Verilog y VHDL diferencias ventajas y desventajas
Verilog and VHDL Differences advantages and disadvantages

1st Javier Chacon
Universidad Industrial de Santander
Bucaramanga, Santander
javierchacon262@gmail.com

2nd Andrea Villamizar
Universidad Industrial de Santander
Bucaramanga, Santander
andreavillamizar139@gmail.com

3rd Daniel Ardila
Universidad Industrial de Santander
Bucaramanga, Santander
danielardila02@gmail.com

Abstract

Hardware description languages are specialized computer languages used to describe the structure and behavior of electronic circuits, there are forms to do this, but some of the most used are **VHDL** (i.e., very high speed integrated circuit description language) and its counterpart named **Verilog** which is a generalized version that is standardized as **IEEE 1364**. This paper intends to compare them in all the possible manners, go deeply into the advantages and disadvantages of using each one, how they can be traduced to each other and what implies to make the choice of using some of these for applied projects.

Los lenguajes de descripción de hardware están especializados en lenguajes de computador usados para describir la estructura y el comportamiento de circuitos electrónicos, hay varias formas de hacer esto, una de las más usadas es VHDL (lenguaje de descripción de circuitos integrados de velocidades muy altas) y su contraparte verilong que es una versión generalizada que está estandarizada como IEEE 1364. Este documento tiene la intención de compararlos de todas las maneras posibles, profundizar en las ventajas y desventajas de usar cada uno, cómo se pueden traducir entre sí y qué implica hacer una elección para usar alguno de estos al realizar proyectos aplicados.

I. Introducción

A finales de los 80 los diseños de circuitos integrados de aplicación crecían y los sistemas gráficos que los describían empezaron a mostrar limitantes, esto debido a que la visualización, depuración, desarrollo y mantenimiento de estos sistemas era cada vez más complicada, gracias a esto creció la necesidad de lenguajes de programación especializados. A partir de 1983, VHDL se desarrolló originalmente a instancias del Departamento de Defensa de los EE. UU. para documentar el comportamiento de los ASIC que las empresas proveedoras incluían en los equipos. En 1985 una compañía llamada Automated Integrated Design Systems (sistemas automatizados de diseño integrado) después renombrada como Gatewat Design Automation introdujo un nuevo producto llamado Verilog, un simulador lógico que integraba un lenguaje de alto nivel y simulación a nivel de compuertas lógicas.

Un lenguaje de programación especializado(HDL) es utilizado principalmente para definir la estructura, diseño y operación de circuitos electrónicos y electrónicos digitales, de estos lenguajes destacan tanto Verilog como VHDL, los cuales tienen numerosas ventajas. Verilog y VHDL se utilizan para llevar a cabo programas para chips electrónicos como son FPGAs y ASICs. Este artículo tiene como finalidad comparar características y elementos de los lenguajes de descripción hardware más populares, y de ser posible se concluirá cual lenguaje de los anteriormente nombrados se prefiera. Para alcanzar el objetivo principal del artículo es necesario inicialmente aclarar conceptos importantes, conceptualizar, y proceder a definir cada uno de los lenguajes, mostrar un ejemplo y así llegar a una conclusión; al concluir, se sabrá con exactitud el uso de los lenguajes de programación especializados y la importancia que tienen hoy en día.

II. Marco teórico

II-A. Lenguaje de descripción de hardware:

Un lenguaje de descripción de hardware (HDL, hardware description language) es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales, como el convertidor analógico-digital o cualquier antena satelital. Así, los lenguajes de descripción de hardware hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación. Los lenguajes de descripción de hardware se parecen mucho a otros lenguajes de programación de ordenadores tales como el C o Java: básicamente consisten en una descripción textual con expresiones, declaraciones y estructuras de control. Sin embargo, una importante diferencia entre los HDL y otros lenguajes de programación está en que el HDL incluye explícitamente la noción de tiempo. Así, los HDL pueden ser usados para escribir especificaciones ”ejecutables” de hardware. Es decir, un programa escrito en HDL hace posible que el diseñador de hardware pueda modelar y simular un componente electrónico antes de que este sea construido físicamente. Es esta posibilidad de ”ejecución” de componentes lo que hace que a veces los HDL se vean

como lenguajes de programación convencionales, cuando en realidad se debería clasificarlos más precisamente como lenguajes de modelado. En la práctica existen distintos tipos de simuladores capaces de trabajar tanto con eventos discretos (digitales) como continuos (analógicos), existiendo lenguajes HDL específicos para cada caso.

II-B. VHDL (Very High Speed Integrated Circuit Hardware Description Language):

VHDL es un lenguaje de descripción de circuitos electrónicos digitales que utiliza distintos niveles de abstracción. El significado de las siglas VHDL es VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. Esto significa que VHDL permite acelerar el proceso de diseño. VHDL no es un lenguaje de programación, por ello conocer su sintaxis no implica necesariamente saber diseñar con él. VHDL es un lenguaje de descripción de hardware, que permite describir circuitos síncronos y asíncronos. Para realizar esto debemos:

- Pensar en puertas y biestables, no en variables ni funciones.
- Evitar bucles combinacionales y relojes condicionados.
- Saber qué parte del circuito es combinacional y cuál secuencial.

¿Por qué usar un lenguaje de descripción hardware?

- Poder descubrir problemas en el diseño antes de su implementación física.
- La complejidad de los sistemas electrónicos crece exponencialmente, es necesaria una herramienta que trabaje con el ordenador.
- Permite que más de una persona trabaje en el mismo proyecto.

En particular VHDL permite tanto una descripción de la estructura del circuito (descripción a partir de subcircuitos más sencillos), como la especificación de la funcionalidad de un circuito utilizando formas familiares a los lenguajes de programación.

II-C. Verilog:

Verilog es un lenguaje de descripción de hardware (HDL, del Inglés Hardware Description Language) usado para modelar sistemas electrónicos. El lenguaje, algunas veces llamado Verilog HDL, soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción. Cabe destacar que es lo más extensamente posible HDL usado con una comunidad de usuario más del active de 50.000 diseñadores. Los diseñadores de Verilog querían un lenguaje con una sintaxis similar a la del lenguaje de programación C, de tal manera que le resultara familiar a los ingenieros y así fuera rápidamente aceptada. El lenguaje tiene un preprocesador como C, y la mayoría de palabras reservadas de control como ”if”, ”while”, etc, son similares. El mecanismo de formateo en las rutinas de impresión y en los operadores del lenguaje (y su precedencia) son también similares. A diferencia del lenguaje C, Verilog usa Begin/End en lugar de llaves para definir un bloque de código. Por otro lado la definición de constantes en Verilog requiere la longitud de bits con su base. Verilog no tiene estructuras, apuntadores o funciones recursivas. Finalmente el concepto de tiempo, muy importante en un HDL, no se encuentra en C.

III. Contenido

III-A. Lenguaje de descripcion de hardware (HDL)

El modelado de comportamiento abstracto y el modelado de la estructura hardware son dos aspectos de los lenguajes de descripción Hardware(HDL), el cual es un lenguaje de programación software que es empleado a la hora de modelar el funcionamiento de un circuito hardware. El modelado de comportamiento abstracto hace referencia a un lenguaje cuando es declarativo con el objetivo de facilitar la descripción del comportamiento hardware de un circuito el cual está siendo implementado, por otro lado, el modelado de la estructura hardware es aquella estructura que puede ser modelada en un HDL independientemente al lenguaje de descripción hardware usado para modelar el comportamiento del diseño, teniendo la posibilidad de ser modelado y representado en distintos niveles de abstracción durante el proceso de diseño.

La diferencia principal entre los modelos con altos y bajos niveles de abstracción es que mientras que los modelos con niveles bajos de abstracción incluyen más detalles, los modelos de bajos niveles de abstracción describen el comportamiento del diseño de forma abstracta. Gracias a los lenguajes de descripción hardware se puede representar diagramas lógicos y circuitos con diferentes grados de complejidad, además, representan sistemas digitales de manera legible tanto para las máquinas como para las personas

La metodología de diseño en un lenguaje de descripción de hardware habla de la manera en la que los procesos, destacando la complejidad y la abstacción del diseño, se ordenan para lograr que el costo y el tiempo de desarrollo sea el menos posible,

de esta manera se logra ofrecer una mejor calidad, garantizando la confiabilidad del producto obtenido. Los pasos a seguir para llevar a cabo la metodología de una mejor forma son: Definir el nivel de abstracción inicial, realizar una descomposición jerárquica, definir la estructuración de los nuevos niveles jerárquicos, desarrollar la arquitectura necesaria, y finalmente, seleccionar la tecnología que se va a utilizar.

El diseño Bottom-Up y Top-Down son dos diseños diferentes que se llevan a cabo para orientar el orden de las acciones denominado flujo de diseño. El diseño Top-Down es el más utilizado, gracias a que sus descripciones son independientes de la tecnología, esto lleva a un aumento en la reutilización del diseño en diferentes casos. El dieño Bottom-Up no es un diseño recomendado ya que en la mayoría de los casos ha resultado ser ineficiente en diseños complejos y dependiente de la tecnología .

Usando HDL se puede describir la operación del sistema con diferentes niveles de abstracción:

- Nivel de conmutadores, se utiliza para describir el circuito en términos de transistores y cables.
- Nivel de compuertas, para describir el circuito en términos de compuertas lógicas y elementos de almacenamiento como flip-flops.
- Nivel de flujo de dato, los cuales describen el circuito en términos de flujo de datos entre registros.
- Nivel algorítmico o comportamental, incluye instrucciones de alto nivel como lazos, comandos de decisión y otros, su comportamiento es similar a un programa en un lenguaje de alto nivel como C.

Dentro de las características principales de los HDL se encuentran que permiten modelizar el concepto de tiempo, esta característica es fundamental para llevar a cabo la descripción de sistemas electrónico; describe actividades que ocurren en forma simultánea, permite tanto la construcción de una estructura jerárquica como la descripción de módulos con acciones que se evalúan en forma secuencial. La construcción de una estructura jerárquica posibilita combinar descripciones estructurales y de flujo de datos con descripciones de comportamiento.

Los lenguajes de descripción hardware se clasifican en tres tipos de lenguaje segpun su nivel:

- De alto nivel: Además de que posibilitan mayor nivel de abstracción, son usados para la simulación, sintesís del generador de estímulos y el monitor de salidas, los lenguajes que entran en este tipo de HDL son los lenguajes: VHDL, VERILOG HDL.
- De nivel medio: Permiten definir la generación condicional/iterativa de hardware, como también definir un circuito en modo jerárquico, en algunos casos permiten el uso de descripciones de comportamiento, como es el lenguaje AHDL.
- De bajo nivel: Permiten la definición de un circuito a nivel de arquitectura como son FlipFlops, compuertas básicas, ecuaciones lógicas. Los lenguajes pertenecientes a bajo nivel son: ABEL, CUPL, PALASM.

  Existen variedad de ventajas al utilizar HDL como herramienta para sistemas de diseño digital. Gracias a sus herramientas de especificación es posible su uso para la especificación general de un sistema en hardware y software, de igual manera describe el hardware, sus sistemas, subsistemas y componentes. Las herramientas de diseño proporcionan mejor documentación y facilita el proceso de reuso, adicional a esto presenta independencia tecnológica y posibilidad de parametrización. Al hablar de herramientas de simulación cabe destacar la facilidad para generar vectores de test complejos, como también la disponibilidad existente de modelos de distintos componentes de fabriantes variados en HDL normalizados.

  Así como se encuentran ventajas al llevar a cabo un proyecto con un lenguaje de descripción hardware, existen distintas desventajas entre estas se encuentra que su uso involuntariamente pierde interés sobre el aspecto físico del diseño ya que se enfoca principalmente a su funcionalidad. La necesidad de adquirir conocimientos desde cero de este lenguaje es otra desventaja ya que supone un esfuerzo de aprendizaje debido a que es una nueva metodología que se debe estudiar y aprender a llevar. Es necesaria la utilización de nuevas herramientas como simuladores y sintetizadores de HDL.

  Las aplicaciones enfocadas en el lenguaje de descripción de hardware se basan en modelización, simulación, síntesis, y reusabilidad. La modelización se enfoca en la creación del diseño, para llegar a esto se lleva a cabo un diseño de flujo de datos que nos ayuda a hacernos una idea de los componentes que se necesitan en el diseño. La simulación

se lleva a cabo para verificar el correcto funcionamiento de los modelos en cualquier etapa del diseño además de corregir fallos de forma teórica, sin necesidad de que el circuito este implementado. Para la síntesis se usan los HDL como entrada en las herramientas de síntesis, como salida se obtiene el diseño del circuito. Reusabilidad se basa en la capacidad que tiene el código que se ha creado en ser usado en otros programas diferentes, es decir, el reúso de este trabajo.

Hay variedad de lenguajes de descripción hardeware, aunque principalmente nos enfocaremos en VHDL y Verilog.

III-B. VHDL

VHDL es un lenguaje de descripción de circuitos electrónicos digitales, mediante él se pueden describir los comportamientos de los circuitos. Este comportamiento puede transferirse a algún dispositivo con sus componentes propios, y con ellos logrará el comportamiento deseado. El comportamiento el dispositivo es independiente al hardware del dispositivo.Para saber diseñar circuito con VHDL, no solo se necesita conocer su sintaxis, sino que también es necesario tener en cuenta que es necesario pensar con puertas lógicas y biestables, no en variables y funciones como en los lenguajes de programación. También es importante evitar los bucles combinacionales y los relojes que estén condicionados. Y la última cuestión a tener en cuenta es que es necesario saber diferenciar las partes del circuito en combinacionales y secuenciales. Este leguaje permite una descripción de la estructura del circuito a partir de subcircuitos más sencillos, y también permite especificar la funcionalidad de un circuito usando un formato similar al de los lenguajes de programación, permitiendo a los usuarios adaptarse más fácilmente a este tipo de lenguaje. El VHDL es un estándar de dominio público llamado IEEE 1076-1993. Al ser un estándar no depende de ningún fabricante o dispositivo, es independiente; esto también provoca que se puedan reutilizar los diseños; y por último, al ser un estándar tiene la ventaja de que es un diseño jerárquico, por lo que se mantiene un orden y se siguen ciertas reglas jerárquicas.

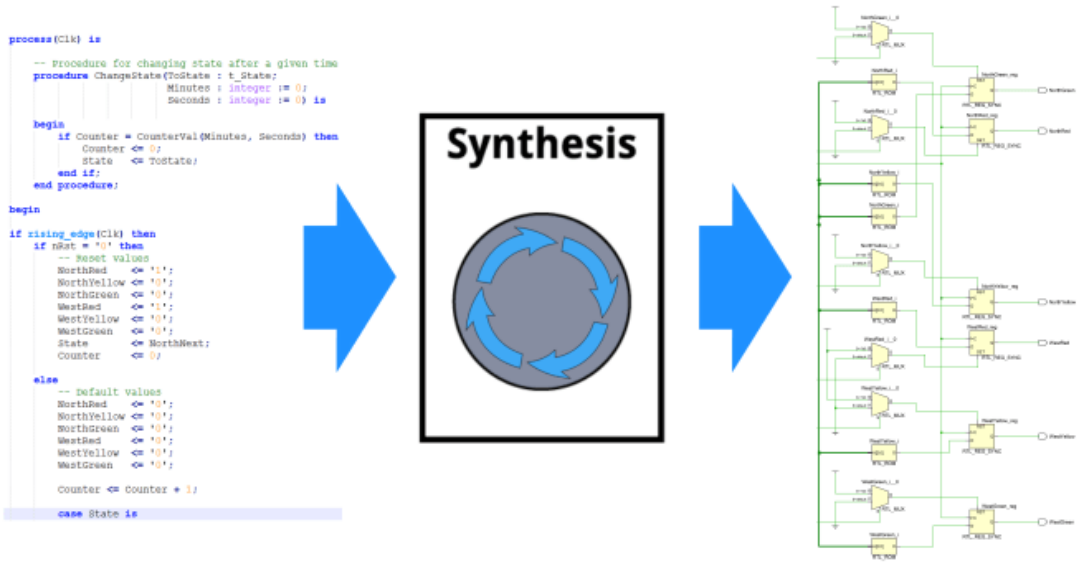

Fig. 1. La salida del sintetizador será una lista de red que a su vez se puede ver en un editor de esquemas.

El ambiente y etapas del proceso de un diseño son factores a tener en cuenta en cualquier diseño basado en VHDL y para esto es posible subdividir el flujo de diseño en dos partes bien diferenciadas:

III-B1. Etapa de Desarrollo: Consiste en varios pasos a tener en cuenta

III-B2. Planteo General del Sistema: Consiste en hacer un Diagrama en Blocks funcional y jerárquico del sistema a desarrollar. Definición de módulos e interfaces generales del sistema.

III-B3. Codificación: Consiste en escribir el código VHDL para todos los componentes planteados, módulos específicos e interfaces. Se trata de un simple discurso de texto, por lo que puede realizarse en cualquier editor conocido por el diseñador.

Sin embargo, los ambientes de diseño incluyen un Editor VHDL especializado, por lo que el desarrollo de esta etapa es más sencilla. Estos editores incluyen características propias del lenguaje, como sangría automática, resaltado de palabras clave, verificación sintáctica, etc.

III-B4. Compilación: En esta etapa el compilador VHDL transforma el programa fuente en objeto, por lo cual analiza la sintaxis de lo escrito y verifica la compatibilidad con cualquier otro módulo ingresados como fuente del presente programa. Junto a esto produce toda la información necesaria para la posterior simulación del proyecto. Nota: A veces, en proyectos complejos conviene realizar compilaciones parciales, ganando así tiempo posterior de desarrollo.

III-B5. Verificación: Esta etapa es muy importante, pues permite establecer si el circuito obtenido funciona como se pretendió al fijar las pautas de diseño. Existen dos características a verificar, y ellas son las Funcional y la Temporal. En el primer caso se analiza el funcionamiento lógico del circuito, sin considerar el tiempo como variable. Aquí los elementos lógicos son considerados como ideales, es decir sin retardos. En la verificación temporal se analizan los resultados del circuito real, considerando retardos estimados, ya que aún no se han seleccionado los circuitos reales de síntesis. De esta manera se verifica la funcionalidad temporal de circuitos combinacionales y especialmente los secuenciales con características de memorización. La verificación funcional y temporal se realiza a través de un proceso complejo conocido como simulación, el cual permite detectar errores en el diseño obtenido, y de esta forma hacer las correcciones adecuadas antes de pasar a la etapa de síntesis.

III-B6. Simulación: Este procedimiento de verificación permite definir entradas y aplicarlas al prototipo de software, analizar el comportamiento de los diversos módulos definidos y observar las salidas. Todo esto sin tener que realizar el prototipo físico con circuitos reales. Para proyectos pequeños se pueden generar entradas y verificar salidas en forma manual; pero en grandes sistemas se crean bancos de prueba con la capacidad de establecer entradas, verificar salidas y realizar las comparativas con los valores esperados. La simulación funcional es completa y precisa, sin embargo la temporal sólo es aproximada pues se basa en valores estimados, y sirve para establecer que vamos en el camino correcto. Esto es así pues es muy dependiente de la síntesis, dónde se establecen los circuitos específicos y es allí dónde salen los verdaderos valores de retardos de acuerdo al circuito, a la tecnología, al layout, tipo de componentes discretos, etc.

III-C. Etapa de Realización

Las características y herramientas son algo diferentes de la anterior. Tiene varios pasos, a saber:

III-C1. Sintesís:: Convierte el modelo descripto por VHDL en un conjunto de primitivasó componentes que se articularán en un circuito real en una tecnología adecuada. Generalmente estas herramientas presentan la posibilidad de establecer filtros, premisasó restricciones específicas del tipo de circuitoó tecnología. Por ejemplo en los circuitos lógicos programables, como PAL, GAL, CPLD, FPGA las herramientas de síntesis pueden generar ecuaciones booleanas; y en el caso de ASICó ASSP generar una lista de compuertas y la malla que fija el cableado correspondiente de interconexión.

III-C2. Ajuste y Enrutamiento: Son herramientas que mapean las ecuacionesó componentes de acuerdo a los recursos disponibles de cada dispositivo. Como en el paso anterior, el diseñador puede especificar restriccionesó asignaciones específicas para lograr el módulo correcto.

III-D. Verificación temporal y total del circuito

Este es la etapa crucial ya que establece la funcionalidad temporal correcta basada en todos los parámetros reales introducidos en el diseño, como circuitos integrados reales, longitud de los buses incorporados, longitud de los alambres, cargas de los elementos físicos, etc. Generalmente se aplican las mismas señalesó bancos de prueba utilizadas en la etapa de desarrollo.

Ya hemos visto que es HDL y VHDL, la teoría y cómo funcionan en la práctica es el momento de explicar que es verilog HDL para dar paso a la comparación y desde el punto de vista de nosotros como estudiantes decir cuál es mejor teniendo en cuenta un criterio en específico.

III-E. Verilog

El Verilog HDL es un estándar IEEE - número 1364. La primera versión del estándar IEEE para Verilog se publicó en 1995. Una versión revisada se publicó en 2001; Esta es la versión utilizada por la mayoría de los usuarios de Verilog. El documento estándar IEEE Verilog se conoce como el Manual de referencia del lenguaje o LRM. Esta es la definición autorizada completa del Verilog HDL.

En 2005 se publicó una nueva revisión del estándar Verilog, aunque tiene poco más en comparación con el estándar 2001. SystemVerilog es un gran conjunto de extensiones de Verilog, y se publicó por primera vez como un estándar IEEE en 2005. Consulte la sección Knowhow correspondiente para obtener más detalles sobre SystemVerilog. IEEE Std 1364 también define la interfaz del lenguaje de programación, o PLI. Esta es una colección de rutinas de software que permiten una interfaz bidireccional entre Verilog y otros lenguajes (generalmente C).

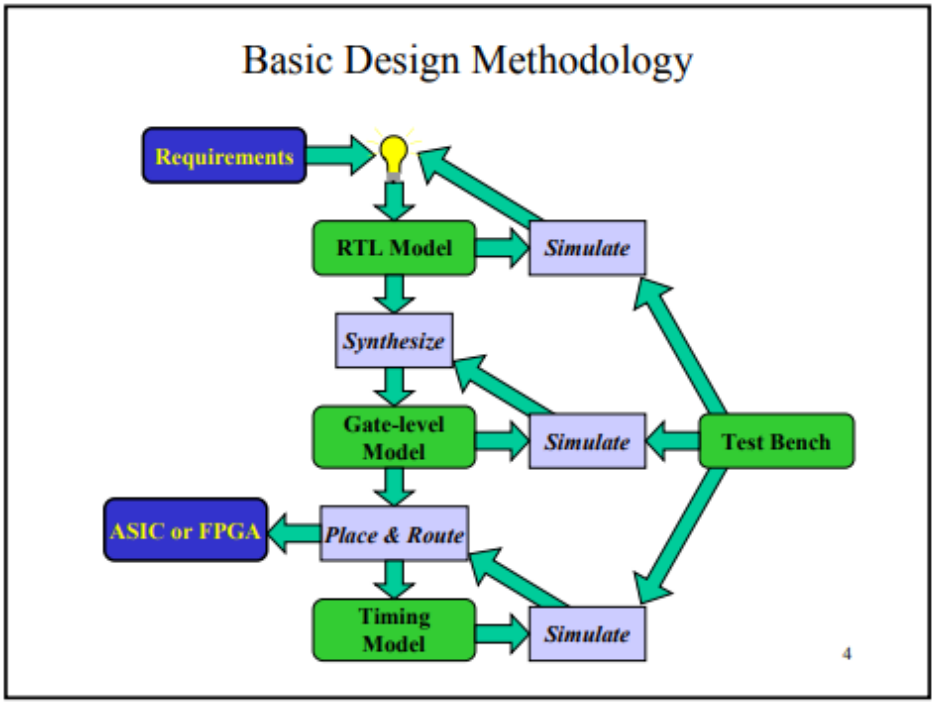

Fig. 2. Diagrama de cómo funciona verilog HDL.

III-E1. ¿Por qué usar Verilog HDL?: Los sistemas digitales son altamente complejos. En el nivel más detallado, los diseños VLSI pueden contener miles de millones de elementos. El lenguaje Verilog proporciona a los diseñadores digitales un medio para describir un sistema digital en una amplia gama de niveles de abstracción. Además, proporciona acceso a herramientas de diseño asistidas por computadora para ayudar en el proceso de diseño en estos niveles. El objetivo de su estudio es crear un diseño de ruta de datos aritméticos por computadora y describir con precisión la funcionalidad del sistema digital.

Verilog proporciona un excelente procedimiento para modelar circuitos destinados a implementaciones de VLSI utilizando programas de lugar y ruta. Sin embargo, también permite a los ingenieros optimizar los circuitos y diseños lógicos para maximizar la velocidad y minimizar el área del chip VLSI. Por lo tanto, conocer a Verilog hace que el diseño de VLSI y sistemas digitales sea más eficiente y es una herramienta útil para todos los ingenieros. Debido a que el lenguaje Verilog es fácil de usar y hay muchos compiladores Verilog disponibles públicamente y comercialmente, es una excelente opción para muchos diseñadores de VLSI.

Aunque hay muchos algoritmos disponibles para el cálculo de la aritmética en los sistemas digitales, se intenta una amplia variedad de diseños completo de muchos de los principales diseños de rutas de datos aritméticos. Varias áreas principales de la aritmética digital, como la aritmética modular del sistema de números residuales, la raíz cuadrada y la raíz cuadrada inversa

y la aritmética de baja potencia. La aplicación de VLSI se utiliza en empresas de productos, empresas de EDA, industria de servicios de diseño y las empresas que lo utilizan son:

- Intel.
- Cadence.
- Synopsys.
- Xilinx.
- Free scale.
- Qualcomm.

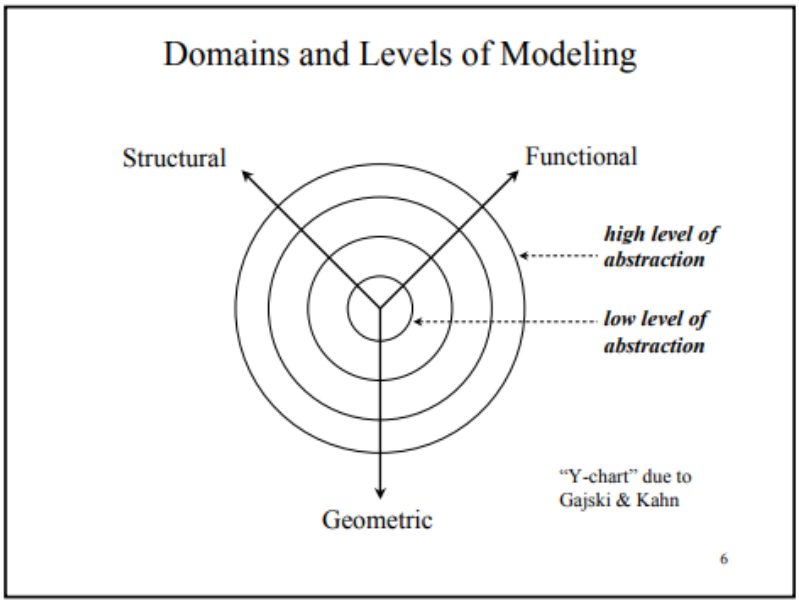

Fig. 3. Abstracion de Verilog HDL

III-F. Comparación y Ejemplo

VHDL y Verilog sirven el mismo propósito, sin embargo su uso depende de las necesidades que tengan los diseñadores de hardware en el momento de codificar sus modelos de bajo nivel. VHDL es un lenguaje fuertemente tipado y determinista, por eso se considera más completo en su estructura, sin embargo al no ser parecido a C, puede no ser amigable con los diseñadores, por otro lado Verilog es débilmente tipado y utiliza una notación mucho más concisa y eficiente, sin mencionar que es parecido a C, esto lo convierte en una herramienta más versátil para diseñadores que no están acostumbrados a VHDL. Cabe destacar que para traducir VHDL a Verilog los diseñadores deben realizar varios pasos extra.

En conclusión, VHDL resulta más eficiente para la máquina mientras que Verilog es más amigable con el programador o diseñador de hardware. Sin embargo ambos cumplen su función y llevan a cabo los mismos objetivos. Programadores y diseñadores de todo el mundo usan ambos estándares, dependiendo de sus necesidades.

El último partido de esta competencia es algo que no se puede decidir fácilmente ... es el desafío de preferencia personal. Esto depende de los lectores. Analice por qué le gusta cada lenguaje de programación y cuál prefiere más. Esta es una buena oportunidad para compartir información sobre lo que los principiantes deben tener en cuenta al comenzar a elegir VHDL o Verilog. Si quieres ver cualquiera de los dos lenguajes en acción, mire algunos de los proyectos de aprendizaje que hay sobre VHDL y Verilog. Por lo general, la elección de uno u otro depende de las herramientas que está utilizando. Algunas de las populares herramientas FPGA funcionan mejor con VHDL. Algunas herramientas ASIC populares funcionan mejor con Verilog. Entonces, lo que es mejor depende de lo que quieras hacer con él. Supongamos que desea construir proyectos pequeños utilizando los FPGA de Altera que son populares en las escuelas de EE. Las herramientas gratuitas admiten ambos HDL. Pero es posible que la comunidad de usuarios esté utilizando principalmente VHDL. Habrá más código de ejemplo, módulos reutilizables, etc. si opta por ese lenguje. Por el contrario, si tiene la intención de trabajar en una gran empresa haciendo un trabajo serio de diseño de chips, casi todos usan Verilog en estos días. Las herramientas de síntesis, simulación

y verificación de servicio pesado están optimizadas para Verilog. Y últimamente, SystemVerilog: extensiones de Verilog para admitir el diseño y la verificación de sistemas de alto nivel.

IV. Conclusiones

- Dependiendo de la necesidad se puede dar una recomendación precisa de cual lenguaje de descripción hardware usar, ya que tanto VHDL como Verilog son apropiados para distintas tareas. VHDL fue creado con un objetivo principal que es resolver necesidades, pasadas, actuales de diseño y otras futuras, tiene mayor amplitud de miras, el resultado de estos objetivos es un lenguaje con sistematización en el código haciéndolo más legible, es un lenguaje rígido. Por otro lado, Verilog nace ante la necesidad de mejorar un flujo de diseño, debido a esto es un lenguaje que cubre todas las necesidades de diseño y simulación, siendo un lenguaje simple, flexible y con bajo nivel de exigencia proporcionando facilidad de aprendizaje y teniendo en cuenta ciertas limitaciones que posee.

- Teniendo en cuenta el tipo de datos cabe destacar que en VHDL se permite el uso de tipos de datos definidos por el lenguaje y por el usuario, necesitando funciones para convertir objetos de un tipo a otro, gracias a esto los modelos son más fáciles de leer y escribir, a diferencia de esto, los tipos de datos en Verilog son simples y están orientados al modelamiento en hardware.

- Hacemos evidente que VHDL no es una abreviatura de Verilog HDL, Verilog y VHDL son dos HDL diferentes. Sin embargo, tienen más similitudes que diferencias.

- La elección de alguno de los dos depende de en qué se esté trabajando, lo que más recomienda la comunidad es aprender ambos para poder hacer una correcta elección dependiendo de la tarea que se esté realizando.

References

[1] Huang, Chi-Lai; Su, S.Y.H. "Approaches for Computer-Aided Logic System Design Using Hardware Description Language". Proceedings of International Computer Symposium 1980, Taipei, Taiwan, December 1980.

[2] Lenguaje descripción de hardware: VHDL Disponible en https://vdocuments.site/vhdl5571fdfa49795991699a6387.html

[3] VHDL Disponible en https://es.scribd.com/document/324249101/VHDL

[4] Basico VHDL Disponible en https://es.scribd.com/document/254205948/Basico-VHDL

[5] Gordon, M. (1995, June). The semantic challenge of Verilog HDL. In Proceedings of tenth Annual IEEE Symposium on Logic in Computer Science (pp. 136-145). IEEE.

[6] Stine, J. E. (2012). Digital computer arithmetic datapath design using verilog HDL. Springer Science Business Media.

[7] Christen, E., Bakalar, K. (1999). VHDL-AMS-a hardware description language for analog and mixed-signal applications. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 46(10), 1263-1272.