2.4 FPGAs

2.4.1 Concepto

Según Eduardo Carlos Bozich en su trabajo "*Introducción a los dispositivos FPGA. Análisis y ejemplos de diseño"* define a los FPGAs como:

> Los FPGA (Field Programmable Gate Array) son circuitos lógicos programables directamente por el usuario, lo cual requiere de herramientas de costo relativamente bajo, como lo son el software de desarrollo y el dispositivo grabador. La grabación o programación de uno de estos dispositivos se puede llevar a cabo en milisegundos (Bozich, 2005, pág. 17)

También este autor indica que los FPGAs son muy utilizados por fabricantes que producen tecnología a baja escala, comparándolos con los ASICs que son circuitos de producción a gran escala, pero también conlleva un gran costo, a diferencia de los FPGAs donde su funcionalidad es similar a los anteriores a bajo costo, pero, con una velocidad ligeramente menor. También menciona que los FPGAs se utilizan para prototipados, los cuales permiten depurar y refinar su diseño (Bozich, 2005).

Por otro lado Sisterna (n.d.) indica "Un FPGA es un dispositivo que un diseñador de sistemas digitales puede programar, después que está soldado en el circuito impreso, para que funcione de un modo determinado. Los FPGAs son fabricados con conexiones y lógica programables" (pág. 1).

Otra definición que nos podría ayudar a clarificar el concepto de FPGA es la dicha por Benjamín Rivas la cual es:

> Un FPGA (Field Programmable Gate Array) es un dispositivo de silicio semiconductor con la capacidad de ser poder programar una operación deseada, su mapeo interno es similar a una matriz construida con elementos electrónicos interconectados entre sí y cuya estructura es propia del fabricante, pueden estar compuestos por CLB (Configurable Logic Block) o por LAB (Logic Array Block), en esencia los dispositivos FPGA siguen una misma lógica no importando el fabricante, lo que puede cambiar son los elementos que componen cada uno de esos bloques (Robles Rivas, 2016, pág. 3).

Como vemos los FPGAs en su definición son ejemplos de funcionalidad a bajo costo y sobre todo personalizables como lo dice Rubén Cárdenas:

> Una FPGA es un chip que según cómo se configure, puede realizar cualquier circuito digital. una FPGA más grande, con más recursos internos, alcanza a implementar diseños más complejos. Pero al final se tiene una manera de poder crear diseños digitales sin tener que utilizar componentes externos. Y lo interesante es que una vez configurada la FPGA, lo que tenemos en su interior es hardware.(Cárdenas Espinoza, 2009, pág. 13)

Y por lo anterior se hará este circuito indispensable para la realización de nuestra mano prostética.

2.4.1.1 Arquitectura

Ahora por la parte de su arquitectura en general, Bozich (2005) menciona que un FPGA "consiste en arreglos de varios bloques programables (bloques lógicos) los cuales están interconectados entre sí y con celdas de entrada/salida mediante canales de conexión verticales y horizontales" (pág. 17) como muestra la figura 1.

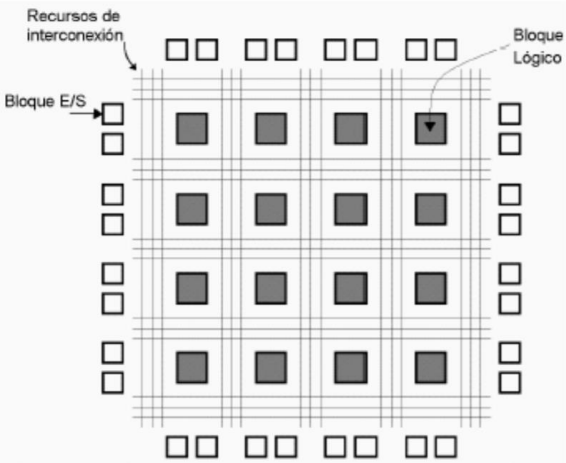


Figura 1). Arquitectura básica de un FPGA

"En general, se puede decir que posee una estructura bastante regular, aunque el bloque lógico y la arquitectura de rutado varía de un fabricante a otro."(Bozich, 2005, pág. 17).

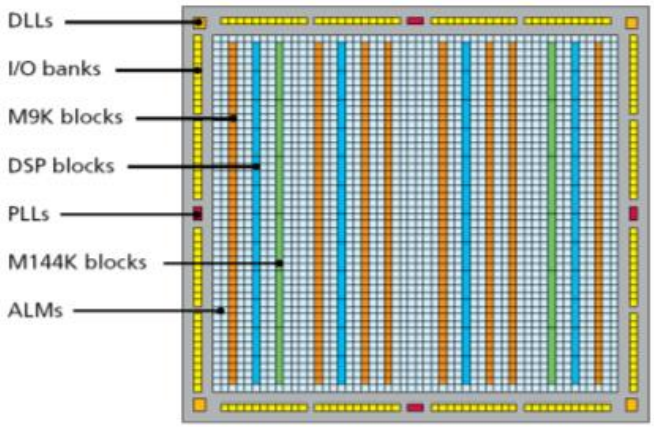
Por consiguiente, también no redacta que:

> La arquitectura de la FPGA cuenta también con un bloque lógico con una parte combinacional y una parte secuencial. La parte combinacional, que permite implementar funciones lógicas booleanas, más una parte secuencial que permite sincronizar la salida con una seña de reloj externa e implementar registros. La parte combinacional varía de un fabricante a otro. (Bozich, 2005, pág. 18).

En la parte de arquitectura (Sisterna, n.d.) nos complementa "Todos los FPGAs, independientemente del fabricante, tienen ciertos elementos en común, tienen un arreglo tipo matricial de elementos lógicos, como flips-flops y lógica combinacional, que se configuran usando cierta tecnología de programación." (pág. 1).

La Figura 2 muestra la composición y disposición de los componentes lógicos en un FPGA de la empresa Altera.

Figura 2). Estructura y componentes de un FPGA de la empresa Altera.



También (Sisterna, n.d.) nos dice:

> Si bien para la configuración de un FPGA particular se usa un software específico del fabricante del FPGA, la tendencia actual es tratar de realizar el diseño digital, en un diagrama esquemático o en Lenguaje de Descripción de Hardware (Hardware Description Language, HDL), con la máxima abstracción del FPGA a usar. De modo que, si por una razón u otra es necesario cambiar de fabricante de FPGA, sea posible una transición lo más fácil posible. Esto se le llama diseño transportable. (pág. 2).

2.4.1.2 Tecnologías de programación.

Además de su arquitectura también es importante mencionar las tecnologías más utilizadas según Bozich E. (2005) para para crear las conexiones entre canales ósea, su tecnología de programación:

> 1. Antifusible (Antifuse): Al igual que la tecnología PROM (memoria de solo lectura programable), un FPGA que utiliza este tipo de tecnología sólo se puede programar una sola vez, y utilizan algo similar a un fusible para las conexiones. Una vez que es programado ya no se puede recuperar. La diferencia entre un fusible y un antifusible es que el primero se desactiva deshabilitando la conexión, en cambio, para el segundo se produce una conexión cuando son programados, por lo que normalmente se encuentran abiertos. La desventaja obvia es que no son reutilizables, pero por otro lado disminuyen considerablemente el tamaño y costo de los dispositivos.
>
> 2. SRAM (StaticRAM): Estas guardan la configuración del circuito. Esto quiere decir que las SRAM son utilizadas como generadores de funciones y además son usadas para controlar multiplexores (que están incluidos en los FPGAs) y la interconexión entre bloques. En éstas el contenido se almacena mediante un proceso de configuración en el momento de encendido del circuito que contiene al FPGA. Ya que al ser SRAM, el contenido de la memoria se pierde cuando se deja de

suministrar energía; la información binaria de las celdas SRAM generalmente se almacena en memorias seriales EEPROM conocidas como memorias de configuración o celdas de configuración. En el momento de encendido del circuito toda la información binaria es transferida a los bloques e interconexiones del FPGA mediante el proceso de configuración el cual es generalmente automático, dado que el propio FPGA contiene un circuito interno que se encarga de hacer toda la programación.

3. Flash: El avance experimentado en los últimos años en el diseño y prestaciones de las celdas de memoria Flash ha permitido su incorporación reciente al mundo de los dispositivos programables como tecnología de programación. Las FPGAs basadas en celdas Flash recogen las ventajas principales de las dos técnicas anteriores situándose en un punto intermedio. Su tamaño es bastante más reducido que el de una celda de SRAM, aunque sin llegar al tamaño reducido de un antifusible; son reprogramables, aunque la velocidad de programación es bastante más lenta que en el caso de una SRAM; y son no volátiles, por lo que no necesitan un dispositivo auxiliar para guardar la configuración interna, como en el caso de la SRAM. (pág. 21)

2.4.2 FPGAs como Coprocesador.

2.4.2.1 Concepto

En el artículo "*A FPGA based Forth microprocessor*" se habla de que:

> Los sistemas que emplean un microprocesador juntos con un coprocesador basado en FPGA específico de la aplicación son comunes hoy en día. Estas aplicaciones pueden reducir el consumo de energía y los costos del sistema al incorporar el microprocesador en la FPGA. Para tales aplicaciones, un microprocesador que tiene un buen rendimiento ocupa una cantidad mínima de recursos de FPGA, tiene un buen entorno de desarrollo de software de lenguaje de alto nivel y una buena densidad de código es deseable.(Leong et al., n.d., párr. 1)

Ahora bien, "con las computadoras personalizadas, el objetivo es acelerar un programa a través de la ejecución de operaciones críticas en un coprocesador que proporciona hardware reconfigurable que evalúa directamente la operación deseada" (Athanas et al., 1995, pág. 20) .

Por consiguiente "Los ordenadores personalizados (basados en FF'GA máquinas de computación o FCCMs) se construyen con básicamente tres elementos: FPGAs, memorias y FieldPrograma ble Interconnection Circuits (FPICs)." (Athanas et al., 1995, pág. 20).

2.4.2.2 Ventajas

También algunas ventajas de FPGA como coprocesador según Cuenca Martínez (2006) en su trabajo *"Diseño e Implementación en Hardware de un Filtro de Gabor para el Mejoramiento de Imágenes de Huellas Digitales-Edición Única"* son la posibilidad de realizar varias 8 operaciones en paralelo mientras que el software opera de manera secuencial. Otra ventaja es que ambos, el procesador y el coprocesador, pueden trabajar en diferentes tareas al mismo tiempo. Ahora bien, en consumo de potencia en el FPG se puede separar en tres partes: Potencia de arranque, potencia estática y potencia dinámica.

2.4.2.2.1 Potencia

En la potencia de arranque en los FPGAs es consumida al encender el FPGA. Debido a que su memoria es SRAM, una vez que se prende, se tienen que descargar los datos y la configuración hacia el FPGA. Esto provoca dos situaciones de consumo de potencia, la primera es que al encender el FPGA, mientras el voltaje de alimentación llega al voltaje correcto, los estados desconocidos de la memoria SRAM provocan picos de corriente altos; la segunda situación se presenta al descargarse la configuración, lo que provoca que haya actividad y comunicación entre el FPGA y la memoria. Compañías como Actel y QuickLogic han lanzado al mercado FPGAs basados en memoria Flash para reducir el consumo de potencia de arranque.

Por su parte La potencia estática, es la potencia que consume el FPGA, aunque no esté realizando ninguna actividad. Esta potencia es debido a la corriente de fuga. La potencia estática se puede calcular con la siguiente fórmula: $P_{Est} = I_{leak} * V_{dd}$

Otro factor que ha hecho que la potencia estática este siendo tomada en cuenta, es la disminución del tamaño en las tecnologías de fabricación de los transistores. Entre menor sea la separación entre las terminales menor es el voltaje umbral y se incrementa la corriente de fuga a tal grado que se estima que la potencia estática alcance y rebase en su magnitud a la potencia dinámica.

Y por último la potencia dinámica es la potencia que consume el FPGA debido a la carga y descarga de las capacitancias durante el cambio de estado de cada nodo. Esta se puede calcular con la siguiente fórmula: $P_{Din} = AC_L V_{dd}^2 F$

Además de esta componente de potencia, existe lo que se llama la potencia de corto circuito y esta ocurre en los inversores CMOS, Figura 3.

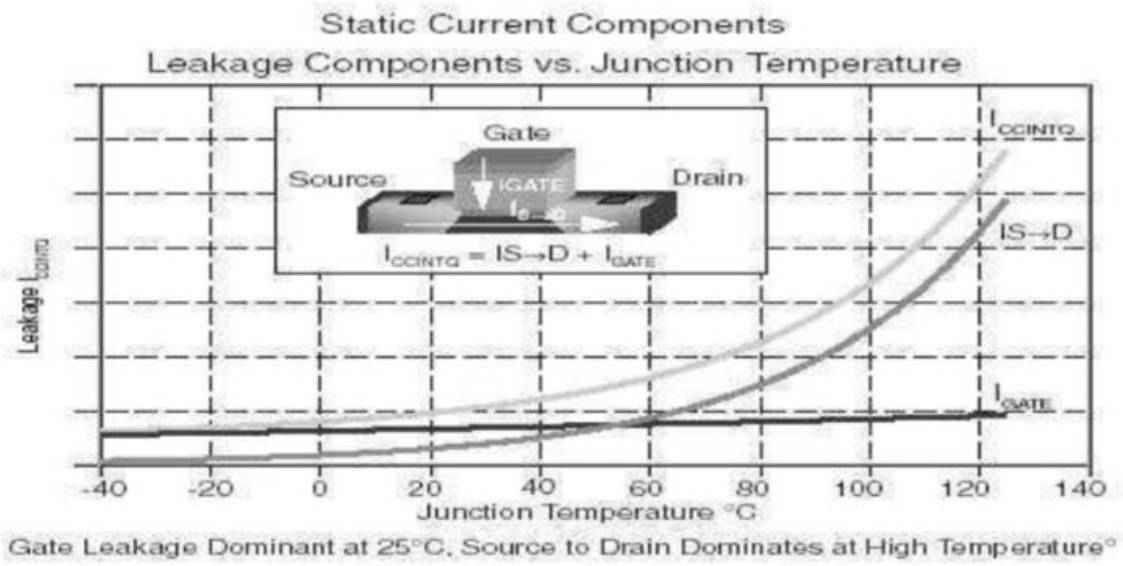


Figura 3). Potencia Estática en Función de la Temperatura

Esta potencia es debida a que en cada transición hay un momento en el que los transistores N y P están prendidos al mismo tiempo. La distribución de la energía dinámica se puede separar en tres sectores: interconexión, bloques lógicos y la distribución del reloj, según investigaciones anteriores esta distribución tiene valores de entre 55-70 % 10-20 % y 10-20 % respectivamente.

2.4.3 Lenguajes de descripción de hardware (HDL)

2.4.3.1 Definición

En general (Díaz Fernández, 2017) define que:

> Un HDL es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos y electrónicos digitales. Así, estos lenguajes hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación. (pág. 3)

También se menciona que los HDL tiene dos aspectos que facilitan las descripciones de hardware, el modelado de comportamiento abstracto y el modelado de estructura de hardware, el primero es un lenguaje declarativo con el fin de facilitar la descripción del comportamiento hardware del circuito implementado y el segundo es la estructura hardware que puede ser modelada en un HDL independiente al HDL usado para modelar el comportamiento del diseño (Díaz Fernández, 2017).

Díaz Fernández especifica que:

> Este comportamiento puede ser modelado y representado en varios niveles de abstracción durante el proceso de diseño. Los modelos con altos niveles de

> abstracción describen el comportamiento del diseño de forma abstracta, mientras que los modelos con niveles bajos de abstracción incluyen más detalle. (2017, pág. 15).

Por otro lado, menciona que "Los lenguajes de descripción hardware son lenguajes especializados en la descripción de estructuras, el diseño de estas y el comportamiento del hardware."(Díaz Fernández, 2017, pág. 15).

Para finalizar se puede decir que con estos lenguajes se pueden representar diagramas lógicos, circuitos con diferentes grados de complejidad, desde expresiones booleanas hasta circuitos más complejos. Estos lenguajes sirven para representar sistemas digitales de manera legible para las máquinas y para las personas.

Para clarificar un poco mas este tema se hará una analogía proporcionada por (Aparicio Olmedo & Ponluisa Marcella, 2014) entre los lenguajes de descripción de software y los lenguajes de descripción de hardware como se muestra en la figura 4) (pág. 36).

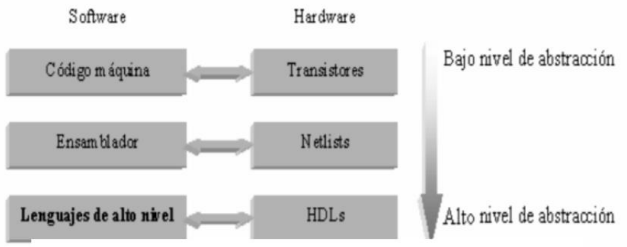


Figura 4). Analogía con lenguaje de descripción de software Fuente: http://oretano.iele-ab.uclm.es/~miniesta/intro%20hdl.pdf

2.4.3.2 Características

Algunas características de los HDL como la descripción de las actividades que ocurren en forma simultánea (concurrencia), también permite describir módulos con acciones que serán evaluadas en forma secuencial (procedural), donde todo el módulo será visto como una acción concurrente más, otra es que posibilita la construcción de una estructura jerárquica, donde es posible combinar descripciones estructurales y de flujo de datos con descripciones de comportamiento (behavior) y por ultimo permite modelizar el concepto de "tiempo", fundamental para la descripción de sistemas electrónicos (Aparicio Olmedo & Ponluisa Marcella, 2014).

2.4.3.3 Metodología de diseño

Según Díaz Fernández (2017) la metodología de diseño:

> Es la manera en la que los procesos, respecto a la complejidad y la abstracción del diseño, se ordenan para conseguir que el costo y el tiempo de desarrollo sea el menor posible. Esto garantiza la confiabilidad del producto final y sus prestaciones además de lograr la mayor independencia de las herramientas y tecnologías (pág. 16).

Díaz Fernández (2017) redacta que la metodología que por lo general se utiliza para el diseño es:

- Definir el nivel de abstracción inicial.
- Realizar una descomposición jerárquica.
- Definir la estructuración de los nuevos niveles jerárquicos.
- Desarrollar la arquitectura necesaria.
- Seleccionar la tecnología que vamos a utilizar. (pág. 17)

También que existen dos diseños diferentes para orientar el orden de las acciones denominado flujo de diseño:

- Diseño Bottom -Up: Este diseño no es recomendado debido a que es ineficiente en diseños complejos y a que depende de la tecnología. Se empiezan describiendo los componentes más pequeños del sistema, estos componentes se agrupan en bloques de mayor complejidad y así sucesivamente hasta conseguir un único bloque.
- Diseño Top-Down: Este diseño es el más utilizado debido a que sus descripciones son independientes de la tecnología lo que aumenta la reutilización del diseño en diferentes casos. Se comienza con un sistema a nivel funcional, incluyendo la descripción y simulación de especificaciones. Se descomponen los diferentes sistemas en subsistemas y bloques hasta llegar a una descripción en componentes sintetizables (pág. 17)

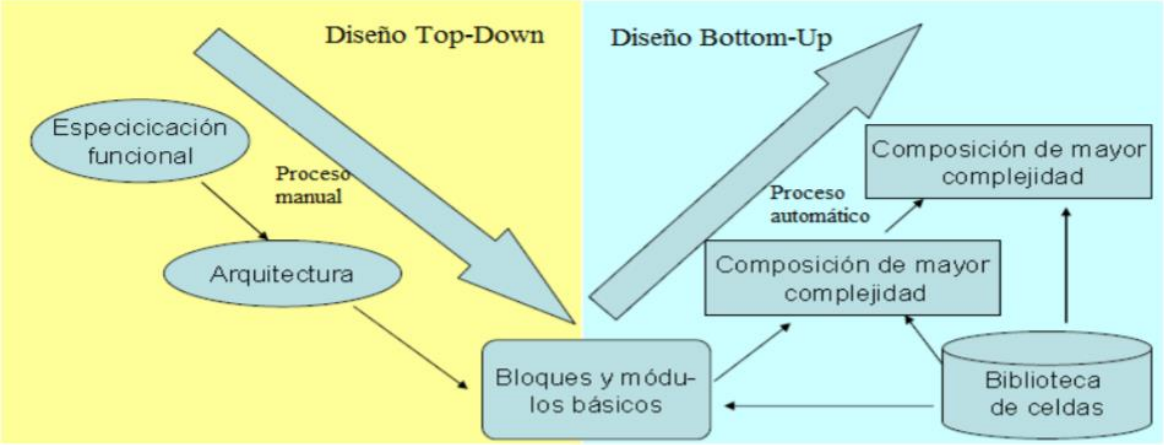


Figura 5). Diseño Top-Down y Bottom-up

2.4.3.3 Esquema típico de diseño de un sistema digital

En la siguiente figura 6 se obtenida de (Aparicio Olmedo & Ponluisa Marcella, 2014) donde se puede observar el esquema típico de diseño de un sistema digital. Las áreas sombreadas en la

figura representan los procesos en el flujo de diseño, las áreas sin sombra los distintos niveles de representación del diseño (pág. 37).

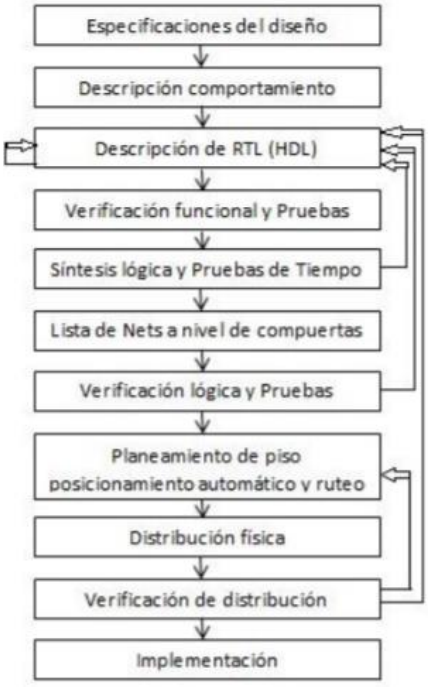


Figura 6). Esquema típico de un diseño digital.

Aparicio Olmedo y Ponluisa Marcella (2014) nos indican que:

> Este esquema representa el típico flujo de diseño utilizado en un diseño basado en HDL. En todo diseño la especificación debe ser descrita primero, esta describe de forma abstracta la funcionalidad, interfaz y arquitectura global del diseño a ser escrito. En este punto no es necesario pensar de forma detallada la manera de la implementación del circuito. La descripción en comportamiento se crea con fines de evaluar el diseño en términos de funcionalidad, desempeño y otras características de alto nivel. Esto se logra utilizando lenguajes de alto nivel (HDL). La descripción a nivel de registros de transferencia (RTL) se crea manualmente en HDL. El diseñador debe describir el flujo de datos que implementará el circuito digital. Desde este punto en adelante el diseño se lo hace con la ayuda de una herramienta EDA (Electronic Design Automation). Generalmente estas herramientas son provistas por los fabricantes de los chips o por empresas aliadas. Las herramientas de síntesis lógica convierten la descripción de RTL a lista de interconexión de Nets o puntos en un nivel de compuertas lógicas. Esta es una lista de interconexiones entre las compuertas lógicas. Las herramientas de síntesis lógica aseguran que las conexiones entre nets cumplan con los tiempos, áreas y especificaciones de potencia. La lista de Nets a nivel de compuertas es entonces alimentada a una herramienta automática de posicionamiento y ruteo la cual crea el "Layout" o distribución de las compuertas. Este layout es verificado y después implementado físicamente en un chip. La mayor parte del tiempo de diseño se

concentra en la optimización manual de la descripción del circuito a nivel de RTL. Una vez que esta está completa y ha satisfacción del diseñador, las herramientas EDA se utilizan para procesos posteriores. El diseño a nivel de RTL ha reducido los tiempos de diseño de años a unos pocos meses. Con la aparición de herramientas de síntesis descriptiva se puede crear descripciones de RTL desde una descripción algorítmica o de comportamiento del circuito. Mientras estas herramientas mejoran su desempeño más se parece el diseño digital a una programación de alto nivel de un computador. Los diseñadores implementan el algoritmo en un HDL a un nivel muy abstracto y las herramientas EDA ayudan a convertir e implementar este algoritmo en un chip o FPGA. (pág. 37-39).

2.4.3.4 Tipos

Existen tres tipos de lenguajes HDL según (Díaz Fernández, 2017):

1. De bajo nivel: Permiten definir un circuito a nivel de arquitectura (FlipFlops, compuertas básicas, ecuaciones lógicas) como son los lenguajes PALASM, CUPL, ABEL.
2. De nivel medio: permiten definir un circuito en modo jerárquico, así como la generación condicional/iterativa de hardware; en ciertos permiten el uso de descripciones de comportamiento (funciones aritméticas, máquinas de estado), como es el lenguaje AHDL.
3. De alto nivel: no sólo posibilitan mayor nivel de abstracción, sino que también son usados para la simulación, para la síntesis del generador de estímulos y el monitor de salidas, como son los lenguajes: VHDL, VERILOG HDL. (pág. 39)

Bibliografía

Aparicio Olmedo, A. A., & Ponluisa Marcella, N. F. (2014). *ESTUDIO COMPARATIVO DE LOS LENGUAJES HDL Y SU APLICACIÓN EN LA IMPLEMENTACIÓN DEL LABORATORIO DE SISTEMAS DIGITALES AVANZADOS MEDIANTE FPGAS EN LA EIE-CRI*.

Athanas, P., Pocek, K. L., & IEEE Computer Society. Technical Committee on Computer Architecture. (1995). *IEEE Symposium on FPGAs for Custom Computing Machines : proceedings, April 19-21, 1995, Napa Valley, California*. IEEE Computer Society Press.

Álvarez Ariza, J. & González Gil, S. (2018). *Lenguaje de descripción de hardware (VHDL).*Corporación Universitaria Minuto de Dios. https://elibro.net/es/ereader/biblioitmorelia/126079?page=1

Bozich, E. C. (2005). *Introducción a los Dispositivos FPGA. Análisis y ejemplos de diseño*.

Cardenas Espinoza, R. D. (2009). *CURSO FPGA (PROGRAMACION DE ARREGLOS DE COMPUERSTAS)*.

Cuenca Martínez, R. (2006). *Diseño e Implementación en Hardware de un Filtro de Gabor para el Mejoramiento de Imágenes de Huellas Digitales-Edición Única*.

Díaz Fernández, B. (2017). *Lenguajes de descripción hardware para la síntesis de circuitos: VHDL y Verilog. Analogías y diferencias. Aplicación a un caso práctico*.

Leong, P. H. W., Tsang, P. K., & Lee, T. K. (n.d.). *A FPGA based Forth microprocessor*. www.dnai.comm-

Robles Rivas, B. A. (2016). *Principios del FPGA y aplicaciones en el control de procesos industriales. INGENIERO ELECTRICISTA*.

Sisterna, C. (n.d.). *FIELD PROGRAMMABLE GATE ARRAYS (FPGAS)*.