I LENGUAJES DE DESCRIPCIÓN DE HARDWARE

1.1 INTRODUCCIÓN

A mediados de los años setenta se produce una fuerte evolución en los procesos de fabricación de los circuitos integrados, promoviendo el desarrollo de circuitos digitales hasta la primera mitad de los años ochenta. En aquellas épocas el esfuerzo de diseño se concentraba en los niveles eléctricos para establecer características e interconexiones entre los componentes básicos al nivel de transistor. El proceso de diseño era altamente manual y tan sólo se empleaban herramientas como PSPICE para simular esquemas eléctricos con modelos previamente caracterizados a cada una de las distintas tecnologías. A medida que pasaban los años, los procesos tecnológicos se hacían más y más complejos. Los problemas de integración iban en aumento y los diseños eran cada vez más difíciles de depurar y de dar mantenimiento. Inicialmente los circuitos MSI (*Medium Scale Integration*) y LSI (*Low Scale Integration*) se diseñaron mediante el desarrollo de prototipos basados en módulos simples. Cada uno de estos módulos estaba formado por compuertas ya probadas, pero este método poco a poco iba quedándose obsoleto conforme aumentaba la complejidad y tamaño de los circuitos. A finales de los años setenta se constata el enorme desfase que existía entre tecnología y diseño.

La considerable dificultad que puede llegar a tomar el fabricar un circuito de alta escala de integración, involucra riesgos y costos de diseño desmesurados e imposibles de asumir por las empresas. Es entonces, cuando diversos grupos de investigadores empiezan a crear y desarrollar los llamados "lenguajes de descripción de hardware", lenguajes en los que no fuera necesario caracterizar eléctricamente cada componente del circuito al nivel de transistor para así enfocarse solamente en el funcionamiento lógico del sistema. Empresas tales como IBM con su IDL, el TI-HDL de Texas Instruments, ZEUS de General Electric, etc., así como los primeros prototipos empleados en las universidades, empezaron a desarrollarse buscando una solución a los problemas que presentaba el diseño de sistemas complejos. Sin embargo, estos lenguajes nunca alcanzaron el nivel de difusión y consolidación necesarios por motivos distintos. Unos, los industriales, por ser propiedad de la empresa permanecieron encerrados en ellas y no estuvieron disponibles para su estandarización y mayor difusión, en tanto que los universitarios perecieron por no disponer de soporte ni mantenimiento adecuado.

1.2 EL CONCEPTO DE HERRAMIENTAS CAD-EDA

CAD son las siglas de **C**omputer **A**ided **D**esign, o diseño asistido por computadora el cual constituye todo un proceso de trabajo utilizando técnicas de análisis apoyadas en gráficos mediante sofisticadas herramientas de software las cuales facilitan el estudio de los problemas asociados con el diseño en cuestión. El concepto CAD se relaciona con el dibujo como parte importante en el proceso de diseño pero, además, el diseño de un circuito debe cumplir con los requerimientos especificados por el equipo de diseño, por las normas de calidad existentes, los costos, etc. por lo que las herramientas CAD intervienen en todas las fases del diseño. Ya que no sólo son importantes por acelerar el desarrollo del sistema al permitir que varias personas puedan trabajar simultáneamente en distintas etapas del diseño sino que, además, es posible verificar el funcionamiento del circuito mediante la simulación del sistema. Todo esto simplifica la tarea del equipo de diseño y conduce a la conclusión del prototipo en menos tiempo. EDA, **E**lectronic **D**esign **A**utomation, es el nombre que se le da a todas las herramientas de hardware y software en el diseño de sistemas electrónicos. Porque no sólo el software es importante, también lo es el hecho de que las computadoras cada día son más veloces y de mayor capacidad de procesamiento, lo cual influye en el proceso de diseño de circuitos electrónicos.

El diseño de hardware tiene un problema fundamental, que no existe en desarrollo de software. Este problema es el alto costo del ciclo *diseño* → *desarrollo del prototipo* → *pruebas* → *reinicio del ciclo*, ya que el costo del prototipo generalmente suele ser bastante elevado. Se impone la necesidad de reducir este ciclo de diseño para no incluir la fase de desarrollo del prototipo más que al final del proceso, evitando la repetición de varios prototipos que es lo que encarece el ciclo. Para ello se introduce la fase de simulación y verificación de circuitos utilizando herramientas EDA, de tal forma que no sea necesario implementar físicamente un prototipo para comprobar el funcionamiento del circuito.

En el ciclo de diseño de circuitos, las herramientas EDA están presentes en todas las fases. Primero en la fase de generación del sistema que puede representarse mediante un diagrama esquemático, a bloques o de flujo.

Se encuentran también en la fase de simulación y comprobación de circuitos, donde diferentes herramientas permiten verificar el funcionamiento del sistema. Estas simulaciones pueden ser de eventos, funcionales, digitales o eléctricas, de acuerdo al nivel de simulación requerido. Después están las herramientas EDA utilizadas en la síntesis y programación de circuitos digitales en dispositivos lógicos programables. Existen, además, las herramientas EDA orientadas a la fabricación de circuitos. En el caso del diseño de hardware estas herramientas sirven para la realización de PCBs (Printed Circuit Boards o placas de circuito impreso), o para desarrollar circuitos integrados de aplicación especifica conocidos como ASICs (Application Specific Integrated Circuits). Este ciclo de diseño de hardware se muestra en la figura 1.1.

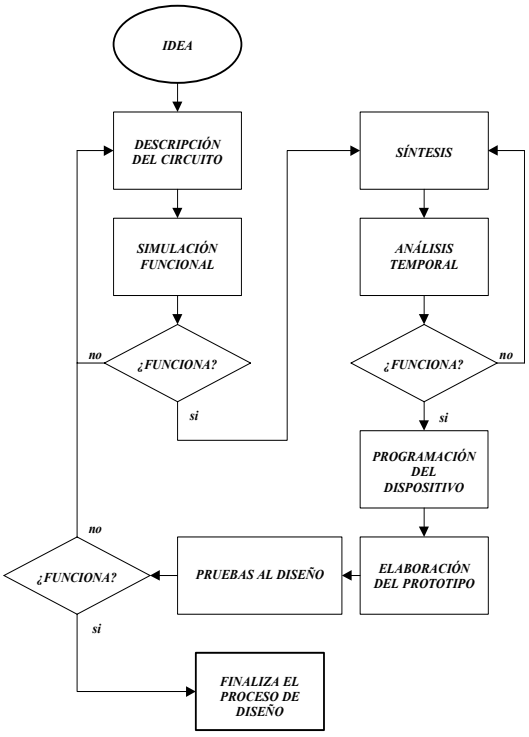

Figura 1.1 Flujo de diseño en el desarrollo de sistemas electrónicos

A continuación se mencionan las principales características y finalidad de algunas herramientas EDA que intervienen en el diseño de circuitos.

I. Lenguajes de descripción de circuitos.

Son lenguajes mediante los cuales es posible describir el funcionamiento y estructura de un circuito eléctrico o digital. La descripción puede ser mediante bloques donde se muestra la arquitectura del diseño, o de comportamiento, es decir, se describe el funcionamiento del circuito en vez de especificar los elementos de los que está compuesto.

II. Diagramas esquemáticos.

Es la forma clásica de describir un diseño electrónico y la más extendida ya que era la única usada antes de la aparición de las herramientas de EDA. La descripción está basada en un "plano" donde se muestran los diferentes componentes utilizados en el circuito.

III. Grafos y diagramas de flujo.

Es posible describir un circuito o sistema mediante diagramas de flujo, redes de Petri, máquinas de estados, etc. En este caso sería una descripción *gráfica* y además sería comportamental, porque no es una descripción mediante componentes.

IV. Simulación de eventos.

Estas herramientas se usan para la simulación de circuitos a grandes rasgos. En esta simulación, los componentes más importantes son elementos de alto nivel como discos duros, buses de comunicaciones, memorias RAM etc.

V. Simulación funcional.

Bajando al nivel de compuertas digitales se puede realizar una simulación funcional de las mismas. Este tipo de simulación comprueba la operación de circuitos digitales a partir del comportamiento lógico de sus elementos con el fin de comprobar el funcionamiento en conjunto del circuito mediante unos estímulos dados. Similar a lo que se realiza en un laboratorio.

VI. Simulación digital.

Esta simulación, también exclusiva de los circuitos digitales, es como la anterior con la diferencia de que se tienen en cuenta los retardos

de propagación de cada compuerta. Es una simulación muy cercana al comportamiento real del circuito y prácticamente garantiza el funcionamiento correcto del circuito en cuestión. En las herramientas EDA este tipo de simulación se conoce como análisis temporal o **timing**.

VII. Simulación eléctrica.

Es la simulación de más bajo nivel donde las respuestas del sistema se verifican al nivel de transistor. Sirven tanto para circuitos analógicos como digitales y su respuesta es prácticamente idéntica a la realidad ya que se prueban retardos de tiempo, niveles de voltaje, disipación de potencia, etc.

VIII. Diseño de PCBs

Con estas herramientas es posible realizar el trazado de pistas para la fabricación de placas de circuitos impresos.

IX. Diseño de circuitos integrados

Son herramientas EDA que sirven para la realización de circuitos integrados. Las capacidades gráficas de estas herramientas permiten la realización de las diferentes máscaras que intervienen en la realización de éstos.

X. Diseño con dispositivos programables.

Estas herramientas facilitan la programación de dispositivos, ya sean PALs, PLDs, CPLDs o FPGAs.

1.3 LENGUAJES DE DESCRIPCIÓN DE HARDWARE

Los lenguajes de descripción de hardware (HDLs) son utilizados para describir la arquitectura y comportamiento de un sistema electrónico los cuales fueron desarrollados para trabajar con diseños complejos.

Comparando un HDL con los lenguajes para el desarrollo de software vemos que en un lenguaje de este tipo un programa que se encuentra en un lenguaje de alto nivel (VHDL) necesita ser ensamblado a código máquina (compuertas y conexiones) para poder ser interpretado por el procesador. De igual manera, el objetivo de un HDL es describir un circuito mediante un conjunto de instrucciones de alto nivel de abstracción para que el programa de síntesis genere (ensamble) un circuito que pueda ser implementado físicamente.

La forma más común de describir un circuito es mediante la utilización de esquemas que son una representación gráfica de lo que se pretende realizar. Con la aparición de herramientas EDA cada vez más complejas, que integran en el mismo marco de trabajo las herramientas de descripción, síntesis, simulación y realización; apareció la necesidad de disponer de un método de descripción de circuitos que permitiera el intercambio de información entre las diferentes herramientas que componen el ciclo de diseño. En principio se utilizó un lenguaje de descripción que permitía, mediante sentencias simples, describir completamente un circuito. A estos lenguajes se les llamó ***Netlist*** puesto que eran simplemente eso, un conjunto de instrucciones que indicaban las interconexiones entre los componentes de un diseño. A partir de estos lenguajes simples, que ya eran auténticos lenguajes de descripción hardware, se descubrió el interés que podría tener el describir circuitos utilizando un lenguaje en vez de usar esquemas. Sin embargo, se siguieron utilizando esquemas puesto que desde el punto de vista del ser humano son mucho más sencillos de entender, aunque un lenguaje siempre permite una edición más rápida y sencilla.

Conforme las herramientas de diseño se volvieron más sofisticadas, y la posibilidad de desarrollar circuitos digitales mediante dispositivos programables era más viable, apareció la necesidad de poder describir los circuitos mediante un lenguaje de alto nivel de abstracción. No desde un punto de vista estructural, sino desde el punto de vista funcional. Este nivel de abstracción se había alcanzado ya con las herramientas de simulación, ya que para poder simular partes de un sistema era necesario disponer de modelos que describieran el funcionamiento de bloques del circuito o de cada componente si fuera necesario. Estos lenguajes estaban sobre todo orientados a la simulación, por lo que poco importaba que el nivel de abstracción fuera tan alto que no fuera sencillo una realización o síntesis a partir de dicho modelo. Con la aparición de técnicas para la síntesis de circuitos a partir de lenguajes de alto nivel de abstracción, se comenzaron a utilizar los lenguajes de simulación para sintetizar circuitos. Que si bien alcanzan un altísimo nivel de abstracción, su orientación era básicamente la de simular, por lo que los resultados de una síntesis a partir de descripciones

con estos lenguajes no eran siempre las más optimas.

Además, los lenguajes de descripción de hardware al formar parte de las herramientas EDA permiten el trabajo en equipo. Así, al estructurar el desarrollo del proyecto, cada integrante del equipo de diseño puede trabajar en subproyectos antes de integrar todas las partes del sistema.

1.3.1 VENTAJAS DE LOS HDLS

Una metodología de diseño que utiliza un HDL posee varias ventajas sobre la metodología tradicional de diseño a nivel compuerta. Algunas de estas ventajas son listadas a continuación.

- Es posible verificar el funcionamiento del sistema dentro del proceso de diseño sin necesidad de implementar el circuito.
- Las simulaciones del diseño, antes de que éste sea implementado mediante compuertas, permiten probar la arquitectura del sistema para tomar decisiones en cuanto a cambios en el diseño.
- Las herramientas de síntesis tienen la capacidad de convertir una descripción hecha en un HDL, VHDL por ejemplo, a compuertas lógicas y, además, optimizar dicha descripción de acuerdo a la tecnología utilizada.
- Esta metodología elimina el antiguo método tedioso de diseño mediante compuertas, reduce el tiempo de diseño y la cantidad de errores producidos por el armado del circuito.
- Las herramientas de síntesis pueden transformar automáticamente un circuito obtenido mediante la síntesis de un código en algún HDL, a un circuito pequeño y rápido. Además, es posible aplicar ciertas características al circuito dentro de la descripción para afinar detalles (retardos, simplificación de compuertas, etc.) en la arquitectura del circuito y que estas características se obtengan en la síntesis de la descripción.
- Las descripciones en un HDL proporcionan documentación de la funcionalidad de un diseño independientemente de la tecnología utilizada.
- Un circuito hecho mediante una descripción en un HDL puede ser utilizado en cualquier tipo de dispositivo programable capaz de soportar la densidad del diseño. Es decir, no es necesario adecuar el circuito a cada dispositivo porque las herramientas de síntesis se encargan de ello.
- Una descripción realizada en un HDL es más fácil de leer y comprender que los netlist o circuitos esquemáticos.

1.4 VHDL

VHDL es un lenguaje de descripción de hardware utilizado para describir circuitos en un alto nivel de abstracción el cual está siendo rápidamente aceptado como un medio estándar de diseño. VHDL es producto del programa **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit (VHSIC) desarrollado por el Departamento de Defensa de los Estados Unidos a finales de la década de los 70's. El propósito era hacer un estándar para diseñar, modelar, y documentar circuitos complejos de tal manera que un diseño desarrollado por una empresa pudiera ser entendido por otra y, además, pudiera ser procesado por software con propósitos de simulación.

VHDL es reconocido como un estándar de los lenguajes HDL por el Instituto de Ingenieros en Electricidad y Electrónica – IEEE – como su estándar 1076 el cual fue ratificado en 1987, y por parte del Departamento de Defensa de los Estados Unidos como el estándar MIL-STD-454L En 1993 el estándar IEEE-1076 se actualizó y un estándar adicional, el **IEEE-1164,** fue adoptado. Para 1996 el estándar **IEEE-1076.3** se convirtió en un estándar de **VHDL para síntesis** siendo éste el que se utiliza en el diseño de sistemas digitales. Los estándares más utilizados en síntesis de circuitos por la mayoría de las herramientas de diseño son el IEEE-1164 y el IEEE-1076.3. En la actualidad VHDL es un estándar de la industria para la descripción, modelado y síntesis de circuitos digitales. Por esto, los ingenieros de la mayoría de las áreas de electrónica, si no es que todas, deben aprender a programar en VHDL para incrementar su eficiencia.

VHDL divide los circuitos en dos "vistas" entidades y arquitecturas. La entidad modela al circuito, componente o sistema externamente

definiendo a este mediante un nombre y sus conexiones que vienen siendo las entradas y salidas del circuito. En tanto que la arquitectura, que es la vista interna, define el funcionamiento del circuito. Después de definir las interfaces de la entidad, otras entidades pueden utilizar a la primera como un subcircuito, al mismo tiempo que todas están siendo desarrolladas, es decir, están siendo detalladas en su funcionamiento. Este concepto de vistas externas e internas es propio de VHDL y permite segmentar un sistema en bloques. Así, una entidad es relativa a otras entidades a través de sus conexiones y comportamiento. Por lo que es posible experimentar cada entidad con diferentes arquitecturas sin necesidad de cambiar el resto del diseño. Y obviamente cada entidad puede ser reutilizada en otros sistemas aunque no hayan sido diseñadas específicamente para estos.

Un modelo de hardware de VHDL es mostrado en la siguiente figura.

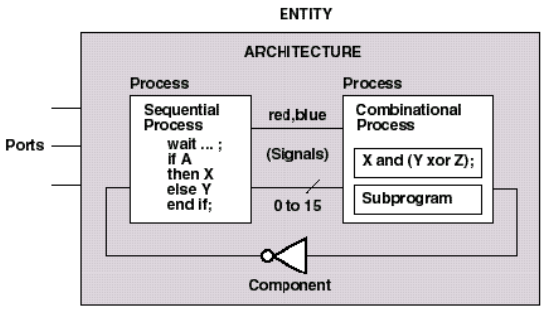

Figura 1.2 Modelo de Hardware de VHDL

Una entidad en VHDL, que ya es todo un diseño, posee una o más conexiones hacia los sistemas que la rodean. Una entidad puede estar compuesta de otras entidades, de procesos y por componentes, todos ellos trabajando concurrentemente. Cada entidad está definida por su arquitectura la cual se forma de instrucciones en VHDL, ya sean aritméticas, asignaciones a señales o de simples instanciaciones de componentes.

Los PROCESS en VHDL son utilizados para modelar tanto circuitos secuénciales como combinacionales utilizando un estilo de descripción secuencial. Para interconectar procesos distintos se utilizan SIGNALS que no son otra cosa que simples cables.

Una señal posee una fuente (driver) y uno más destinos (receptores) y un tipo de dato que le proporciona características de interconexión. Por ejemplo, una señal que se define como tipo bit puede manejar los valores binarios '0' y '1' solamente, en tanto que una señal que se define como bit_vector puede manejar mas de una posición binaria.

La forma de diseñar circuitos en VHDL se divide en tres categorías de acuerdo a su complejidad: flujo de datos, comportamental, y estructural. Estos tres estilos de diseño se detallan a continuación.

- **FLUJO DE DATOS**

En este estilo el diseño del circuito no es complicado por lo que basta con describir como fluyen los datos través de la entidad, de las entradas hacia las salidas. La operación del sistema está definida en términos de un conjunto de transformaciones de datos expresadas como instrucciones concurrentes.

- **COMPORTAMENTAL**

El diseño es un poco más complicado ya que requiere de varias decisiones antes de definir los datos de salida correctos. Por lo que se requiere de una descripción algorítmica del funcionamiento del circuito para facilitar el diseño del sistema. En VHDL esto se obtiene expresando el funcionamiento del diseño mediante una estructura PROCESS la cual se compone de instrucciones secuénciales.

- **ESTRUCTURAL**

Una descripción estructural se utiliza en circuitos que requieren de más de una función, hablando en términos de hardware, para realizar la finalidad del sistema. Para ello segmentamos el sistema en subcircuitos o componentes para facilitar el diseño. Cada componente es caracterizado en particular ya sea utilizando una descripción de flujo de datos o comportamental. Y a la entidad donde se describen las interconexiones de estos componentes recibe el nombre de descripción estructural.

Lo que ha hecho que VHDL sea en un tiempo tan corto el lenguaje de descripción de hardware más utilizado por la industria electrónica, es su independencia con la metodología utilizada por cada diseñador, su capacidad de descripción a diferentes niveles de abstracción, y en definitiva la

posibilidad de poder reutilizar en diferentes aplicaciones un mismo código.

1.5 METODOLOGÍA DE DISEÑO UTILIZANDO VHDL

I. Definición de los requerimientos del sistema.

Antes de comenzar a realizar la descripción del diseño, es muy importante que se tenga una idea clara de los objetivos y requerimientos. Tales como: funciones del circuito, máxima frecuencia de operación, y los puntos críticos del sistema. Esto servirá para poder definir a grandes rasgos cual será la arquitectura del circuito y así comenzar a realizar la descripción.

II. Descripción del circuito en VHDL.

Antes de comenzar a escribir el código es recomendable seleccionar alguna metodología de diseño como: **Top-Down**, **Bottom-Up**, o **Flat**. Los dos primeros involucran la creación de diseños jerárquicos que generalmente son grandes, y el último es utilizado normalmente en el diseño de circuitos pequeños.

La metodología **Top-Down** consiste en dividir el sistema en varios bloques de tal manera que se puedan resolver los problemas por separado, además, cada bloque a su vez se puede dividir en otros bloques si es necesario. El objetivo es que cada bloque tenga una función especifica representada mediante un componente que desempeñe dicha función. **Bottom-Up** es todo lo contrario, comenzamos por caracterizar los componentes básicos del circuito y con estos formamos bloques de mayor tamaño que representen un circuito más complejo que sus partes individuales. La metodología **Flat** es comúnmente utilizada para diseños pequeños, donde los requerimientos son pocos y no muy complejos por lo que no nos distraen y no perdemos de vista la funcionalidad del circuito. Este método de diseño es el que utilizamos cotidianamente en el diseño de circuitos digitales, y se le llama **Flat** por que no es necesario seccionar el circuito para poder diseñarlo.

Después de decidir cual será la metodología que debemos implementar entonces comenzamos a describir el circuito de acuerdo con lo que se había establecido. Es recomendable utilizar algún tipo de diagrama a bloques con la descripción del funcionamiento de cada bloque, diagramas de estado, o usar alguna tabla de funcionamiento donde se resumen las funciones de cada bloque en particular. Obviamente existe la posibilidad de cometer errores en VHDL, pero generalmente estos son de son de sintaxis, como ";" al final de cada instrucción, o simplemente por no utilizar adecuadamente alguna instrucción. Algunas ocasiones se podrán tener problemas al tratar de sintetizar el código y esto se debe a que se comete el error de pensar en términos de programación en vez de enfocarnos en la descripción del circuito.

Cuando se utiliza VHDL el objetivo principal es el diseño de hardware y para ello debemos de utilizar técnicas de síntesis apropiadas al lenguaje, ya que se suele cometer el error de comenzar a programar en vez de describir y esto provoca que nos olvidemos del objetivo que es el hardware.

LA CLAVE PARA DESCRIBIR Y SINTETIZAR FÁCILMENTE CIRCUITOS DIGITALES CON VHDL ES PENSAR EN TÉRMINOS DE COMPUERTAS Y REGISTROS Y NO EN FUNCIÓN DE VARIABLES Y SUBRUTINAS

III. Simulación de la descripción en VHDL.

La simulación del código, o simulación funcional, nos permite detectar y corregir errores antes que se implemente en el dispositivo. La modularidad implementada facilita la evaluación del circuito, porque al describir el circuito por bloques podemos analizar cada uno por separado antes de unirlos. Esta simulación equivale a la depuración de programas en los lenguajes de computación.

IV. Síntesis

Síntesis consiste en reducir una descripción realizada en un lenguaje de alto nivel de abstracción a un nivel de compuerta que pueda ser implementada en un circuito. Dicho de otra manera, síntesis es el proceso mediante el cual una descripción es convertida en un listado de conexiones (*netlis*t) entre las compuertas, registros, multiplexores, etc. de un dispositivo lógico programable.

Por ejemplo, una compuerta XOR puede ser sustituido por su equivalente: A XOR B = A'B+AB', o una instrucción IF puede ser en algunas ocasiones una compuerta AND, en otras

una OR, o inclusive toda una función booleana que involucra diferentes tipos de compuertas. Por lo que el proceso de síntesis depende del dispositivo utilizado. Generalmente una misma función es implementada de diferentes formas de acuerdo al dispositivo que estemos utilizando y esto no cambia la funcionalidad del diseño y será la misma si se selecciona el componente adecuado a la complejidad del diseño. El proceso utilizado para sintetizar un código en un CPLD es conocido como **Fitting** o ajuste y consiste en adaptar las ecuaciones booleanas en los diferentes bloques lógicos del dispositivo. Cuando se utiliza un FPGA el proceso empleado se le llama **Place And Route** y consiste en adecuar las ecuaciones a través de varias celdas lógicas. Aunque la finalidad es la misma, la manera en que se sintetiza un código en un CPLD es totalmente distinta a la síntesis de circuitos utilizando FPGAs.

Por otro lado la optimización en la conversión del código VHDL a ecuaciones booleanas depende de tres factores:

I. La descripción del circuito.

II. Los recursos disponibles en el dispositivo seleccionado.

III. Las directivas de síntesis seleccionadas por el diseñador.

La **descripción** es el punto más importante porque de esto dependen los otros dos. En la descripción no solamente tenemos que "decir" como funciona el circuito, además, tenemos que describir en que "forma" debe de hacerlo. No es lo mismo describir el diseño de un sumador de cuatro bits utilizando cuatro módulos que realizan la suma basándose en propagación de bits de acarreo entre estos, a describir un circuito que realice la suma de manera paralela sin utilizar retroalimentaciones. Finalmente suman pero no lo hacen igual. Los **recursos** afectan la forma en que son implementadas las ecuaciones lógicas en el dispositivo. Por ejemplo, un contador de 4 bits con borrado asíncrono no puede ser implementado en un 16V8, porque el registro de la macrocelda de salida del dispositivo no cuenta con esta característica. Finalmente las **directivas de síntesis** influyen directamente en el proceso de cálculo de las ecuaciones que son implementadas en el dispositivo. Algunas de estas directivas son: asignación de pines, sintetizar para maximizar velocidad, sintetizar para optimizar área, y algunas que son descritas en el mismo código, como por ejemplo forzar a que un nodo no sea simplificado o eliminado y pueda ser retenido a la salida de una macrocelda. Cuando se sintetiza para maximizar la frecuencia generalmente quedan funciones con varios términos e incluso hay términos que se repiten en las ecuaciones de los nodos de salida, pero esto se hace para evitar la retroalimentación.

V. Simulación del código sintetizado

Aún y cuando la simulación funcional se haya realizado con éxito, debemos volver a evaluar el circuito que realmente quedó sintetizado en el dispositivo. Ya que las sustituciones de funciones, como el caso de la compuerta XOR, dependerán de las características del dispositivo utilizado. Y es posible que ciertas funciones se ejecuten en más tiempo de los esperado y esto altere el funcionamiento del resto del diseño. Simular el código sintetizado en el circuito permite verificar los retrasos de tiempo de un nodo a otro, evaluar la máxima frecuencia de operación del circuito y verificar que éste funcione adecuadamente. En dado caso que el código no pudiera ser sintetizado podemos tratar de mejorar la descripción, es decir, mejorar el diseño tratando de eliminar registros, compuertas, buffers, etc., encontrar algún error en la descripción, cambiar las directivas de síntesis o definitivamente seleccionar otro dispositivo.

VI. Programación del dispositivo

Después de completar la descripción, la síntesis y la simulación del circuito con éxito, el siguiente paso es generar el archivo que nos permite implementar físicamente nuestro diseño en un dispositivo programable. Todos los programas de VHDL para síntesis generan un archivo con el que podemos programar el dispositivo. Ya sea JEDEC, JTAG, BITSTREAM, etc. de acuerdo al tipo dispositivo y fabricante.