticamente de generar el ruteo y la extracción de impedancias, sin embargo, algunas herramientas incluyen editores para realizar las conexiones de modo personalizado.

9. – *Simulación Postrazado (Postlayout Simulation*). Una vez que se han colocado y ruteado las celdas lógicas, la configuración física que ha adquirido el dispositivo puede simularse. A este tipo de simulación se le conoce como *física*, debido a su proximidad con el comportamiento real que tendrá el diseño.

6.2 Niveles de Abstracción.

Independientemente de la descripción abstracta global del diseño (*Comportamiento o Estructural*), es necesario definir el método que se seguirá para jerarquizar los diferentes niveles de concepción con los que se trabajará. Principalmente, los diseñadores definen dos métodos para jerarquizar los niveles de desarrollo en función de una operatividad por bloques, donde cada bloque realiza de manera independiente una tarea, dividiendo el proceso en módulos.

1. *- Diseño de Abajo hacia Arriba(Bottom –Up).*

Se trata de un método *ascendente* mediante el cual se realiza la descripción del circuito a modelar, empezando por describir los componentes más básicos del sistema (primitivas) para posteriormente agruparlos en diferentes módulos, y éstos a su vez en otros módulos hasta llegar a uno solo que representa el sistema completo. Una aproximación a este método se representa en el diagrama a bloques de la **figura 6.4a**.

2. *- Diseño de Arriba hacia Abajo (Top – Down).*

Este método parte de una idea en un alto nivel de abstracción y después proseguir la descripción hacia abajo, incrementando el nivel de detalle según sea necesario. En otras palabras, el circuito inicial se divide en diferentes módulos, cada uno de los cuales se encuentra a su vez subdividido hasta llegar a los elementos primarios de la

descripción, siguiendo un flujo *descendente* (**figura 6.4b**). No necesariamente se debe alcanzar un nivel de primitivas, ya que un planteamiento correcto no lo permite.

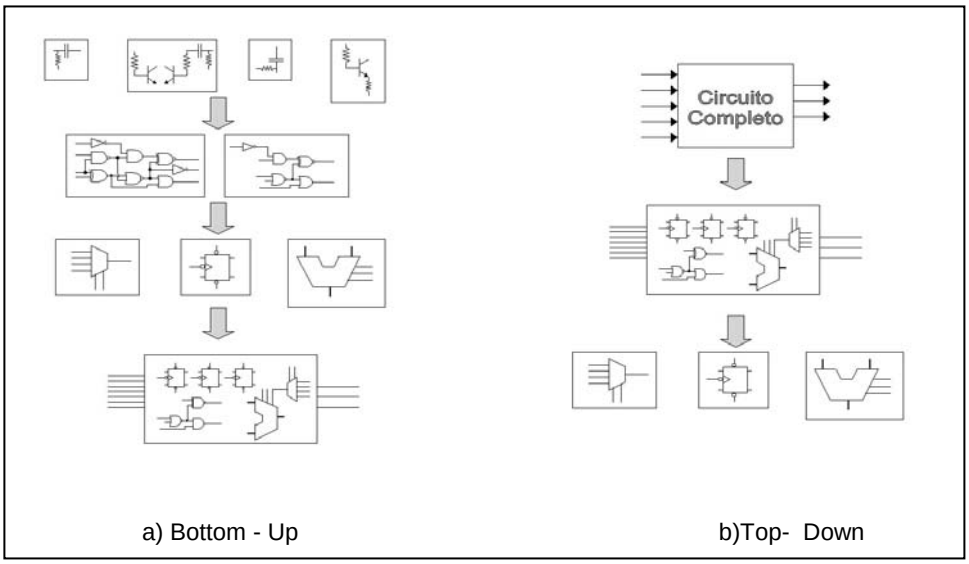

Figura 5.4. – Abstracciones en el diseño, de acuerdo planteamiento.

6.2.1 Caso de Estudio: Comparación entre Métodos de Jerarquización.

Una de las principales comparaciones entre ambos métodos es la complejidad que representa llevar a cabo las conexiones entre las primitivas implicadas. Para diseños grandes, no es factible conectar miles de componentes a bajo nivel y pretender que el diseño completo funcione adecuadamente, por lo tanto el flujo *Bottom – Up* no es recomendable. El hecho de unir un número elevado de componentes básicos entre sí, sin una estructura que permita separarlos en bloques, hace que sea más complejo el análisis del circuito, con esto aumenta la probabilidad de cometer errores debido a que no resulta fácil detectarlos.

El método *Top – Down* es ampliamente utilizado en la actualidad, ya que permite dividir un circuito grande en otros circuitos más pequeños derivados del mismo, lo que permite tratar de manera más personal un módulo sin llegar a un nivel de abstracción muy bajo. Expresamente, este tipo de jerarquización presenta tres ventajas considerables:

a) *Incrementa la productividad del diseño*. Al especificar un diseño en HDL, el software de desarrollo generará automáticamente el nivel correspondiente de compuertas lógicas, por lo que el tiempo utilizado en un diseño disminuye radicalmente.

b) *Incrementa la reutilización del diseño*. En el proceso de diseño VLSI se utilizan *tecnologías genéricas (Circuitos Integrados Genéricos*), esto es que la tecnología a utilizar no se fija sino hasta llegar al *diseño físico (physical design*, ver **figura 5.1***)*, permitiendo reutilizar los datos del diseño únicamente cambiando la tecnología de implementación. Así es posible crear un nuevo diseño a partir de uno ya existente.

c) *Rápida detección y predicción de errores*. Como es necesario un claro análisis en la definición de la descripción del diseño, es posible detectar y predecir errores desde el momento de modularizar.