ORGANIZACIÓN Y ESTRUCTURA DEL LENGUAJE VHDL

Felipe Alvarado Galicia

UNIVERSIDAD POLITÉCNICA
DE LA ZONA METROPOLITANA DE GUADALAJARA

11 DE FEBRERO DE 2020

UPZMG
Prof: Carlos Enrique Moran Garabito

introducción

En la actualidad, el lenguaje de programación VHDL (Hardware Description Language) constituye una de las herramientas de programación con mayor uso en el ambiente industrial y en el ámbito universitario, debido a la versatilidad con la cual se pueden describir y sintetizar circuitos y sistemas digitales en la búsqueda de soluciones de aplicación inmediata. El uso correcto del lenguaje hace obsoleto el diseño tradicional, que organiza bloques lógicos de baja y mediana escala de integración, “compuertas, contadores, registros, decodificadores, etcétera”. En otras palabras, para qué organizar diversas estructuras lógicas en una determinada solución, si esta se puede crear en una sola entidad, la cual no solo proporciona una reducción considerable de espacio físico, sino que además también produce un resultado más directo y menos susceptible a los errores derivados de la conexión entre varios componentes.

En el caso de que la solución requiera de un mayor número de componentes lógicos o “pins” de salida, pueden utilizarse circuitos CPLD (por sus siglas en inglés; Dispositivos Lógicos Programables Complejos), los cuales también se consiguen de forma individual o como parte de un sistema de desarrollo. Por otro lado, si el proyecto lo amerita, es posible utilizar los FPGA (por sus siglas en inglés; Arreglo de Compuertas Programables en Campo), recomendados para ser empleados y programados dentro de un sistema de desarrollo y en aplicaciones con tendencia hacia el desarrollo de sistemas completos dentro de un solo circuito integrado SOC (System On Chip). Las opciones que brinda el hardware para la integración de la aplicación permiten introducir y utilizar el lenguaje en diversas materias, cuyo nivel de profundidad y diseño depende de los bloques lógicos por emplear; de esta manera, el diseño lógico y el de sistemas digitales, la robótica, la arquitectura de computadoras y la algorítmica, entre otras, son algunas de las áreas donde VHDL presenta una alternativa de diseño. Por ejemplo, en un curso de diseño lógico, quizá lo más recomendable sea utilizar el lenguaje y programar en dispositivos GAL; sin embargo, esta no es una condición necesaria e indispensable, pues depende en

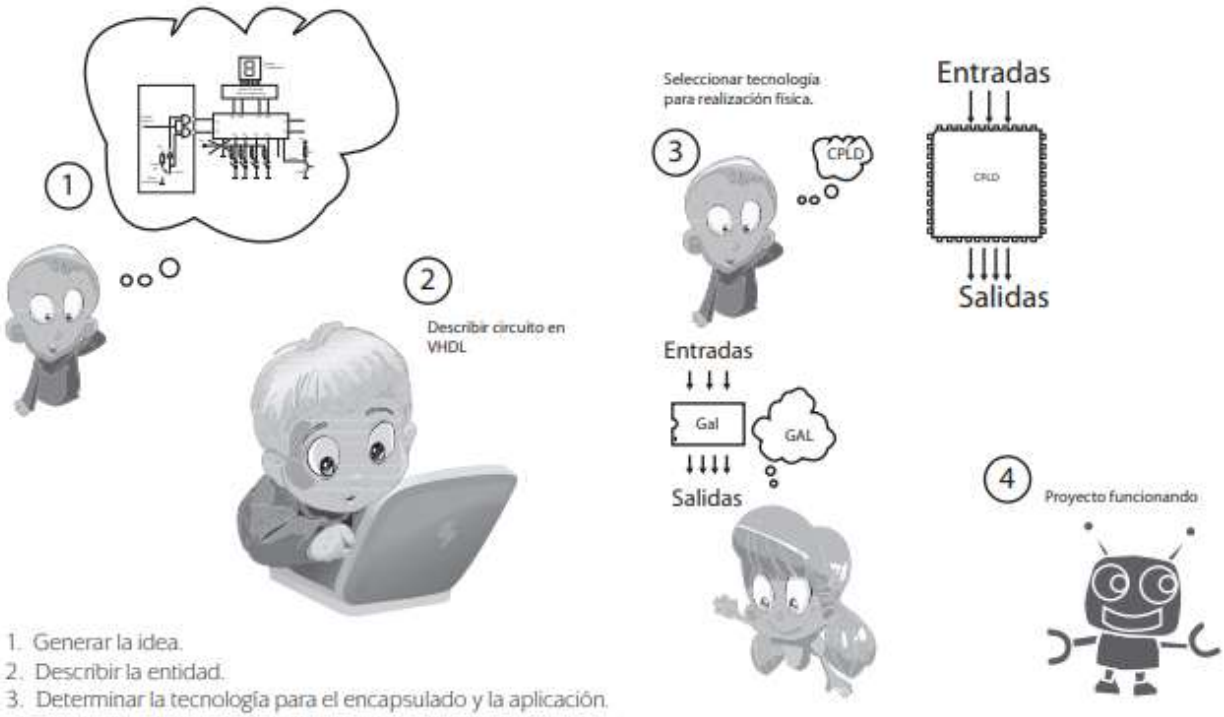
gran medida de la economía personal o institucional. Como se verá adelante, los cambios en la programación son irrelevantes respecto de la utilización de diversas tecnologías de fabricación de los chips. Por tanto, quizás los aspectos que deban cuidarse con más atención son la conceptualización y el entendimiento de la arquitectura interna del circuito; no es lo mismo programar en Arreglos Lógicos Genéricos que en FPGA, cuyas arquitecturas internas son muy diferentes.

VHDL su estructura.

El lenguaje de descripción en hardware VHDL se estructura en módulos o unidades funcionales, identificados mediante una palabra reservada y particular de este lenguaje (véase figura 1.2). En tanto, a su vez, cada módulo tiene una secuencia de instrucciones o sentencias, las cuales, en conjunto con las declaraciones de las unidades involucradas en el programa, permiten la descripción, la comprensión, la evaluación y la solución de un sistema digital. Al interior de la estructura de un programa, las unidades Entidad (Entity) y Arquitectura (Architecture) —en conjunto— forman la columna vertebral de este lenguaje. Por su parte, los módulos restantes, no necesariamente utilizados en la búsqueda de una solución, sirven entre otras cosas para optimizar y generalizar la aplicación en futuros desarrollos, como se verá cuando la ocasión se presente. Sin embargo, en este momento nuestra atención se centra en describir la función de la entidad y la arquitectura.

Entidad (entity) Una entidad básicamente representa la caracterización del dispositivo físico; es decir, exhibe las entradas y las salidas del circuito (llamados pins) que el diseñador ha considerado pertinentes para integrar su idea o aplicación; en la figura 1.3, se puede observar con detalle la secuencia de desarrollo. Con base en esta idea, una entity —por la palabra reservada del programa— constituye un bloque de diseño que puede ser analizado y programado como un elemento individual, ya sea como una compuerta, un sumador o un decodificador, entre otros, incluso ser considerado como un sistema a través de su relación entre entradas y

salidas, las cuales representan los puntos de observación o de conexión a elementos periféricos propios de la aplicación.



1. Generar la idea.
2. Describir la entidad.
3. Determinar la tecnología para el encapsulado y la aplicación.

Figura 1.3 Representación del proceso de aplicación y desarrollo de la entidad.

Puertos de entrada–salida

Cada una de las señales de entrada y salida en una entidad es referida como un puerto, el cual es equivalente a una terminal (pin) de un símbolo esquemático. Todos los puertos que son declarados deben tener un nombre, un modo y un tipo de dato. El nombre es utilizado como una forma de llamar al puerto; el modo permite definir la dirección que tomará la información, mientras que el tipo precisa qué clase de información se transmitirá a través del puerto. Por ejemplo, en el caso de los puertos de la entidad representada en la figura 1.5, aquellos que son de entrada están indicados por las variables a y b; mientras que el puerto de salida se representa por la variable c. Por otra parte, el tipo de dato será tratado más adelante.

}

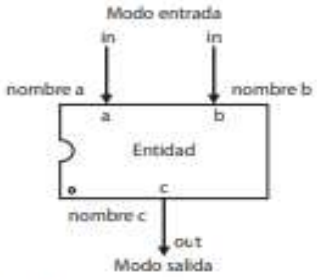


Figura 1.5 Puertos de entrada-salida.

Modos Como se mencionó antes, un modo permite definir la dirección hacia donde el dato es transferido. Un modo puede tener uno de cuatro valores: in (entrada), out (salida), inout (entrada/salida) y buffer (véase figura 1.6).

- Modo in. Se refiere a las señales de entrada a la entidad. El modo in es solo unidireccional y únicamente permite el flujo de datos hacia dentro de la en tidad.
- Modo out. Indica las señales de salida de la entidad.
- Modo inout. Permite declarar a un puerto de forma bidireccional, es decir como de entrada/salida, además hace posible la retroalimentación de señales dentro o fuera de la entidad.
- Modo buffer. Permite realizar retroalimentaciones dentro de la entidad; pero, a diferencia del modo inout, el puerto declarado se comporta como una terminal exclusiva de salida.

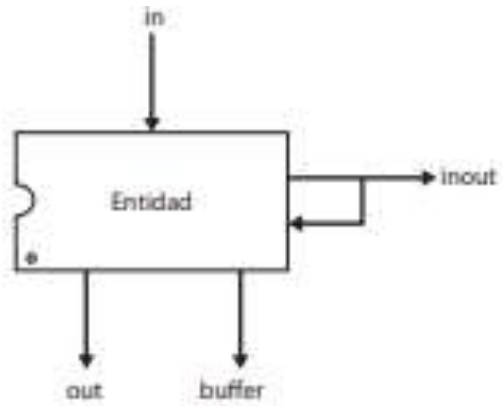


Figura 1.6 Tipos de modos.

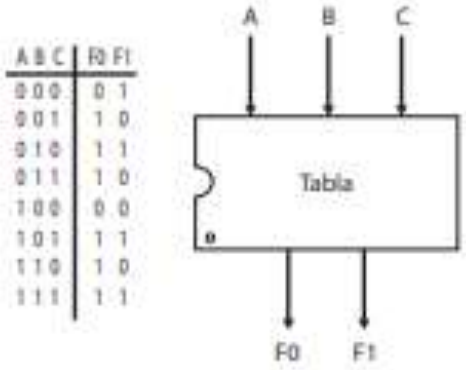
Tipos de datos

Los tipos son los valores (datos) que el diseñador establece para los puertos de entrada y salida dentro de una entidad, y que son asignados de acuerdo con las características de un diseño en particular. Algunos de los tipos más utilizados son el bit, el cual tiene valores de 0 y 1 lógico; el tipo boolean (booleano) define valores de verdadero o falso en una expresión; el bit_vector (vectores de bits), el cual representa un conjunto de bits para cada variable de entrada y/o salida, y el tipo integer (entero), que representa a un número entero. Los anteriores son solo

algunos de los tipos que maneja VHDL, aunque no son los únicos.1 Los tipos de datos y su uso se introducirán conforme se vayan requiriendo y empleando a lo largo del texto.

Declaración de entidades

La declaración de una entidad consiste en describir las entradas y las salidas de un circuito identificado como Entity (entidad); en otras palabras, la declaración señala las terminales o los pines de entrada y de salida con los que cuenta el circuito. Por ejemplo, considérese la tabla de verdad que se muestra en la figura 1.7; como se puede observar, esta tiene tres entradas, A, B y C, y dos salidas, F0 y F1. En este caso, a la entidad se le ha identificado con el nombre de TABLA, tal y como se muestra.



Identificadores

Los identificadores son simplemente los nombres o las etiquetas que se usan para hacer referencia a variables, constantes, señales, procesos, etc. Estos identificadores pueden ser números, letras del alfabeto y/o guiones bajos que separan caracteres. Es importante resaltar que no existe una restricción en cuanto a su longitud. Todos los identificadores deben seguir ciertas especificaciones o reglas para que puedan ser compilados sin errores (véase tabla 1.1).

Tabla 1.1

Regla	Incorrecto	Correcto
El primer carácter siempre debe ser una letra mayúscula o una minúscula.	4suma	Suma4
El segundo carácter no puede ser un guión bajo.	S_4bits	S4_bits
No se permite el uso de dos guiones juntos.	Resta__4	Resta_4_
Un identificador no puede utilizar símbolos.	Clear#8	Clear_8

Es importante destacar que VHDL cuenta con una lista de palabras reservadas, las cuales no pueden utilizarse como identificadores (véase apéndice E).

ARQUITECTURA

Una arquitectura define el algoritmo o la estructura de solución de una entidad, en esta se describen las instrucciones o los procedimientos “programa” que deben llevarse a cabo para obtener la solución deseada por el diseñador. La gran ventaja que presenta VHDL con respecto a los compiladores tradicionales de diseño PALASM, OPAL, PLP, ABEL y CUPL (véase apéndice A), entre otros disponibles para la programación de Dispositivos Lógicos Programables, es la extensa variedad de formatos con los que cuenta y que utiliza en la descripción de una entidad. Es decir, en VHDL es posible describir en diferentes niveles de abstracción, que van desde el uso de ecuaciones y estructuras lógicas, la descripción mediante la simbología de la transferencia de registros RTL, hasta la simplicidad que puede representar una caja negra —sistema—, cuya interpretación “función de transferencia”, la relación entre entradas y salidas, describe el funcionamiento de la entidad. No obstante, en VHDL no existe de manera formal un procedimiento de diseño.

Como ya se mencionó, la versatilidad en la abstracción y la descripción de una entidad permiten al usuario la tarea de planear la estrategia por utilizar para una solución en particular. Sin duda, la experiencia en programación puede ser un factor importante de diseño, sin embargo, VHDL posee una estructura de programación tan amigable que es muy fácil de entender, aun por personas que no han tenido relación alguna con lenguajes de programación. De manera general y con base en cómo se presenta la entidad (entity), a primera vista, en VHDL se pueden distinguir los siguientes estilos de programación.

- Estilo por flujo de datos
- Estilo funcional
- Estilo estructural

Operadores lógicos

Los operadores lógicos que se utilizan en la descripción con ecuaciones booleanas y que están definidos dentro de los diferentes tipos de datos —bit, boolean o std_logic— son los operadores: and, or, nand, xor, xnor y not. Las operaciones que se efectúen entre estos (excepto not) deben realizarse con datos que tengan la misma longitud o palabra de bits. Los operadores lógicos presentan el siguiente orden y prioridad al momento de ser compilados:

1. Expresiones entre paréntesis.
2. Complementos.
3. Función and.
4. Función or.

Las operaciones xor y xnor son transparentes al compilador, el cual las interpreta mediante la suma de productos correspondiente a su función.

Descripción funcional

En una descripción funcional lo más importante es el conocimiento global del sistema, razón por la cual las entidades diseñadas bajo este estilo son programadas como una caja negra; es decir, no importa la organización o la estructura interna de la entidad, solo se requiere que el programador conozca lo que espera obtener en la salida y la forma en que operan las pins de entrada. Por ejemplo, considérese una entidad en la cual existe una salida C que adopta el valor C=1 cuando las señales de entrada, A y B, son iguales; en caso contrario, la salida C toma el valor de cero C=0. Entonces, la síntesis del problema sería la siguiente:

```
Si A=B entonces C=1, sino C=0
if A=B   then   C=1, else C=0
```

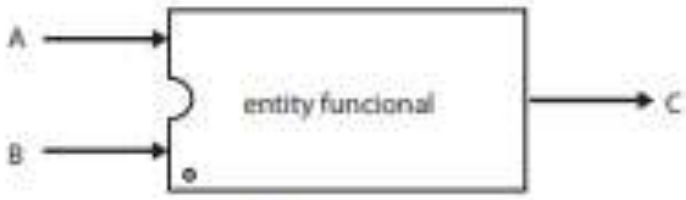


Figura 1.28

Descripción estructural

Como su nombre lo indica, una descripción estructural basa su comportamiento en modelos lógicos ya establecidos (compuertas, sumadores, contadores, proyectos especiales, etc.). Es importante destacar que estas estructuras pueden ser diseñadas por el usuario y guardadas para su posterior utilización o extraídas de los paquetes contenidos en las librerías de diseño del software que se esté utilizando, como se verá más adelante. En la figura 1.31 se muestra una representación esquemática de una entidad que cuenta con dos variables de entrada de 2 bits (a0, a1 y b0, b1). En su construcción se han empleado dos compuertas nor exclusivas y una compuerta and.

Figura 1.31 Estilo estructural.

BIBLIOGRAFIA

https://editorialpatria.com.mx/pdffiles/9786074386219.pdf