UNIDAD 4

SISTEMAS DIGITALES

TEMA 1:
Fundamentos FPGA

FR0018/ v1.0 / 18-11-2019

UNEMI ONLINE

ÍNDICE

1. Unidad 4: Procesamiento digital de datos

Tema 1: Fundamentos de FPGA

Objetivos:

Al finalizar el tema 7 el estudiante aprenderá a:

- Describir la arquitectura de una FPGA.
- Explicar el funcionamiento básico de una tabla de consulta LUT.
- Explicar las funciones integradas que tienen el FPGA.
- Representar señales en VHDL, escribir código simple VHDL.

Antes de hablar de FPGA y su arquitectura revisaremos un poco del concepto de lógica programable que es el objetivo final de aprendizaje, de poder programar los FPGA mediante lenguajes de programación a nivel de hardware y construir diversas aplicaciones.

“La lógica programable requiere tanto de hardware como de software. Los dispositivos lógicos programables pueden programarse para que el fabricante o el usuario pueda llevar a cabo funciones lógicas específicas. Una ventaja de la lógica programable frente a la lógica fija es que los dispositivos utilizan menos espacio de la tarjeta de circuito impreso para una cantidad equivalente de lógica. Otra ventaja es que, con la lógica programable, los diseños se pueden modificar fácilmente sin tener que recablear o reemplazar componentes.” (Floyd, 2006, pág. 26)

“Existen muchos tipos de dispositivos lógicos programables, desde pequeños dispositivos que pueden reemplazar a algunos de los dispositivos de función fija hasta complejos dispositivos de alta densidad que pueden reemplazar a miles de dispositivos de función fija. Las dos principales categorías de los dispositivos lógicos programables de usuario son los PLD (Programmable Logic Device, dispositivo lógico programable) y las FPGA (Field Programmable Gate Array, matrices de puertas programable por campo), que se muestran en la figura 1. Los PLD pueden ser SPLD (Simple PLD, PLD simple) o CPLD (Complex PLD, PLD complejo).” (Floyd, 2006, pág. 26)

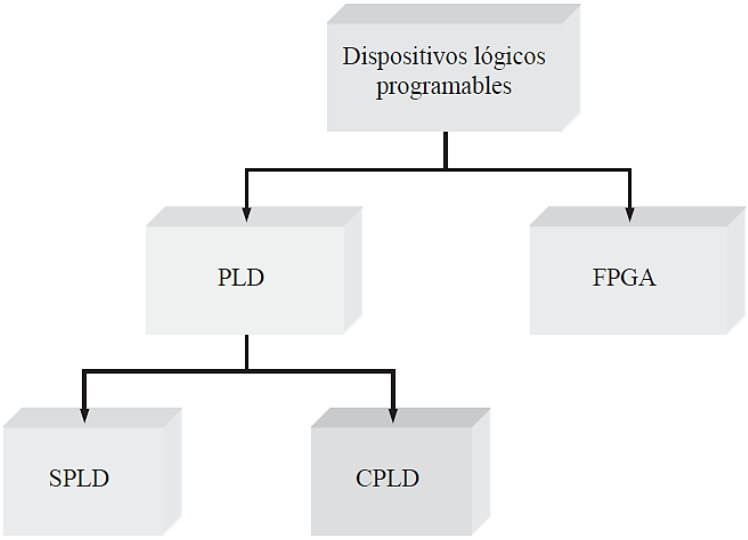


FIGURA 1: Clasificación de dispositivos lógicos programables
Fuente: "Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pag 26

2. Información de los subtemas

2.1 Subtema 1: Arquitectura

"La FPGA (Field Programmable Gate Array) tiene una estructura interna (arquitectura) diferente como se ilustra en la figura 2. Los tres elementos básicos en una FPGA son el bloque lógico configurable (CLB, Configurable Logic Block), las interconexiones y los bloques de entrada/salida (E/S). Los bloques CLB de una FPGA no son tan complejos como los bloques LAB o FB de un CPLD, pero suele haber muchos más bloques CLB. Cuando los bloques CLB son relativamente simples, decimos que la arquitectura FPGA es de granularidad fina. Cuando los bloques CLB son de mayor tamaño y más complejos, decimos que la arquitectura es de granularidad gruesa." (Floyd, 2006, pág. 27)

"Los bloques de E/S situados alrededor del perímetro de la estructura proporcionan un acceso de entrada/salida o bidireccional, individualmente seleccionable, hacia el mundo exterior. La matriz distribuida de interconexiones programables permite interconectar los bloques CLB entre sí y conectarlos a las entradas y a las salidas. Los dispositivos FPGA de gran tamaño pueden tener decenas de miles de bloques CLB, además de memoria y otros recursos." (Floyd, 2006, pág. 27)

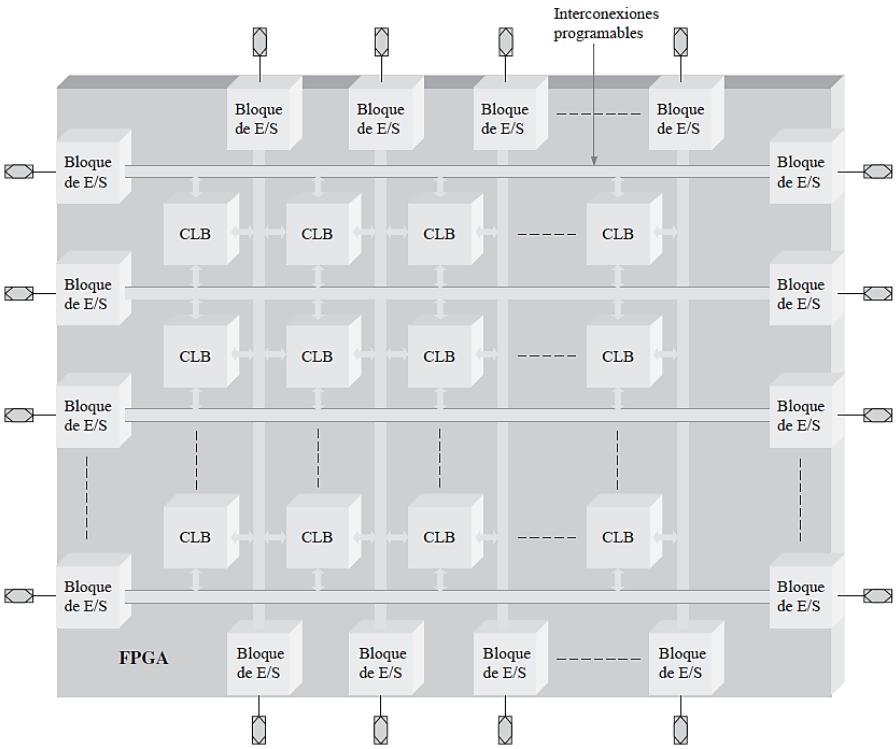


FIGURA 2: Estructura interna de la FPGA
Fuente: "Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pág. 707

En la Figura 3 se muestra un encapsulado BGA (*Ball-Grid Array*) típico para FPGA. Estos tipos de encapsulados pueden tener unos 1.000 pines de entrada y salida.

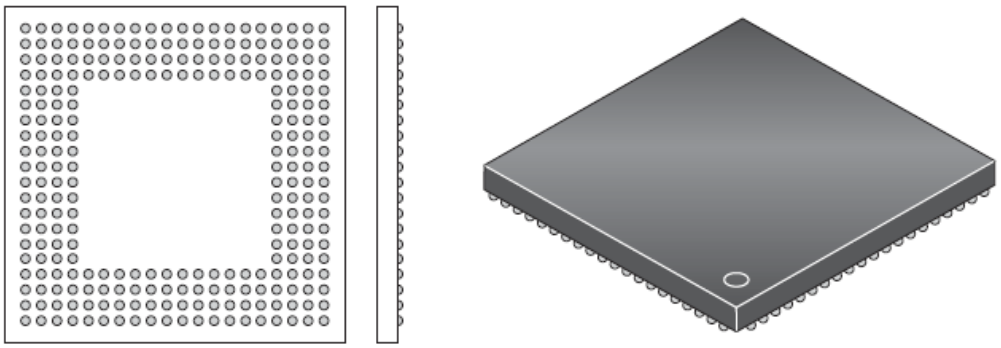


FIGURA 3: Encapsulado BGA
Fuente: "Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pág. 29

Las FPGA son reprogramables y utilizan tecnología de proceso SRAM o de antifusibles para implementar las conexiones programables. Las densidades pueden ir desde los centenares de módulos lógicos hasta aproximadamente 180.000 módulos lógicos, en encapsulados de hasta más de 1000 pines. Las tensiones de alimentación continua están comprendidas habitualmente entre 1,2 V y 2,5 V, dependiendo del dispositivo específico.

Bloques lógicos configurables

"Normalmente, un bloque lógico de FPGA está compuesto por varios módulos lógicos más pequeños, que son las unidades componentes básicas y que en cierto modo resultan análogos a las macroceldas de un CPLD.
La Figura 4 muestra los bloques lógicos configurables fundamentales (CLB) dentro de la matriz global de interconexiones programables dispuestas en filas/columnas y que se utilizan para conectar entre sí los bloques lógicos. Cada ***CLB*** está formado por múltiples módulos lógicos más pequeños y por una serie de interconexiones programables locales que se emplean para conectar entre sí los módulos lógicos que componen el CLB." (Floyd, 2006, pág. 30)

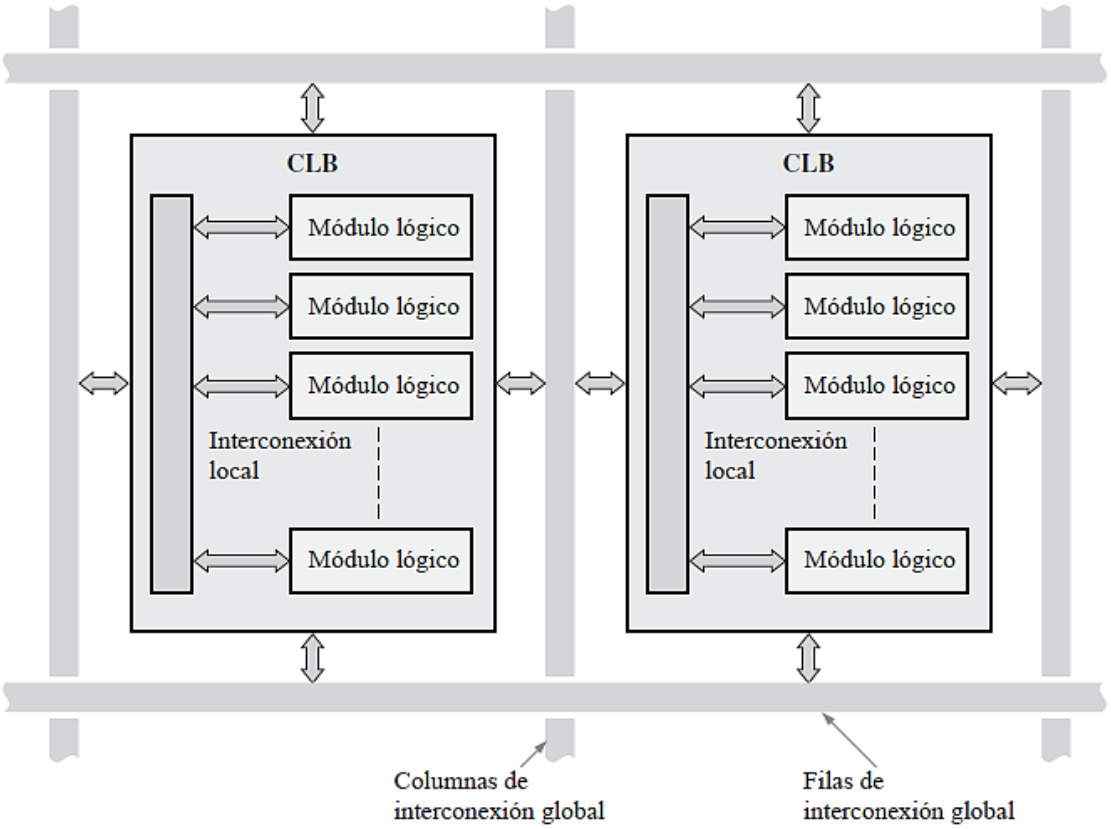


FIGURA 4: Bloques lógicos configurables
Fuente: "Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pag 708

Módulos lógicos. "Un módulo lógico de un bloque lógico de una FPGA puede configurarse para implementar lógica combinacional, lógica registrada o una combinación de ambas. Se emplea un flip-flop que forma parte de la lógica asociada para implementar lógica registrada. En la Figura 5 se muestra un diagrama de bloques de un módulo lógico típico basado en LUT. Como ya sabemos, una LUT (*Look-Up Table*) es un tipo de memoria programable que se utiliza para generar funciones lógicas combinacionales suma de productos. La LUT realiza, esencialmente, el mismo trabajo que una PAL o una PLA." (Floyd, 2006, pág. 686)

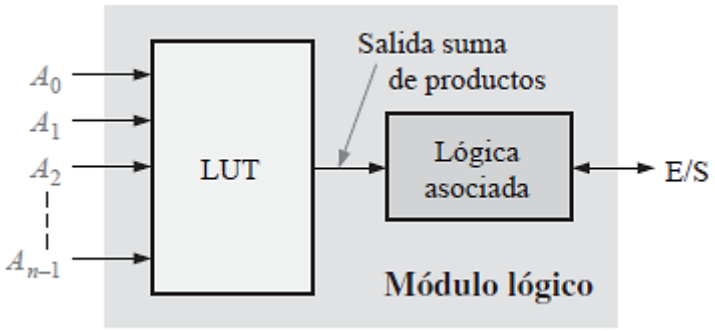


FIGURA 5: Módulo lógico basado en LUT
Fuente: dx"Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pag 708

Generalmente, la organización de una LUT consiste en una serie de 2*n* celdas de memoria, siendo *n* el número de variables de entrada. Por ejemplo, mediante tres entradas se pueden seleccionar hasta ocho celdas de memoria, por lo que una LUT con tres variables de entrada permite generar una suma de productos con hasta ocho términos.

Dentro de las celdas de memoria LUT puede programarse un patrón de 1s y 0s, como se ilustra en la Figura 6 para una función suma de productos especificada. Cada 1 significa que el término producto asociado aparecerá en la salida suma de producto mientras que un 0 significa que dicho término producto asociado no aparecerá en la salida suma de productos. La expresión de la salida suma de productos resultante es:

$$\overline{A}_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1A_0 + A_2\overline{A}_1A_0 + A_2A_1A_0$$

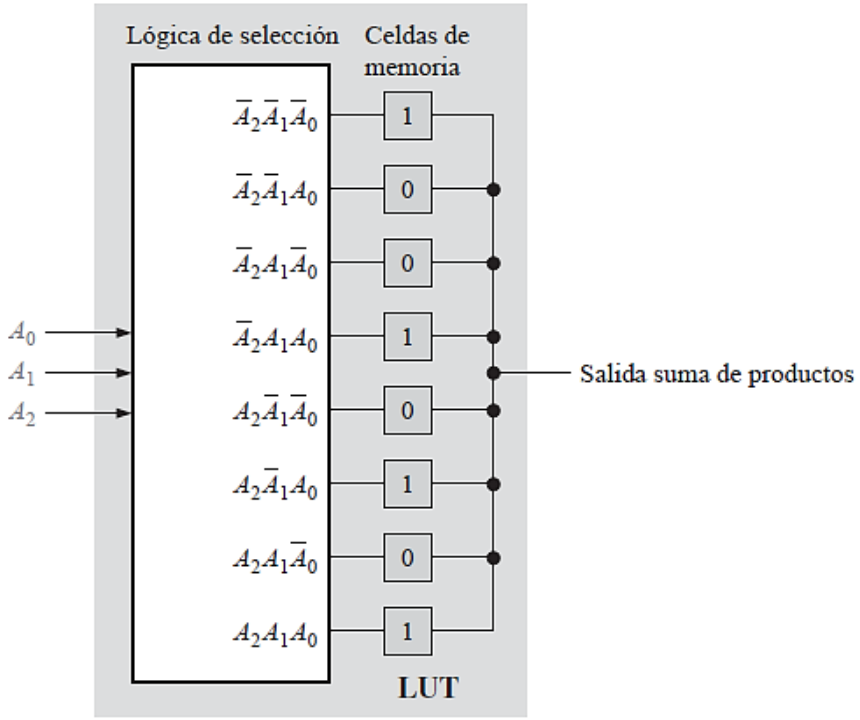


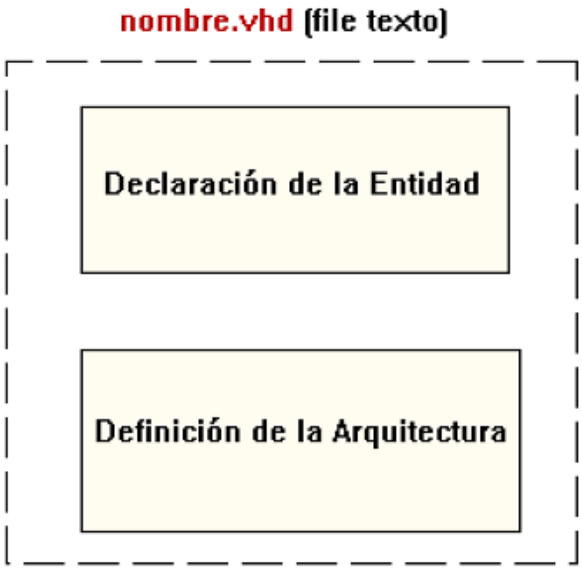
FIGURA 6: Suma de productos en LUT
Fuente: "Fundamentos de Sistemas Digitales, Floyd, 9na edición" Pag 709

2.2 Subtema 2: Lenguaje VHDL

Un lenguaje de descripción de hardware (HDL) es una herramienta formal que sirve para describir el comportamiento y la arquitectura de un circuito o sistema electrónico utilizando diferentes niveles de abstracción y en muchos casos de modo jerárquico.
El VHDL Very High Speed Integrated Circuit Hardware Desing/Description Language. surge de Intermetrics, Texas Instruments e IBM. Inicialmente fue llamado VHD2L. Es un lenguaje amplio y prolijo; tanto así que algunos le han adjudicado un segundo significado: Very Hard Description Language.
A mediados de los años 80 y desde 1987 se convirtió en un estándar de la IEEE.
Utilizando VHDL se puede diseñar, simular y sintetizar desde un CLC sencillo, hasta sistemas digitales de gran complejidad tal como un microprocesador.

El Lenguaje VHDL ha ganado mucha popularidad entre los diseñadores de sistemas digitales en los últimos años gracias a la aparición en el mercado de variadas herramientas de diseño que utilizan el VHDL.
Numerosas firmas han adoptado el lenguaje VHDL como herramienta básica y lo han incorporado a sus productos. Entre las mismas están: Altera [Quartus Prime], Xilinx [Vivado]

Dentro de la estructura, VHDL está formado por la pareja Entidad-Arquitectura (Entity-Architecture). A la pareja entidad-arquitectura se le llama modelo en VHDL. Un fichero nombre.vhd puede contener uno o varios modelos.



Por lo tanto, un diseño en VHDL será un fichero texto **(no un dibujo)**, escrito en cualquier editor, donde en la primera parte aparece la declaración de la entidad (entity) y a continuación la definición de la arquitectura (architecture).

Entidad

"En la declaración (Carpio, 2000) de entidades, se definen las entradas, salidas y tamaño de un circuito, explicitando cuáles son, de qué tamaño (de 0 a n bits), **modo** (entrada, salida, ...) y **tipo** (integer, bit,...) .

• Las entidades pueden definir bien las entradas y salidas de un diseño más grande o las entradas y salidas de un chip directamente.

• La declaración de entidades es análoga al símbolo esquemático de lo que queremos implementar, el cual describe las conexiones de un componente al resto del proyecto, es decir, si hay una entrada o puerto de 8 bits, o dos salidas o puertos de 4 bits, etc." (Carpio, 2000, pág. 40)

```
Entity nombre_entidad is

Port ( Nombre de señal : modo tipo;
       .
       .
       .
Nombre de señal : modo tipo);

End nombre_entidad;
```

Entity. Cabecera del programa

"Port. Se indica que a continuación viene los puertos (o grupos señales) de entrada y/o salida. Aquí se declaran las entradas y/o salidas con la sintaxis que se verá a continuación. Las líneas empezadas por dos guiones son ignoradas por el compilador. Así mismo, recordamos que el compilador no distingue las mayúsculas de las minúsculas.

End. Se indica que se ha acabado la declaración de puertos de entrada y/o salida, y que se ha acabado la entidad" (Carpio, 2000, pág. 41)

En VHDL las señales puede ser de cuatro modos diferentes:

Modo	Descripción
IN	señales que sólo entran en la entidad
OUT	señales que salen de la entidad
INOUT	señales bidireccionales
BUFFER	señales que además de salir de la entidad, pueden usarse como señales realimentadas

Los tipos de señales posibles se muestran a continuación:

Tipo	Características
Bit	En este tipo las señales sólo toman los valores '0' y '1'.
Boolean	En este tipo las señales sólo toman los valores True y False.
Std_logic	En este tipo las señales toman nueve posibles valores: 'U'= no inicializado, 'X'= desconocido (un 0 ó un 1 fuerte) '0'= Un 0 fuerte, '1'= Un 1 fuerte, 'Z'= alta impedancia, 'W'= desconocido (un 0 ó un 1 débil), 'L'= un 0 débil, 'H'= un 1 débil, '-' = combinación opcional (don'care). En síntesis sólo se utilizan cuatro: '1', '0', '-' y 'Z'.
Integer	En este tipo las señales toman valores enteros, aquí los 1 y 0 van sin ''.
Bit_vector	En este tipo las señales son una cadena de 1 y 0 . Una cadena se escribe entre comillas. Por ejemplo: "1010".
Std_logic_vector	En este tipo las señales son una cadena de valores permisibles para el tipo std_logic. Por ejemplo: " 1 - 0 Z "

Veremos ahora un ejemplo de designación de puertos:

“nombre_variable: modo tipo; Forma genérica de designar un puerto

Puertoa: in bit; El primer puerto es un bit de entrada, y su nombre es "puertoa"

Puertob: in bit_vector(0 to 7); El segundo puerto es un vector de 8 bits de entrada siendo el MSB el puertob(0) y el LSB el puertob(7)

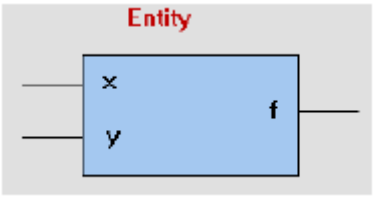
Puertoc: out bit_vector(3 downto 0); El tercer puerto es un vector de 4 bits de salida siendo el MSB el puertoc(3) y el LSB el puertoc(0)

Puertod: buffer bit; El cuarto puerto es un buffer de un solo bit, cuyo nombre es "puertod"

Puertoe: inout std_logic; El quinto puerto es una entrada/salida del tipo estándar logic de un solo bit” (Carpio, 2000, pág. 83)

Ahora que ya hemos revisado como declarar entradas y los tipos disponibles, vamos a realizar a modo de ejemplo una declaración de entidad.

Escriba la declaración de identidad para un circuito digital con dos entradas (a, b) y una salida f, como el mostrado:



Solución:

```
ENTITY ejemplo1 IS
PORT ( x, y : IN std_logic ;
f : OUT std_logic ) ;
END ejemplo1 ;
```

Arquitectura

“La arquitectura indica el tipo de procesado que se realiza con la información correspondiente a las señales de entrada, (declarados previamente en la entidad) para llegar a tener los puertos de salida (también declarados en la entidad). En la declaración de arquitecturas es donde reside todo el funcionamiento de un circuito, ya que es ahí donde se indica que hacer con cada entrada, para obtener la salida. Si la entidad es vista como una "caja negra", para la cual lo único importante son las entradas y las salidas, entonces, la arquitectura es el conjunto de detalle interiores de la caja negra.” (Carpio, 2000, pág. 102)

ARCHITECTURE nombre_arq OF nombre_ent IS	Cabecera de la arquitectura
(Declaración de señales) (Declaración de tipos) (Declaración de variables) (Declaración de constantes) (Declaración de componentes) (Especificación de atributos)	Declaraciones de apoyo que se verán más adelante.
BEGIN	Se da comienzo al programa.
{Instanciación de componentes} {Enunciado concurrente} {Enunciado secuencial (Proceso)}	Conjunto de sentencias, bucles, procesos, funciones, que dan operatividad al programa.
END nombre_arq;	

Fin del programa.

La estructura más común de un programa VHDL consiste en tres partes:

1. Declaración de las bibliotecas utilizadas (library).
2. Una entidad (entity)
3. Una arquitectura (arquitecture)

Además, los comentarios se inician con dos guiones seguidos y terminan al final de la línea.

"El lenguaje VHDL se organiza y trabaja con bibliotecas. Una biblioteca VHDL es un lugar donde el compilador VHDL almacena información referente a un proyecto de diseño particular, incluyendo documentos intermedios que son usados en el análisis, la simulación y la síntesis." (Carpio, 2000, pág. 102)

Una biblioteca estándar muy usada es la biblioteca IEEE. La biblioteca IEEE es solo de lectura y es del tipo global pues permite ser usada por varios programas a la vez. El diseñador no puede interactuar con ella ni modificar su contenido. Para usarla se debe escribir al inicio del programa:

library ieee;

Uno de los paquetes estándares más utilizados es el std_logic_1164, en el cual están contenidas todas las definiciones y tipos más usados en el diseño digital.

Para usar este paquete debe escribirse antes de la declaración de entity lo siguiente:

```
library ieee;
use ieee.std_logic_1164.all ;
```

```
library ieee;
use ieee.std_logic_1164.all ;
```

std_logic_1164 es un paquete contenido en la biblioteca estándar IEEE declarada con la cláusula library.

El sufijo all da acceso a todas las definiciones y objetos contenidos en dicho paquete.

Veremos ahora unos cuantos operadores básicos con la nomenclatura VHDL.

lógicos: AND, OR, XOR, NOT, NAND, NOR y XNOR

aritméticos : +, -, * (multiplicación sólo por 2)

de relación: =, /=, <, >, <=, >=

de concatenación: &

Operadores	Definidos para los tipos:
Lógicos	Bit, Boolean, Bit_vector, std_logic y std_logic_vector
De relación	Integer, Bit y Bit_vector
Aritméticos	Integer
Concatenación	Bit, Bit_vector y para las cadenas.

Declaraciones concurrentes

La descripción por flujo de datos utiliza en su escritura ecuaciones simples de asignación de señales, declaraciones de asignación condicional de señales (usando when-else) o de asignación de señales seleccionadas (usando with-select-when).

```
-- asignación de señales
Nombre de la señal <= expresión;
```

```
-- asignación condicional de señales
Nombre de la señal <= expresión
when expresión booleana else

Nombre de la señal <= expresión
when expresión booleana else
...
Nombre de la señal <= expresión
when expresión booleana else
expresion;
```

```
-- asignación de señal seleccionada
with expresión select
señal <= valor1 de señal when alternativa 1,
         valor2 de señal when alternativa 2,
                  .
                  .
         valorn de señal when others;
```

Asignación de señales

El miembro de la derecha del operador de asignación <= se le conoce como expresión, cuyo valor se obtiene evaluándola completa. Luego ese valor se deposita en el miembro de la izquierda. En la implementación de circuitos simples, el miembro de la derecha es la función minimizada, que asigna un valor de verdad a la salida.

Asignación de condición de señales

"Cuando una señal cambia de valor se dice que se ha producido un evento. La existencia de eventos da lugar a otro evento. De esta manera se actualizan los valores de las señales" (Carpio, 2000, pág. 110). De esta manera cuando la expresión booleana cambie de valor, la expresión que se le asigne a la señal será distinta.

Asignación de condición de señales

"Este tipo de declaración de asignación evalúa la expresión que acompaña a la declaración with y cuando el valor coincide con una de las alternativas señaladas después de la palabra clave when, entonces el valor correspondiente a esta alternativa se le asigna al nombre de la señal." (Carpio, 2000, pág. 103)

Cuando el valor de la señal para varias alternativas es el mismo, pueden ser expresadas como una lista de valores separadas por barras verticales "|" La palabra clave others sirve para cubrir todos los casos posibles faltantes.

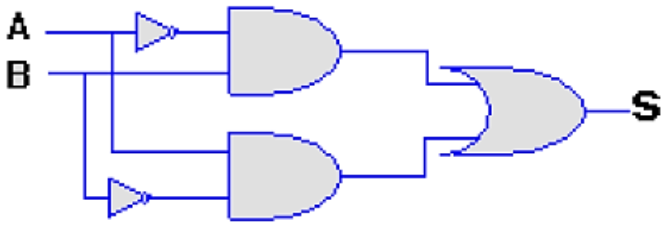
Valor de la señal when alternativa1 | alternativa2 | alternativa3 ..... |alternativaN.

Las barras verticales (|) tienen el mismo significado que la operación lógica OR.

Ejercicio

Realizar la descripción en VHDL de tipo flujo de datos de la compuerta XOR.

$$S = /A.B + A./B$$



A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

```
ARCHITECTURE XOR1 OF XOR IS
BEGIN
    S <= (not A and B) or (not B and A);
END XOR1;
```

- asignación de señales.

```
ARCHITECTURE XOR2 OF XOR IS
BEGIN
    S <= '0' WHEN (A = B) ELSE '1';
END XOR2;
```

- asignación condicional de señales.

```
ARCHITECTURE XOR3 OF XOR IS
BEGIN
    WITH A&B SELECT
    S <='1' WHEN "01" | "10",
        '0' WHEN OTHERS;
END XOR3;
```

- asignación de señales seleccionadas.

2.3 Subtema 3: Aplicaciones

"El rango de aplicaciones de las FPGA es muy amplio, debido a la versatilidad y a la flexibilidad de estos dispositivos; siendo la principal aplicación de las FPGAs el **procesamiento digital de señales (DSP)**, comunicaciones, procesado de datos, etc. La elección de una FPGA para aplicaciones de tratamiento de señal se debe a su alta frecuencia de trabajo, a su capacidad de procesamiento en paralelo, y a su bajo precio en comparación con los ASICs. En general, la lógica de un CPLD es insuficiente para realizar dicho procesamiento." (Carpio, 2000, pág. 201)

En la siguiente ilustración puede apreciarse la distribución de las aplicaciones de las FPGA.

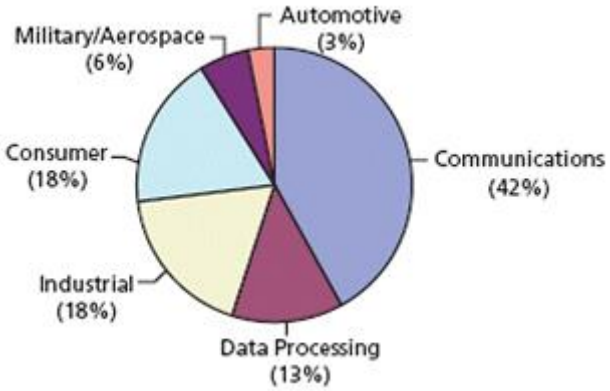


FIGURA 7: Distribución de aplicaciones de las FPGA
"VHDL: Lenguaje para síntesis y modelado de circuitos, 9na edición" Pag 250

De esta aplicación se derivan una gran variedad de aplicaciones de las FPGA, citándose algunas de ellas a continuación:

- **"Sistemas de visión artificial:** En el mundo actual existen cada vez en más números dispositivos que disponen de un sistema de visión artificial. Ejemplo de esto son las cámaras de videovigilancia, robots, etc. Muchos de estos dispositivos precisan de un sistema para conocer su posición, reconocer los objetos de su entorno, reconocer rostros de personas, y poder actuar e interactuar con ellos de la forma adecuada. Esta característica requiere manejar unos volúmenes de imágenes muy elevados, tratar dichas imágenes para detectar objetos, reconocer rostros, etc., en la gran mayoría de ocasiones en tiempo real." (Carpio, 2000, pág. 210)

- **"Sistemas de imágenes médicas:** Cada vez con más frecuencia se están empleando las FPGAs para el tratamiento de imágenes biomédicas obtenidas mediante

procesos de PET, escáner CT, rayos X, imágenes tridimensionales, etc. Estos sistemas de visión médica cada vez precisan de más resolución y de una capacidad de procesamiento mayor, incluso muchas necesitan poder desarrollarse en tiempo real, por lo que las prestaciones que ofrecen las FPGAs de frecuencia y procesamiento en paralelo se adaptan muy bien a estas necesidades." (Carpio, 2000, pág. 210)

- **"Radio definida por software (SDR):** De forma tradicional, una radio consistía en una antena, encargada de recibir y enviar una señal, y un hardware encargado de procesar esa señal, filtrarla, modificar su frecuencia, etc. Este hardware no podía modificar de forma notable la funcionalidad para la cual había sido diseñada. En la actualidad gran parte de esta funcionalidad se traslada a un dispositivo electrónico, que con frecuencia suele ser una FPGA, pudiendo limitarse la parte analógica a una antena y a los convertidores ADC y DAC." (Carpio, 2000, pág. 211)

- **"Codificación y encriptación:** La seguridad en el envío de mensajes es fundamental en la vida diaria, por ejemplo, a la hora de enviar un email o de realizar una compra por internet, y lo es más aún en el ámbito militar, aeronáutico y gubernamental. En este terreno, la encriptación eficiente y segura de mensajes se convierte en un objetivo prioritario. Las FPGA pueden aportar en este terreno su capacidad de manejar grandes volúmenes de información y sus bloques optimizados para realizar operaciones aritméticas." (Carpio, 2000, pág. 211)

- **"Radioastronomía:** La radioastronomía es la ciencia que se encarga de estudiar los fenómenos que ocurren en el espacio mediante la captación de la radiación electromagnética procedente de éste. De forma similar a las aplicaciones anteriores, precisa del procesamiento de una gran cantidad de información en el que la FPGA puede aportar todo su potencial." (Carpio, 2000, pág. 211)

- **"Reconocimiento de voz:** El reconocimiento de la persona que habla es una técnica empleada en seguridad, sistemas de recuperación de información, etc., y se espera que en el futuro su ámbito de aplicación aumente. En este contexto, la FPGA resulta muy eficiente a la hora de realizar la comparación de la voz de una persona con unos patrones previamente almacenados." (Carpio, 2000, pág. 211)

- **"Aeronáutica y defensa:** Además de las mencionadas previamente, existen multitud de aplicaciones aeronáuticas y de defensa que emplean FPGA debido a las buenas características que éstas ofrecen." (Carpio, 2000, pág. 211)

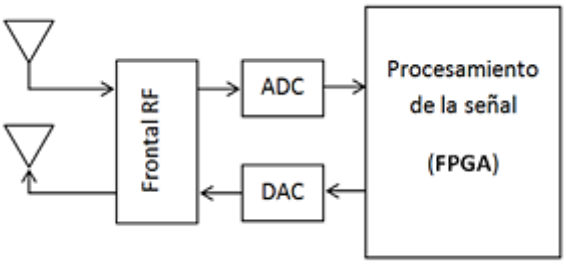


FIGURA 8: Diagrama de bloques de FPGA
"VHDL: Lenguaje para síntesis y modelado de circuitos, 9na edición" Pag 250

"La principal ventaja de este tipo de radio es que su funcionalidad viene definida por el diseño del software, de forma que su modificación o actualización es sencilla y no precisa de la sustitución de ningún elemento de hardware." (Carpio, 2000, pág. 212)

Data Center / Cloud: El internet de las cosa (IoT), y en general el big data, están generando un crecimiento exponencial de los datos adquiridos y procesados, que junto con el análisis computacional de los mismos mediante técnicas de aprendizaje profundo de múltiples operaciones paralelas/concurrentes, están conllevando una alta demanda de capacidad computacional de baja latencia, flexible y segura que no se resuelve añadiendo más servidores / blades, debido al disparatado incremento del coste en espacio, consumo y dinero. Bajo este panorama, las puertas del mundo del Data Center se están abriendo de manera masiva a las FPGAs, debido a su capacidad de aceleración computacional, flexibilidad de configuración y la seguridad que garantiza el hardware frente al software.

3. Bibliografía

» Carpio, F. P. (2000). VHDL: Lenguaje para sintesis y modelado de circuitos. Madrid: Alfaomega.

» Floyd, T. L. (2006). FUNDAMENTOS DE SISTEMAS DIGITALES. Madrid: PEARSON EDUCACIÓN.