UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL DE TELECOMUNICACIONES

TEMA:

Herramienta FPGA para el diseño en bloques y programación VHDL en la asignatura de Sistemas Digitales I

Previa la obtención del Título

INGENIERO EN TELECOCUMINACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL DE TELECOMUNICACIONES

ELABORADO POR:

Asanza Briones Angel Steven

GUAYAQUIL, JULIO DE 2012

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el señor Angel Steven Asanza Briones, como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL EN TELECOMUNICACIONES.

Guayaquil, 02 Julio 2012

MsC. Luzmila Ruilova Aguirre.
DIRECTOR DE TESIS

REVISADO POR

RESPONSABLE ACADÉMICO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

ASANZA BRIONES ANGEL STEVEN

DECLARO QUE:

El proyecto de grado denominado “Herramienta FPGA para el diseño en bloques y programación VHDL en la asignatura de Sistemas Digitales I”, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 02 Julio 2012

EL AUTOR

ASANZA BRIONES ANGEL STEVEN

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, ASANZA BRIONES ANGEL STEVEN

Autorizo a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: “Herramienta FPGA para el diseño en bloques y programación VHDL en la asignatura de Sistemas Digitales I”, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Guayaquil, 02 Julio 2012

EL AUTOR

ASANZA BRIONES ANGEL STEVEN

ÍNDICE GENERAL

Índice de Figuras

Capítulo 2

Capitulo 3

Capítulo 4

Agradecimientos

A mi familia por el apoyo en todo el transcurso de la carrera de telecomunicaciones.

A mi director de tesis MsC. Luzmila Ruilova Aguirre y en especial al MsC. Fernando Palacios por brindarme su apoyo, en el desarrollo del presente proyecto de tesis de grado.

A los revisores metodológicos por su colaboración, observaciones y aportes.

Al MsC. Manuel Romero Paz, Decano de la Facultad de Educación para el Desarrollo, por su aportes en las materias impartidas en la carrera de Ingeniería en Telecomunicaciones.

Dedicatoria

A nuestros padres por su entrega abnegada y sublime siendo pilares fundamentales para culminar nuestra vida universitaria, a nuestros hermanos y amigos por su apoyo incondicional ayudándonos con sus experiencias y consejos para así cumplir nuestras metas, convirtiéndonos en profesionales de éxito y mejores personas.

Resumen

El estudio del diseño vhdl es capaz de simular perfectamente el comportamiento lógico de un circuito por lo que el estudiante de ingeniera en telecomunicaciones se darán cuenta del proceso en el ámbito estudiantil y laboral se tomara características especificas en el hardware que se requiere utilizar, para el estudio del diseño vhdl utilizaremos una herramienta de trabajo Quartus II de altera.

Con este trabajo pretendemos, además, que el lector en general y los estudiantes de ingeniería en telecomunicaciones en particular, dispongan de una herramienta operativa para comprobar la valides de su hardware. Por ejemplo, como lenguaje de entrada para las herramientas de síntesis CAD, tales como VHDL Logic Synthesis de Synopsys Inc., Autologic de Mentor Graphics o Metamor, utilizadas en el diseño automático de circuitos integrados. Para analizar y medir diferentes modelos de procesadores segmentados, RISC, vectoriales, superescalares, sistólicos, VLIW- y multiprocesadores, así como arquitecturas especificas de la aplicación. O bien para el estudio de modelos no interpretados de análisis del rendimiento de configuraciones de sistemas.

En pocas palabras, disponer de un lenguaje que permita llevar al terreno operativo los conocimientos del dominio del hardware, de la misma manera que un lenguaje de propósito general como Pascal o Modula lo hace con los conocimientos del dominio del software.

Summary

The study design is able to simulate vhdl perfectly logical behavior of a circuit so that the student of engineering in telecommunications will realize the process in the field study and work was taken in the hardware specific features are required for use, for vhdl design study will use a working tool Quartus II altered.

In this paper we further that the general reader and students of telecommunications engineering in particular, have an operational tool to check the validity of your hardware. For example, as input language for CAD synthesis tools such as VHDL Logic Synthesis of Synopsys Inc., Mentor Graphics Autologic or Metamor, used in the automated design of integrated circuits. To analyze and measure different processor models segmented, RISC, vector, superscalar, systolic and VLIW multiprocessor-and application-specific architectures. Or models for the study of uninterpreted performance analysis of system configurations.

In short, have a language to bring the operating field of the domain knowledge of the hardware, just as a general purpose language like Pascal or Modula does so with knowledge of the domain of software.

CAPÍTULO 1: GENERALIDADES

1.1. INTRODUCCIÓN

VHDL es el idioma de muy alta velocidad descripción de circuitos integrados de hardware. Que es uno de los lenguajes de programación utilizado para modelar un sistema digital de flujo de datos, el estilo de comportamiento y estructurales de los modelos. Este lenguaje fue introducido por primera vez en 1981 por el Departamento de Defensa (DoD) bajo el programe VHSIC. En 1983, IBM, Texas Instruments e Intermetrics comenzó a desarrollar este lenguaje. In 1985 En 1985 VHDL 7.2 versión fue lanzada.

Aparece como un proyecto del Departamento de Defensa para EEUU (1982), con el fin de disponer de una herramienta estándar e independiente para la especificación y documentación de los sistemas electrónicos a lo largo de todo su ciclo de vida. Tras las primeras versiones del lenguaje, el IEEE lo adopta y desarrolla como el HDL estándar.

1.2. ANTECEDENTES

La carrera de telecomunicaciones abarca una importancia muy amplia en la sociedad por la cual los estudiantes de la facultad técnica se encuentra en una investigación constante de la tecnología en la actualidad dicho esto se fundamenta en las simulaciones, características y descripciones de hardware en un circuito electrónico .Con la finalidad de recibir un estudio analítico de los elementos se recurre al diseño VHDL.

El desarrollo electrónico de los últimos tiempos se ha visto fuertemente dominado y conducido por la impresionante evolución de la microelectrónica desde su nacimiento en 1959-60. Durante los años setenta, junto con la revolución que suponen las memorias RAM y procesadores en forma de chip monolítico, se preparan las condiciones para el gran salto que el diseño micro electrónico dará en los años ochenta.

Durante los años ochenta, tras detectarse la necesidad de un lenguaje para dar soporte a las distintas etapas y niveles de abstracción del proceso de diseño, se desarrollan y consolidan dos de ellos: Verilog y VHDL.

El VHDL (VHSIC Hardware Description Language) aparece como un proyecto del Departamento de Defensa de EEUU (1982), con el fin de disponer de una herramienta estándar e independiente para la especificación y documentación de los sistemas electrónicos a lo largo de todo su ciclo de vida. Tras las primeras versiones del lenguaje, el IEEE lo adopta y desarrolla como el HDL estándar (1ª versión en 1987 y 2ª en 1993).

El desarrollo, difusión y estandarización de los lenguajes Verilog y VHDL, aunque sólo sean herramientas básicas para la descripción de circuitos y sistemas electrónicos fue, y sigue siendo, un hecho determinante en el desarrollo de las nuevas metodologías y herramientas de diseño electrónico.

1.3. JUSTIFICACIÓN DEL PROBLEMA

El estudio de la materia de digitales en la Facultad de Educación Técnica para el Desarrollo (FETD) se realiza en una PC que hace el trabajo de simular el comportamiento de un circuito electrónico. Las simulaciones se hacen prácticas ya que los elementos de cada circuito podrían tener un precio excesivo. En la plataforma del diseño VHDL y diseño de bloques podemos describir el hardware genérico y acelerar el proceso de diseño.En el desarrollo del tema se ve el estudio analítico que se da uso en materias de la ingeniería.

1.4. DEFINICIÓN DEL PROBLEMA

En la asignatura de Digitales I de V Ciclo de la Carrera de Ingeniería en Telecomunicaciones surge la necesidad de analizar el software y hardware (equipo electrónico), para el diseño de sistemas digitales basado en puertas lógicas, a través de prácticas para aprender el diseño de circuitos digitales mediante el uso de VHDL y dispositivos lógicos programables (CPLD o FPGA).

1.5. OBJETIVOS DE LA INVESTIGACIÓN

1.5.1OBJETIVOS GENERALES

Analizar y evaluar la herramienta FPGA de los componentes digitales y técnicas de diseño en bloques y programación VHDL en la asignatura de Sistemas Digitales I.

1.5.2 OBJETIVOS ESPECÍFICOS

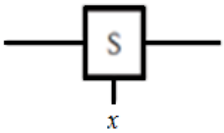
- Describir el estado del arte de VHDL y de la Tecnología de implementación FPGA en circuitos digitales.
- Apreciar las similitudes y diferencias entre los lenguajes HDL (por ejemplo, Verilog, SystemC
- Diseñar circuitos digitales simples (por ejemplo, circuitos aritméticos simples) con VHDL.
- Evaluar los diseños de los circuitos digitales mediante la herramienta FPGA y de simulación Quartus II.

CAPÍTULO 2: ESTADO DEL ARTE DE LÓGICA COMBINACIONAL YVHDL

2.1 VARIABLES Y FUNCIONES

Los circuitos binarios predominan en los sistemas digitales gracias a su simplicidad que resulta de restringir las señales para que adopten solo dos valores posibles el elemento binario más sencillo es el interruptor de dos estados si una variable de entrada controla un interruptor entonces este se dice que se abre x=0 y se cierra si X=1 como se ilustra en la figura 2.1a usaremos el símbolo grafico de la figura 2.1b para representar estos tipos de interruptores en los diagramas que siguen.

(a) Switch de dos estados

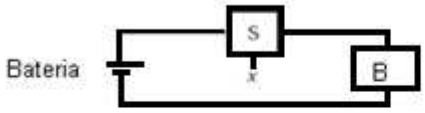


(b) Símbolo para un switch

Figura 2.1: Un switch binario.
Elaborado: El autor

Considérese una aplicación simple de un interruptor donde este se enciende o apaga una pequeña bombilla. Esta acción se logra con el circuito de la figura 2.2a. Una batería proporciona la fuente de poder. La bombilla brilla cuando pasa la corriente necesaria para su filamento que es una resistencia eléctrica. La corriente fluye cuando el interruptor se cierra; es decir

(a) Conexion simple a una bateria



b) Uso de una conexion a tierra como trayectoria de regreso

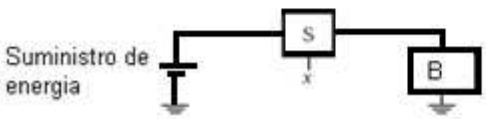


Figura 2.2: Bombilla por interruptor
Elaborado: El autor

De acuerdo a la figura 2.1 (a)$x = 1$, la entrada que ocasiona el cambio en el comportamiento del circuito es el control x del interruptor. La salida se define como el estado (o condición) de la luz que se denota con la letra L. Si la luz se enciende, entonces $L = 1$; si se apaga entonces $L = 0$, con esta convención es posible describir el estado de la luz como función de la entrada de la variable x. Ya que si $L = 1$ entonces sabemos que $x = 1$ y $L = 0$; pero si$x = 0$ puede describirse que:

$$L(x) = x \qquad \text{Ecuación 2.1}$$

Esta sencilla expresión lógica describe la salida en función de la entrada, se dice que $L(x) = x$es una función lógica; y que x es una variable de entrada.En la figura 2.2a se muestra el circuito que representa a una linterna ordinaria, donde el interruptor es un dispositivo mecánico sencillo. En un circuito electrónico el interruptor se implementa como un transistor y la luz puede ser un LED (diodo emisor de luz).

Un circuito electrónico recibe la energía de una fuente de cierto voltajede 5 voltios. Un lado de la fuente se conecta a tierra, como muestra la figura 2.2b. La conexión a tierra también puede usarse como la trayectoria de regreso para la corriente, a fin de cerrar el circuito, lo que se logra conectando un lado de luz a tierra, como se indica en la figura. Desde luego la luz puede conectarse con un cable directamente al lado aterrizando de la fuente de poder, como se advierte en la figura 2.2a.

Considérese ahora la posibilidad de usar dos interruptores para controlar el estado de la luz. Sean x_1y $x_2$sus entradas de control. Los interruptores pueden conectarse en serie o paralelo, como se muestra en la figura 2.3. Si se usa una conexión en serie la luz se encenderá solo si ambos interruptores están cerrados. Si uno esta abierto, la luz estará apagada. Este comportamiento puedes describirse con la expresión

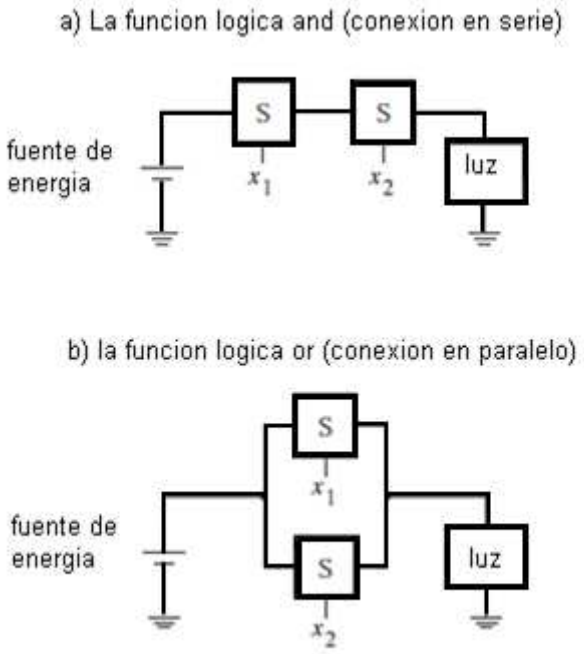


Figura 2.3: Dos funciones básicas
Elaborado: El autor

El símbolo "." representa al operador lógico "and" y se dice que el circuito de la figura 2.3a es la implementación de una función lógica AND.En la figura 2.3b se presenta la conexión en paralelo de dos interruptores. En este caso la luz se encenderá si cualquiera de los interruptores, x_1y x_2, se cierran o si ambos se cierran; el LED se apagará, en el caso de que los dos interruptores están abiertos. Este comportamiento puede expresarse como:

$$L(x_1, x_2) = x_1 + x_2 \qquad \text{Ecuación 2.2}$$

El símbolo "+" es el operador OR y se dice que el circuito de la figura 2.3b implementa la función lógica OR.En la expresiones anteriores para AND y OR la salida $L(x_1, x_2)$es una función lógica con variables de entrada x_1y x_2, La función AND y OR son de las funciones lógicas mas importantes, junto con algunas otras funciones simples se usan como los bloques fundamentales de la implementación de todos los circuitos lógicos. En la figura 2.4 se muestra como usar tres interruptores para controlar la luz de forma más compleja esta conexión serie paralelo realiza la función lógica:

$$L(x_1, x_2, x_3) = (x_1 + x_2) \cdot x_3 \qquad \text{Ecuación 2.3}$$

La luz se enciende si $x3 = 1$y, al mismo tiempo, al menos unas de las entradas $x1$ o $x2$ es igual a 1

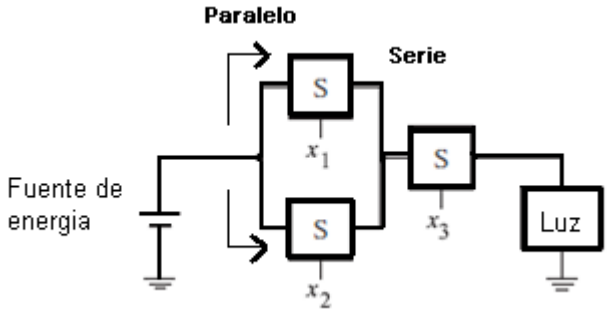


Figura 2.4 Conexión serie - paralelo
Elaborado: El autor

2.2 INVERSIÓN

Hasta el momento hemos supuesto que cierta acción positiva como encender la luz tiene lugar cuando se cierra un interruptor. Es igualmente interesante y útil considerar la posibilidad que suceda una acción positiva cuando se abre un interruptor, supóngase que conectamos la luz como se muestra en la figura 2.5. En este caso el interruptor se conecta paralelo de la luz, en lugar de en serie.

En consecuencia un interruptor cerrado ocasionara un cortocircuito y evitara que la corriente pase por el. Nótese que hemos incluido en el circuito un resistor adicional para garantizar que el interruptor cerrado no ocasionara un cortocircuito en la fuente de energía. La luz se encenderá cuando el interruptor se abra. Formalmente, este comportamiento funcional se expresa como

$$L(x) = \bar{x}$$

Donde $L = 1\ if\ x = 0$

$L = 0\ if\ x = 1$

El valor de esta función es el inverso de esta variable de entrada, en lugar de utilizar la palabra inverso, es más común utilizar el término complemento. Por tanto se dice que $L(x)$ es un complemento de x en este ejemplo. Otro término empleado con frecuencia para la misma operación es la operación NOT. Diversas anotaciones se utilizar para indicar el complemento.

En la expresión precedente se coloca una barra sobre la x. quizá esta notación sea mejor de un Angulo visual. Sin embargo cuando se requiere complemento en la expresión se escribe con

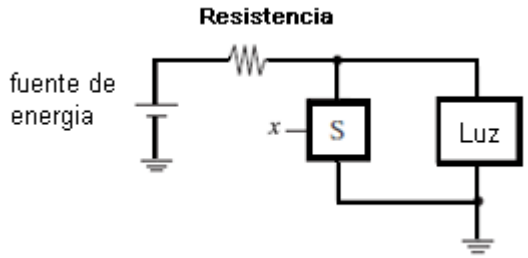


Figura 2.5 Un circuito inverso
Elaborado: El autor

El teclado de una computadora lo que a menudo sucede cuando se emplean herramientas CAD:

$$\overline{x} = x' = !x = \sim x = \text{NOT } x$$

La operación complemento puede aplicarse a una sola variable o aoperaciones más complejas. Por ejemplo, si

$$f(x1, x2) = x1 + x2$$

Entonces el complemento de f es:

$$\tilde{f}(x1, x2) = \overline{x1 + x2}$$

2.3 TABLA DE VERDAD

Hemos presentado las tres operaciones lógicas más básicas And, OR y complemento relacionando con circuitos sencillos construidos con interruptores. Este enfoque confiere a tales operaciones cierto “significado físico”.

Podemos representar una función de conmutación mediante varias expresiones de conmutacióndiferente, pero equivalentes si evaluamos una función de conmutación para todas las posibles combinaciones de entrada y presentamos los resultados como una tabla, obtenemos unarepresentaciónúnica de la función llamada tabla de verdad (Victor P. Nelson ; H. Troy Nagle ; Bill D. CArroll ; J.David Irwin, Mexico)

Las mismas operaciones también pueden definirse en formas de tablas de verdad, como se muestra en la figura 2.6 las primeras dos columnas a la izquierda de la línea vertical doble proporcionan las cuatros posibles combinaciones de valores lógicos que la variable $x1$ y $x2$ pueden tener.La siguiente columna define la operación and para cada combinación de valores $x1$ y $x2$, y la ultima columna define la operación OR. Puesto que con frecuencia es necesario hacer referencia a “combinaciones de valores lógicos” aplicados a una variable, se adoptará un términomás corto, valoración, para denotar tal combinaciones de valores lógicos (Espinosa, Febrero de 2009)

x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
		AND	OR

Figura 2.6 tabla de verdad

Elaborado: sitio web (blogspot-circuitoslogicos.geovanni, 2010)

x_1	x_2	x_3	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura 2.7 tabla de tres entradas
Elaborado: sitio web (blogspot)

La tabla de verdad es un auxiliar útil para describir información relacionada con funciones lógicas.Se utiliza para definir funciones especificas y demostrar la valides de ciertas relaciones funcionales. Las tablas de verdad pequeñas son fáciles de manejar. Sin embargo, crecen exponencialmente en

tamaño con el número de variables. Una tabla de verdad de tres variables tiene ocho filas porque hay ocho posibles valoraciones para esas variables.

La figura 2.7 proporciona una figura semejante, que define las funciones AND y OR para tres entradas, la tabla de verdad tiene $2n$ filas. Las operaciones AND y OR pueden entenderse a n variables. Una función AND de variables $x1, x2 \ldots Xn$ tiene el valor de 1 solo si la n variable es iguales a 1. Una funciono r de variables $X1, X2, \ldots Xn$ tiene el valor de 1 solo si uno o mas de las variables es igual a 1.

2.4 COMPUERTAS LOGICAS Y CIRCUITOS

Las tres operaciones lógicas básicas de un circuito en la secciones previa pueden usarse para implementar funciones lógicas de cualquier complejidad. La puesta en marcha de una función compleja puede requerir muchas de esas operaciones básicas. Cada operación lógica puede implementarse electrónicamente con transistores. Se han introducidos las cinco compuertas lógicas (And, Or, Inversor, Nand,y Nor) y los símboloslógicosestándar que se usan para representarlas en diagrama de circuitos lógicos. Aunque usted puede encontrar que en algunos diagramas de circuitos aun se usan estos símbolosestándar exclusivamente. (Tocci, Ronald.dj. Y Widmer,NEal.S., 2009)

Los símbolos gráficos de las compuertas AND, OR, NOT se muestra en la figura 2.8 en el lado izquierdo reindica como dibujar las compuertas AND Y OR cuando hay pocas entradas en el lado derecho se muestra como aumenta los símbolos para dar cabida a un mayor número de entradas.

Un circuito más grande se implementa mediante una red de compuertas. Por ejemplo la función lógica de la figura 2.4 puede realizarse mediante la red de la figura 2.9 la complejidad de una red tiene efecto directo en su costo como simple es deseable reducir su costo. Como siempre es deseable reducir los costos de los productos fabricados

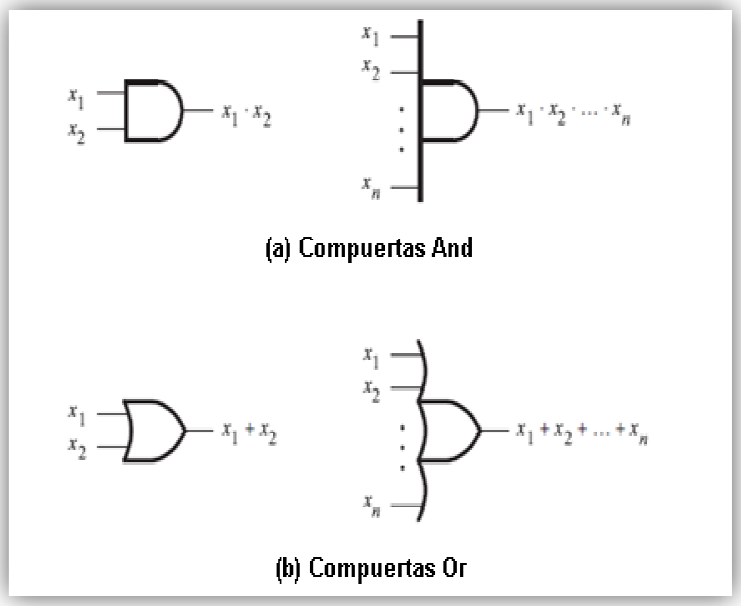
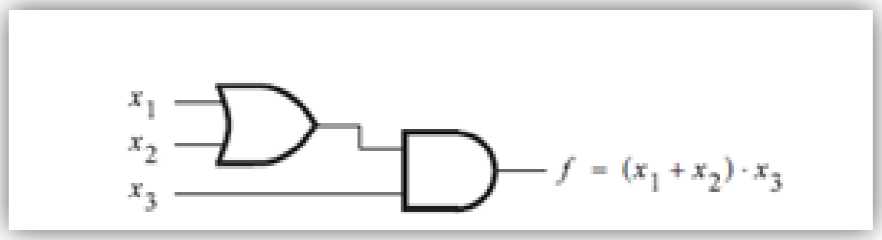


Figura 2.8 Compuertas básicas
Elaborado: El Autor



Figuras 2.9 Función de la figura 2.4

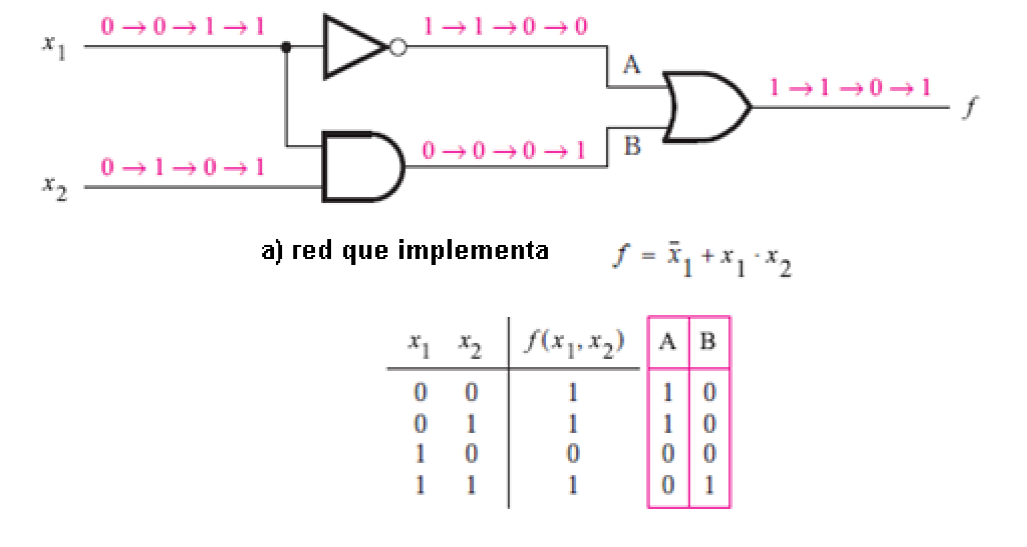
Elaborado: sitio web (blogspot-circuitoslogicos.geovanni, 2010)

2.4.1 ANÁLISIS DE UNA RED LOGICA

Un diseñador de sistemas digitales enfrenta dos conflictos básicos. Debe ser posible determinar la función de una red lógica existente. Esta tarea se conoce como proceso de análisis. La tarea inversa de diseñar una nueva red desempeñe cierto comportamiento funcional se denomina proceso de síntesis. El proceso de análisis es mucho más abierto y sencillo que el de síntesis.

En la figura 2.10 (a) se muestra una red simple formada por tres compuertas. A fin de determinar su comportamiento funcional considérese si se aplican todas las señales de entradas posibles. Supóngase que se inicia

esta obliga a la salida de la compuerta NOT hacer igual a y la salida de la compuerta AND a ser igual a cero, puesto una de las entradas de las compuertas OR es igual a . Por tanto si si se hace entonces no ocurrirá cambio en el valor de f, pues las salidas de las compuertas NOT y AND seguirán siendo respectivamente. a continuación , si se aplica entonces la salida de la compuerta NOT cambiara a cero mientras que de la compuerta AND continuara sendo 0. Ambas entradas a la compuerta OR serán iguales a ; por ende el valor de será 0 finalmente, sea entonces la salida de la compuerta AND será .Lo que produce que sea igual a . La explicación verbal puede resumirse en la tabla de verdad de la figura 2.1b



a) red que implementa $f = \bar{x}_1 + x_1 \cdot x_2$

x_1	x_2	$f(x_1, x_2)$	A	B
0	0	1	1	0
0	1	1	1	0
1	0	0	0	0
1	1	1	0	1

Figura 2.10 Red de implementación
Elaborado: El Autor

2.4.2 DIAGRAMA DE TIEMPO

los diagramas de sincronización se usan ampliamente para mostrar como las señales digitales cambian con el tiempo, y especialmente para mostrar la relación entre dos o más señales digitales en el mismo circuito o sistema , al representar una o más señales digitales en un osciloscopio o analizador lógicos se pueden comparar las señales con los diagramas de temporización esperados esta es una parte muy importante de los

procedimientos de prueba y detección de fallas que se usan en los sistemas digitales (Tocci, Ronald.dj. Y Widmer,NEal.S., 2009)

El comportamiento de la red de la figura 2.10a se determino al considerar los cuatros posibles valores de entrada $x1\ y\ x2$, supóngase que las señales correspondientes a esas valoraciones se aplican a la red en el orden que acabamos de describir; esto es: $(x1, x2) = (0,0)$ seguido de $(0,1), (1,0)\ y\ (1,1)$. El tiempo corre de izquierda a derecha la valoración de cada entrada se mantiene cierto periodo fijo .En la figura se muestra las formas de ondas de las entradas y salidas de la red así como de las señales internas en los puntos AyB

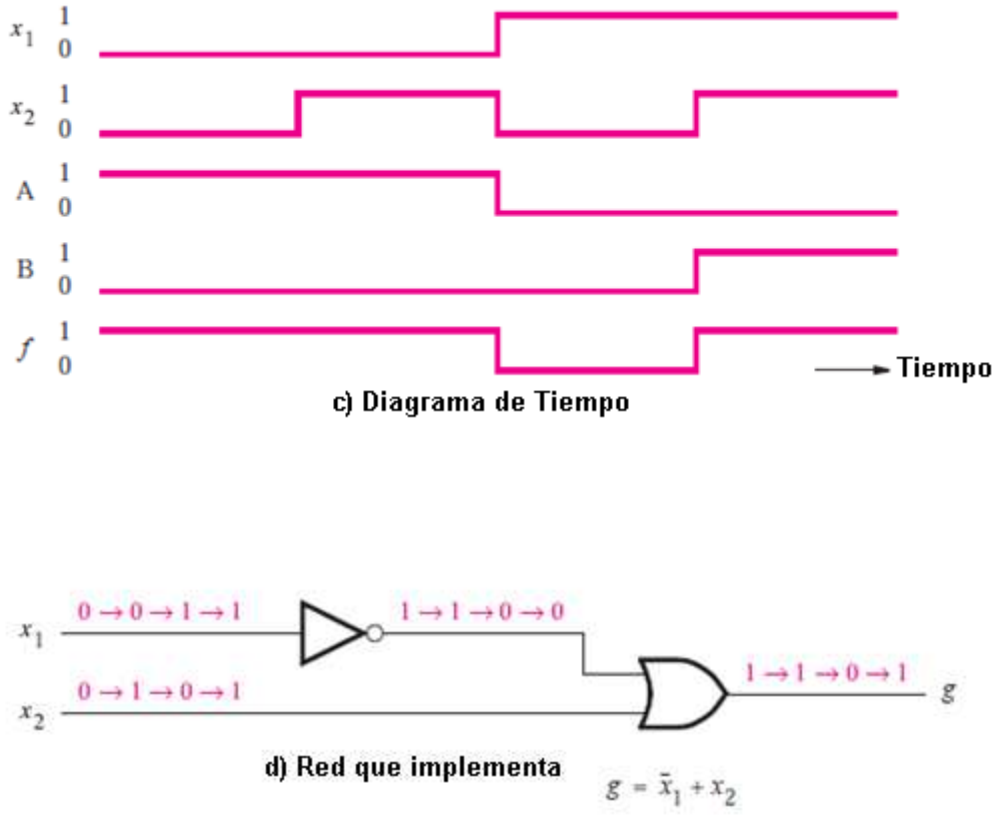


Figura 2.11 Ejemplo de redes lógicas
Elaborado: sitio web (blogspot)

2.5 ALGEBRA BOOLEANA

En 1849 GeorgeBoole publico un esquema de las descripciones algebraicas de los procesos relativos al pensamiento y al razonamiento lógico. Luego ese esquema y sus posteriores refinamientos recibieron el nombre de algebra booleana. Fue casi 100 años después que esta algebra hayo aplicación en la ingeniería.

En electrónica digital se trabaja con variables que pueden tener sólo dos valores, 0 ó 1. Por suerte los primeros ingenieros que trabajaron con electrónica digital ya encontraron las matemáticas necesarias desarrolladas, pues en 1854, mucho antes de que se hubiera inventado el transistor, George Boole definió un álgebra para manipular variables que sólo podían tener los valores cierto y falso, (Espinosa, Febrero de 2009)

A fines de la dedicada de 1930, Claude Shannon demostró que es algebra booleanas constituye un medio eficaz para describir circuitos construido con interruptores por tanto esta algebra sirve para describir circuitos lógicos. En este apartado veremos que el algebra booleanas constituye una poderosa herramienta para diseñar y analizar circuitos lógicos. El lector advertirá que sienta las bases de gran parte de la tecnología digital de nuestros días.

2.5.1 AXIOMAS DEL ALGEBRA BOOLEANA

Como cualquier algebra, la booleana se basa en un conjunto de reglas derivadas a partir de un pequeño número de suposiciones fundamentales que reciben el nombre de axiomas supóngase que el algebra booleanas b comprende elementos que toman uno de dos valores, 0 y 1. Supóngase asimismo que losaxiomassiguientes son verdaderos

1a. $0 \cdot 0 = 0$
1b. $1 + 1 = 1$
2a. $1 \cdot 1 = 1$
2b. $0 + 0 = 0$
3a. $0 \cdot 1 = 1 \cdot 0 = 0$
3b. $1 + 0 = 0 + 1 = 1$
4a. If $x = 0$, **entonces** $\overline{x} = 1$
4b. If $x = 1$, **entonces** $\overline{x} = 0$

TEOREMAS DE UNA SOLA VARIABLE

A partir de los axiomas se pueden definirse ciertas reglas para usar las variables individuales, a menudo esas reglas se denomina teoremas.Por otro lado, veremos que en el álgebra de Boole todos los teoremas tienen unaversión dual, que es la obtenida cambiando los ceros por unos, los productos porsumas y viceversa. (Muñoz Frías, 2012)

5a. $x \cdot 0 = 0$
5b. $x + 1 = 1$
6a. $x \cdot 1 = x$
6b. $x + 0 = x$
7a. $x \cdot x = x$
7b. $x + x = x$
8a. $x \cdot \overline{x} = 0$
8b. $x + \overline{x} = 1$
9. $\overline{\overline{x}} = x$

DUALIDAD

Nótese que hemos numerado por pares de axiomas y los teoremas de una sola variable. Lo hicimos para reflejar la importancia del principio de dualidad. Dada una expresión lógica, su dual se obtiene sustituyendo todos lo operadores + con operadores y viceversa y sustituyendo todos los $0\ con\ 1$, y viceversa. El dual de cualquier proposición verdadera (axiomas o teoremas) en algebra booleanas también es una proposiciónverdadera. Si bien este punto de la explicación el lector no advertirá porque la dualidad es un concepto útil le quedara claro más adelante cuando se muestre que la dualidad implica la existencia de al menos dos formas de expresar toda función lógica con algebra

booleanas. Con frecuencia una expresión conduce a una implementaciónfísicamás simple que la otra y por lo tanto es preferible.

PROPIEDADES DE DOS A TRES VARIABLES

Para que sea posible tratar con varias variables es útil definir algunas identidades algebraicas de dos a tres variables. Para cada una de ellas también se proporciona su versióndual. Estas identidades suelen denominarse propiedades. Se conocen por los nombres que se indican a continuación si $x = y\ y\ z$ son variables en B, entonces se cumplen las siguientes propiedades

10*a*.	$x \cdot y = y \cdot x$	**Conmutativa**
10*b*.	$x + y = y + x$	
11*a*.	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	**Asociativa**
11*b*.	$x + (y + z) = (x + y) + z$	
12*a*.	$x \cdot (y + z) = x \cdot y + x \cdot z$	**Distributiva**
12*b*.	$x + y \cdot z = (x + y) \cdot (x + z)$	
13*a*.	$x + x \cdot y = x$	**Absorcion**

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0
		LHS		RHS		

Figura 2.12 Prueba de teorema de Morgan
Elaborado: El Autor

13*b*.	$x \cdot (x + y) = x$	**Combinacion**
14*a*.	$x \cdot y + x \cdot \overline{y} = x$	
14*b*.	$(x + y) \cdot (x + \overline{y}) = x$	**Teorema de Morgan**
15*a*.	$\overline{x \cdot y} = \overline{x} + \overline{y}$	
15*b*.	$\overline{x + y} = \overline{x} \cdot \overline{y}$	
16*a*.	$x + \overline{x} \cdot y = x + y$	
16*b*.	$x \cdot (\overline{x} + y) = x \cdot y$	
17*a*.	$x \cdot y + y \cdot z + \overline{x} \cdot z = x \cdot y + \overline{x} \cdot z$	**Consenso**
17*b*.	$(x + y) \cdot (y + z) \cdot (\overline{x} + z) = (x + y) \cdot (\overline{x} + z)$	

Una vez más, podemos comprobar la validez de estas propiedades, ya sea por inducción perfecta o la realización de la manipulación algebraica. Hemos enumerado una serie de axiomas, teoremas y propiedades. No todos ellos son necesarios definir el álgebra de Boole. Por ejemplo si suponemos que las operaciones + y. están definidas basta incluir los teoremas 5 y 8 y las propiedades 10 y 12. Estos a veces se refieren como postulados básicos. El resto de las identidadespueden derivarse de ellos. Los axiomas anteriores, teoremas y propiedades proporcionan la información necesaria para la realización de la manipulación algebraica de las expresiones más complejas.

2.5.1 DIAGRAMAS DE VENN

Se han sugerido que la inducción perfecta puede ser utilizada para verificar los teoremas y propiedades. Este procedimiento es bastante tedioso y no muy informativo desde el punto de vista conceptual. Una ayuda visual simple que se puede utilizar para este propósito también existe.

El diagrama de Venn se ha utilizadotradicionalmente en lasmatemáticas para proporcionar una interfaz gráfica ilustración de las diversas operaciones ylas relaciones en el álgebra de conjuntos. Un conjunto colección de ISAde elementos t se dice que son los miembros del s.

En el diagrama de Venn loselementos de un conjunto están representados por el área encerrada por un contorno tal como un cuadrado, un círculo, o una elipse. Por ejemplo, en un universo de N números enteros del1 a 10, el conjunto de números pares es $E = \{2, 4, 6, 8, 10\}$.

Un contorno que representa E encierra los números pares. Los números impares formar el complemento de E, por lo que el área fuera del contorno representa $E = \{1, 3, 5, 7, 9\}$. Dado que en el álgebra de Boole, sólo hay dos valores (elementos) en el universo, $B = \{0, 1\}$, diremos que el área dentro de un contorno que corresponde a denota un conjunto s que $s = 1$, mientras que el área fuera del contorno denota $s = 0$.

En el diagrama que darán sombra a la zona donde $s = 1$. El concepto del diagrama de Venn se ilustra en la Figura 2,13. El universo de B está representado por un cuadrado. Entonces las constantes 1 y 0 sonrepresentados como se muestra en las partes (a) y (b) de la figura. Una variable, por ejemplo, x, está representado por un círculo, de tal manera que el área dentro el círculo se corresponde con $x = 1$, mientras que el área fuera del círculo se corresponde con $x = 0$. Esto se ilustra en la parte (c). Una expresión que implique una o más variables se representa por sombreado la zona donde el valor de la expresión es igual a 1. Parte (d) indicar mostrar la complemento de x es representado.

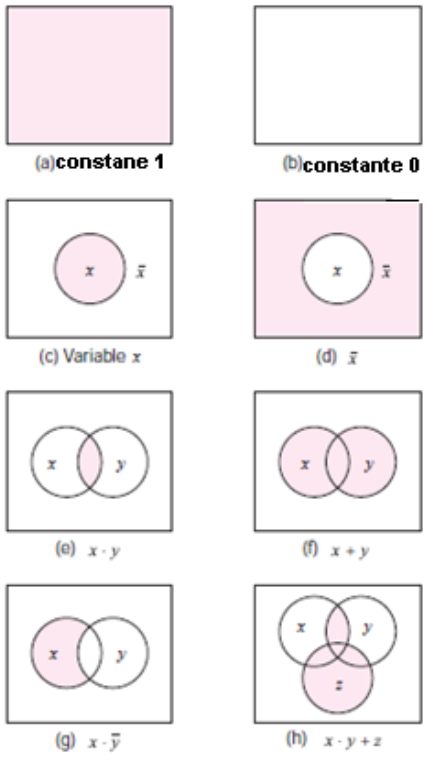


Figura 2.13 Representación de diagrama 1
Elaborado: El Autor

2.5.2 NOTACION Y TERMINOLOGIA

Álgebra booleana se basa en el AND y OR operaciones. Hemos adoptado los símbolos $\cdot$ y $+$ para denotar estas operaciones. Estos son también los símbolos normalizados para la multiplicación aritmética familiar y operaciones de adición .Existe una considerable similitud entre las operaciones booleanas y las operaciones aritméticas, que es la razón principal por la que los mismos símbolos que se utilizan. De hecho, cuando los dígitos individuales están implicados sólo hay una diferencia significativa; el resultado de $1 + 1$ es igual a 2 en la aritmética ordinaria, donde ya que es igual a 1 en álgebra booleana como se define por 7bteorema en la sección 2.5. Cuando se trata con circuitos digitales, la mayor parte del tiempo el símbolo $+$ obviamente representa la operación OR.

Sin embargo, cuando la tarea implica el diseño de circuitos lógicos que realizan operaciones aritméticas, alguna confusión puede desarrollar sobre el uso del símbolo$+$. Para evitar tal confusión, un conjunto alternativo de los símbolos existe para operaciones AND y OR. Es bastante común usar el símbolo ∧ para denotar la operación AND, y ∨ para la operación OR. Así, en lugar de $x1 \cdot x2$, podemos escribir $x\ 1\ \cap\ x\ 2$, y en lugar de $x1 + x2$, podemos escribir $x1\ v\ x2$.

2.5.3PRECEDENCIA DE LAS OPERACIONES

Uso de las tres operaciones básicas AND, OR y NOT se puede construir un número infinito de expresiones lógicas. Se pueden utilizar paréntesis para indicarel orden en que las operaciones deben realizarse. Sin embargo, para evitar un uso excesivo de paréntesis, otro convenio establecerá la prioridad de las operacionesbásicas. Afirma que, en ausencia de paréntesis, las operaciones en una expresión lógica se debe realizar en el orden: NOT, AND, y OR. Así, en la expresión:

$$\boldsymbol{x1 \cdot x2 + x1 \cdot x2}$$

Primero es necesario para generar los complementos de $X1\ y\ X2$. A continuación, los términos producto $x1 \cdot x2\ y\ x1\ x2 \cdot$ se forman, seguido por la suma de los dos términos producto. Tenga en cuenta que, en ausencia de este convenio, tendríamos que utilizar paréntesis para lograr el mismo efecto de la siguiente manera:

$$(\boldsymbol{x1} \cdot \boldsymbol{x2}) + ((\boldsymbol{x1}) \cdot (\boldsymbol{x2}))$$

Finalmente, para simplificar la apariencia de las expresiones lógicas, se acostumbra a omitir el operador · cuando no hay ninguna ambigüedad. Por lo tanto, La expresión anterior se puede escribir como

$$\boldsymbol{x1x2} + \boldsymbol{x1x2}$$

2.6LA SINTESIS CON COMPUERTAS AND, OR Y NOT

Con algunas ideas básicas, podemos trabajar momentáneamente con implementaciones de funciones arbitrarias utilizando (AND, OR y NOT). Suponiendo que queremos diseñar un circuito lógico con dos entradas.

Supongamos que $X1\ y\ X2$ representan los estadosde dos interruptores, uno de que puede ser abierto (0) o cerrado (1).La función del circuito es para controlar continuamente el estado de los interruptores y para producir un valor lógico de salida 1 cuando los interruptores$(x1, x2)$ están en los estados $(0,\ 0), (0,\ 1),\ o\ (1,\ 1\ )$. Si el estado de los interruptores $(1, 0)$, la salida debe ser 0. Otra manera de indicar el comportamiento requerido funcional de este circuito es que la salida debe ser igual a 0 si el interruptor está cerrado $x1\ y\ x2$ está abierto, de lo contrario, la salida debe ser 1. Podemos expresar el comportamiento deseado mediante una tabla de verdad, como se muestra en la Figura 2,15.

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Figura 2.14 Función que va a sintetizar
Elaborado: El Autor

Un posible procedimiento para diseñar un circuito lógico que implementa la tabla de verdad es crear un término producto que tiene valor de 1 para cada valoración para el cual la función f salida tiene que ser 1. Entonces podemos tomar una suma lógica de estos términos de productos a darse cuenta de f. Comencemos con la cuarta fila de la tabla de verdad, lo que corresponde a $x1 = x2 = 1$. El término producto que es igual a 1 para esta valoración es $x1 \cdot x2$, que es sólo la Y de $X1\ y\ X2$. Consideremos ahora la primera fila de la tabla, por lo que $x1 = x2 = 0$. Para esta valoración el valor 1 es producida por el término de producto $x1 \cdot x2$. Del mismo modo, la segunda fila lleva a la expresión $x1 \cdot x2$. Así f puede ser realiza do como

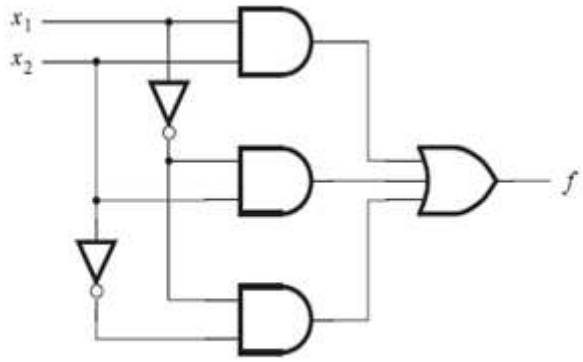
$$F(x1, x2) = x1x2 + \overline{x1x2} + \overline{x1}\ x2$$

La red lógica que corresponde a esta expresión se muestra en la Figura 2.16a. Aunque esta red implementa f correctamente, no es el más simple de red tal. A encontrar una red más sencilla, podemos manipular la expresión obtenida utilizándolos teoremas y las propiedades de la sección 2.5. De acuerdo con el teorema de 7b, podemos replicar cualquier término en una expresión de suma lógica. Replicar el término tercer producto, la expresión anterior se convierte en

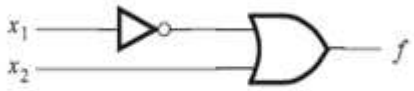
$$F(x1, x2) = x1x2 + \overline{x1x2} + \overline{x1}\ x2 + \overline{x1}x2$$

Uso de los 10b propiedad conmutativa para el intercambio de los términos de productos ofrece la segunda y tercera

$$F(x1, x2) = x1x2 + \overline{x1}x2 + \overline{x1\ x2} + \overline{x1}x2$$



a) Suma canonica de productos



b) Realizacion de costo minimo

Figura 2.15 Dos implementaciones
Elaborado: El Autor

Ahora, el 12a propiedad distributiva nos permite escribir

$$f(x_1, x_2) = (x_1 + \overline{x}_1)x_2 + \overline{x}_1(\overline{x}_2 + x_2)$$

Aplicando el teorema 8b que obtenemos

$$f(x_1, x_2) = 1 \cdot x_2 + \overline{x}_1 \cdot 1$$

Finalmente, 6a teorema conduce a

$$f(x_1, x_2) = x_2 + \overline{x}_1$$

2.6.1 FORMAS DE PRODUCTOS DE SUMAS

Tras exponer el proceso de síntesis con ejemplo muy simple ahora lo presentaremos en términos más formales empleando la terminología de la biografía técnica. También mostraremos como aplicar el principio de dualidad, explicando en la sección 2.5 en el proceso de síntesis

MINTERMINOS

Teniendo en cuenta que cada minitérmino sólo vale 1 para una combinación de las entradas, podemos asociar a cada fila de una tabla de verdad un único minitérmino que valdrá 1 sólo cuando las entradas tomen el valor de esa fila. (Muñoz Frías, 2012)

Para una función n variables, un término producto en la que cada una de la n variable aparezca una vez llamada minitérmino en forma sin complementar o complemento para una fila la tabla de verdad el minitérmino se forma incluyendo $si\ xi\ =\ 1\ y\ xi\ =0$

Para ilustrar este concepto considérese la tabla de verdad de la figura 2.17 las filas están numeradas del $0\ al\ 7$, con le podemos hacer referencia a ella con facilidad. (El lector familiarizado con la representación en números binarios notara que los números de la fila elegidos son justo los representados por lo patrones de bits de las variables $x1\ x2\ x3$

FORMAS DE SUMAS DE PRODUCTOS

Si una función dada se especifica mediante una tabla de verdad, entonces su complemento f puede ser representado por una suma de términos mínimos para los cuales $f\ =\ 1$, que son las filas donde $f\ =\ 0$. Para ejemplo, para la función en la Figura 2,15

$$f\ (x1, x2)\ =\ m2 =\ X1\ x2$$

Si complementamos esta expresión usando el teorema de De Morgan, el resultado es

$$f\ =\ f\ =\ x1\ x2 =\ X1\ +\ X2$$

Tenga en cuenta que hemos obtenido esta expresión con anterioridad por la manipulación algebraica de la canónica de suma de productos de forma para la función f. El punto clave aquí es que

$$f\ =\ m2\ =\ M2$$

2.7CICUITOS LOGICOS NAND Y NOR

Hemos discutido el uso de AND, OR y NOT en la síntesis de circuitos lógicos. Hay otras funciones lógicas básicas que también se utilizan para este propósito. Particularmente útil es la NAND y NOR las funciones que se obtienen complementando los resultados generados por operaciones AND y OR, respectivamente. Estas funciones son atractivas debido a que se implementan con circuitos electrónicos más simples que las funciones AND y OR, como veremos en el capítulo 3. Figura 2.20 da los símbolos gráficos para

la NAND y ÑOR. Una burbuja se coloca en el lado de salida del AND y OR símbolos de compuerta para representar la señal de salida complementada.

Si NAND y NOR cuenta con circuitos más sencillos que AND y OR, entonces debemos preguntarnos si estas puertas se pueden utilizar directamente en la síntesis de circuitos lógicos. En la sección 2.5 se introdujo teorema de De Morgan. Su interpretación puerta lógica se muestra en la Figura 2,21. 15a identidad se interpreta en la parte (a) de la figura. En él se especifica que una memoria NAND de variables $X1\ y\ X2$ es equivalente a la primera complementando cada una de las variables y luego ellos. Note que en la parte situada más a la derecha que nos han indicado las puertas no

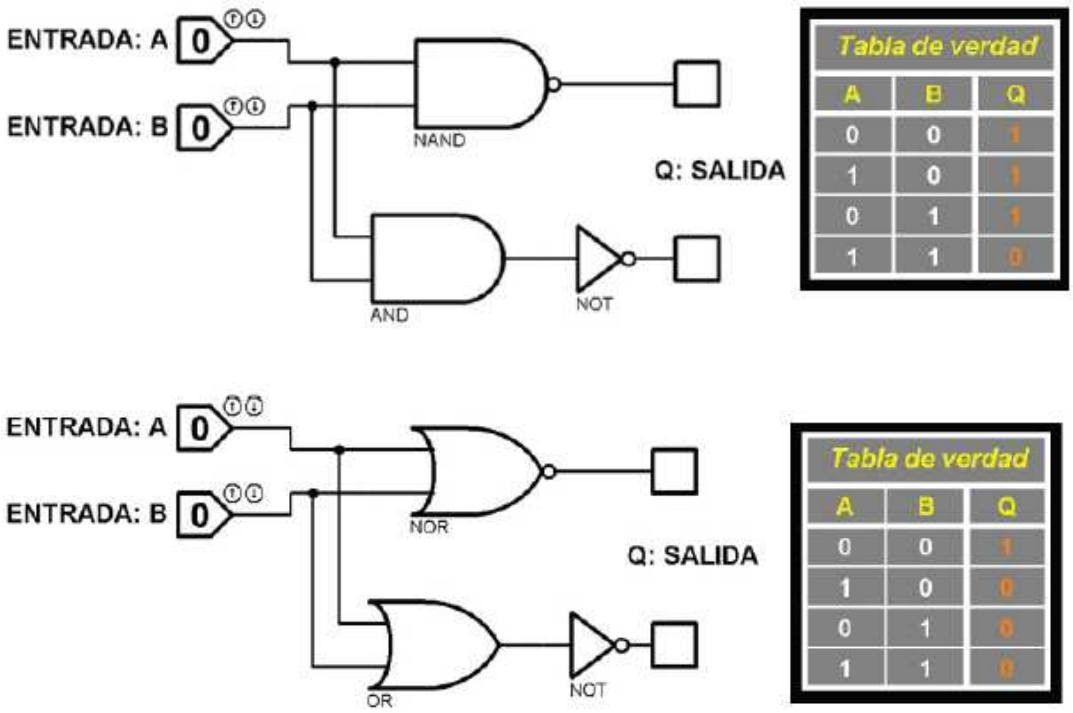


Figura 2.16 Compuertas NAND Y NOR
Elaborado: (M., semestre 2011-A)

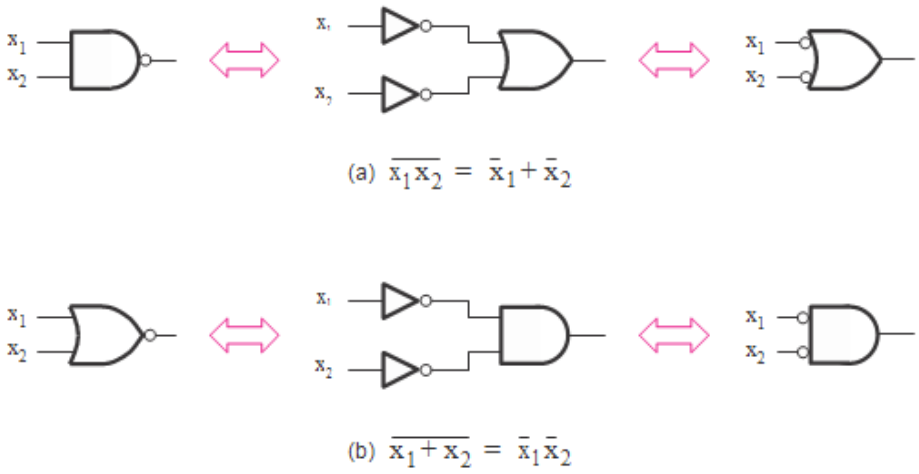


Figura 2.17 teorema de Morgan
Elaborado: El Autor

2.8 INTRODUCCIÓN A VHDL

En la década de 1980 los rápidos avances en tecnología de circuitos integrados para desarrollar las prácticas estándar de diseño de circuitos digitales. VHDL fue desarrollado como una parte de ese esfuerzo. VHDL se ha convertido en el lenguaje estándar de la industria para describir circuitos digitales, en gran parte debido a que es un funcionario de la norma IEEE. El estándar original para VHDL fue adoptado en 1987 y llamado IEEE 1076. Una norma revisada se aprobó en 1993 y llamado IEEE 1164.

Los diseños hardware presentan una gran cantidad de concurrencia, inherente a su modo normal de operación. El lenguaje VHDL refleja este hecho estructurando la descripción de cada diseño como una colección de procesos concurrentes, interconectados por señales. Cada proceso representa un componente de circuito (puertas, biestables, contadores, etc.). (Dopico, 2009)

Como funcionario estándar IEEE, VHDL proporciona una forma común de la documentación de los circuitos diseñados por numerosos diseñadores. En segundo lugar, VHDL proporcionan características para modelar el comportamiento de un circuito digital, lo que permitió como entrada a los programas de software que se utilizaron luego para simular el funcionamiento

del circuito. En los últimos años, además de su uso para la documentación y simulación, VHDL también ha hecho popular para su uso en la entrada de diseño en sistemas CAD. Las herramientas CAD se utilizan para sintetizar el código VHDL en una implementación de hardware del circuito descrito.

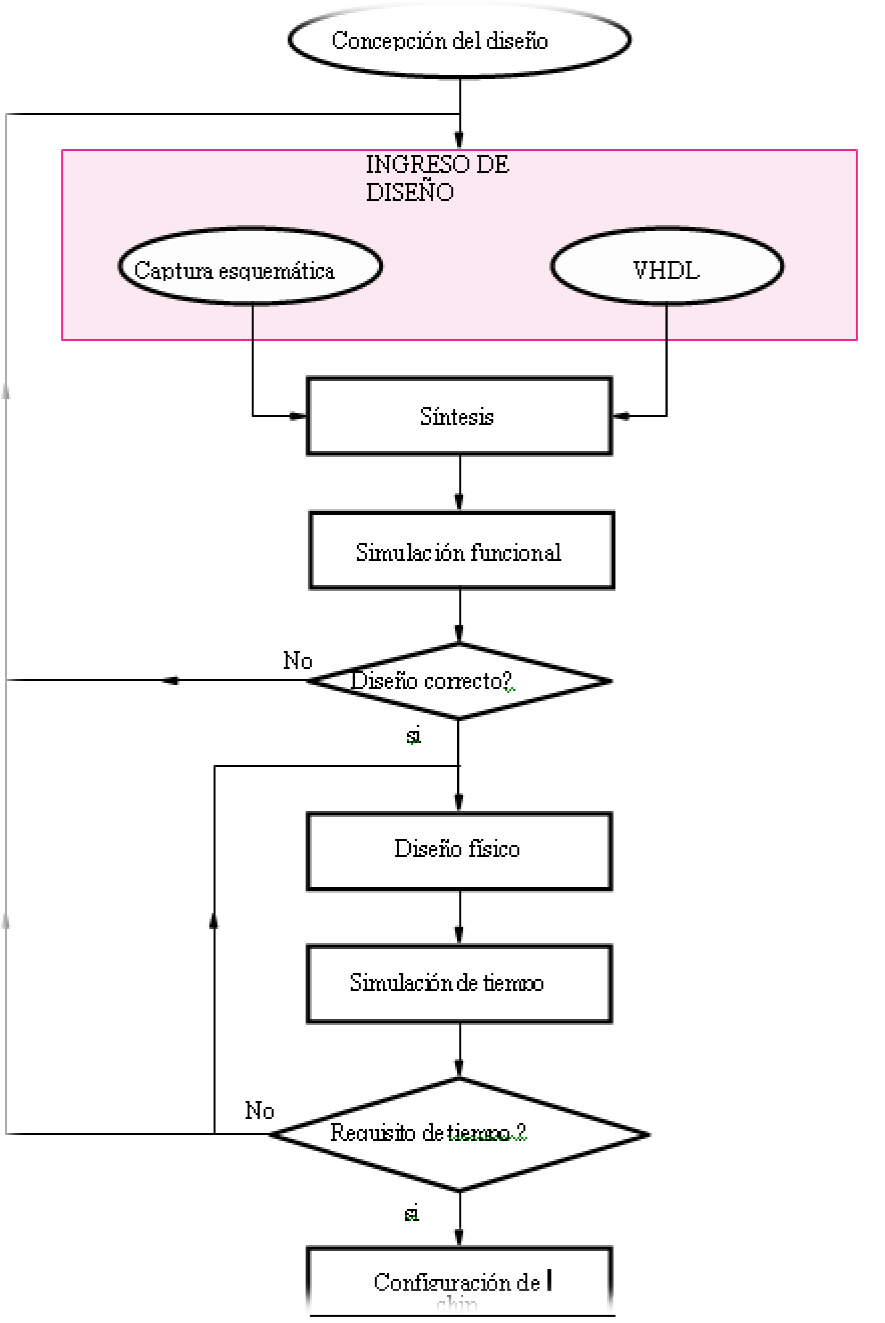


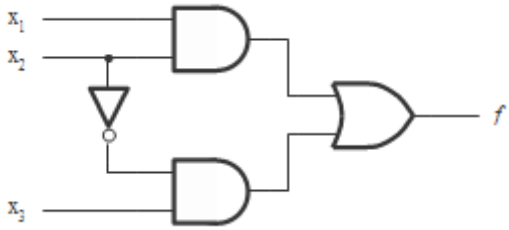
Figura 2.18: Sistema CAD típico
Elaborado: El Autor

2.8.1 REPRESENTACIÓN DE SEÑALES DIGITALES EN VHDL

Cuando el uso de herramientas CAD para sintetizar un circuito lógico, el diseñador puede proporcionar la descripción inicial del circuito de varias maneras diferentes, Una manera eficaz es escribir esta descripción en forma de código fuente VHDL. El compilador traduce el código VHDL en un circuito lógico. Cada señal lógica en el circuito se representa en VHDL código objeto ASA datos. Al igual que los tipos de las variables declaradas en cualquier lenguaje de programación de alto nivel se han asociado, como enteros o caracteres, los objetos de datos en VHDL puede ser de varios tipos. El estándar original VHDL, IEEE 1076, incluye un tipo de datos llamada TBI. Un objeto de este tipo es muy adecuado para la representación de las señales digitales ya que los objetos bit puede tener solamente dos valores, 0 y 1. En este capítulo todas las señales en nuestros ejemplos serán de tipo bit.

2.8.2 COMO ESCRIBIR CÓDIGO SENCILLO EN VHDL

Vamos a utilizar un ejemplo para ilustrar cómo escribir código simple fuente VHDL. Consideremos el circuito lógico de la figura 2,19. Si queremos escribir código VHDL para representar este circuito, el primer paso es declarar la entrada y salida. Esto se hace utilizando un constructo llamado una entidad.




```
ENTITY example1 IS
    PORT ( x1, x2, x3  : IN   BIT ;
           f           : OUT  BIT ) ;
END example1 ;
```

Figura 2.19: Función Lógica Simple
Elaborado: El Autor

Se le asigna un nombre, hemos elegido el nombre de ejemplo1, por ejemplo, esta primera. Las señales de entrada y salida de la entidad se llaman sus puertos, y se identifican por el Puerto de palabras clave. Este nombre deriva de la jerga eléctrica en la entrada o conexión de salida a un circuito electrónico. Cada puerto tiene un modo asociado que especifica si la entrada ISAN (IN) a la entidad o una salida (OUT) de la entidad. Cada puerto representa una señal, por lo tanto tiene un tipo asociado. Los puertos de la

entidad ejemplo1 cuatro en total. Los tres primeros, $x1, x2\ y\ x3,$ son señales de entrada de tipo bit. El puerto de salida f es de un tipo bit. En la Figura 2.19 se han utilizado los nombres de señal simple $x1, x2, x3,$ y f de la entidad. Al igual que en la mayoría de los lenguajes de programación, VHDL tiene reglas que especifican qué caracteres se permiten en los nombres de señal. Una pauta sencilla es que los nombres de señal puede incluir cualquier letra o número, carácter de subrayado --. Hay dos advertencias: a nombre de la señal debe comenzar con una letra y un nombre de señal no puede ser una palabra clave VHDL.Una entidad especifica la entrada y salida de un circuito , pero no se dan detalles en cuanto a lo que representa el circuito. La funcionalidad del circuito se debe especificar con una construcción VHDL llamada arquitectura. Una arquitectura para nuestra figura ejemplo, 2,19. Se le debe dar un nombre, y hemos elegido el nombre de LogicFunc. Aunque el nombre puede ser cualquier cadena de texto, es conveniente asignar un nombre que sea significativo para el diseñador.

En este caso hemos elegido el nombre de LogicFunc porque la arquitectura especifica la funcionalidad del diseño utilizando una expresión lógica. VHDL se ha incorporado en el apoyo a los siguientes operadores booleanos: AND, OR y NOT, NAND, NOR, XOR y XNOR. (Hasta ahora hemos introducido AND, OR y NOT, NAND, NOR y los operadores;) A raíz de la palabra clave BEGIN, nuestra arquitectura especifica, utilizando la señal de VHDL operador de asignación <=, que f de salida debe Se asigna el resultado de la expresión lógica en el lado derecho del operador. Debido a que VHDL no asume ninguna precedencia de operadores lógicos, se utilizan paréntesis en la expresión. Uno podría esperar que una sentencia de asignacióncomo:

$$f <= x1ANDx2ORNOTx2ANDx3$$

Implicara los paréntesis que siguen

$$f <= (x1ANDx2)OR((NOTx2)ANDx3)$$

Sin embargo, para el código VHDL este supuesto no es cierto. De hecho, sin los paréntesis del VHDL compilador genera un error en tiempo de compilación de esta expresión. En este ejemplo se ha puesto de manifiesto que una fuente de archivo de código VHDL tiene dos secciones principales: una entidad y una arquitectura.

$ARCHITECTURELogicFuncOF\ example1IS\ BEGIN$

$f <= (x1ANDx2)OR(NOTx2ANDx3);\ ENDLogicFunc;$

$ENTITYexample1IS$

$PORT(x1, x2, x3 : IN \quad BIT; f : OUTBIT);$

$ENDexample1;$

$ARCHITECTURELogicFuncOF\ example1IS$

$BEGIN$

$f <= (x1ANDx2)OR(NOTx2ANDx3);$

$ENDLogicFunc;$

$ENTITYexample2IS$

$PORT(x1, x2, x3, x4 : IN \quad BIT;$

$f, g \quad : OUTBIT)\ ;$

$ENDexample2;$

$ARCHITECTURELogicFuncOFexample2IS$

$BEGIN$

$f <= (x1ANDx3)OR(x2ANDx4);$

$g <= (x1ORNOTx3)AND(NOTx2ORx4); ENDLogicFunc;$

CAPITULO 3 TECNOLOGÍA DE IMPLEMENTACIÓN EN FPGA

3.1 HARDWARE DIGITAL

Los circuitos lógicos se utilizan para construir equipos informáticos, así como muchos otros tipos de productos. Todos estos productos son ampliamente clasificados como de hardware digital. La razón de que el nombre digital se utiliza se pondrá de manifiesto deriva de la forma en que la

información se representa en los ordenadores, como señales electrónicas que corresponden a los dígitos de información. La tecnología utilizada para construir el hardware digital ha evolucionado dramáticamente durante las últimas cuatro décadas. Hasta la década de 1960 los circuitos lógicos se construyen con componentes voluminosos, tales como transistores y resistencias que vienen como partes individuales. El advenimiento de los circuitos integrados hizo posible colocar un número de transistores, y por lo tanto un circuito completo, en un solo chip.

En el principio estos circuitos tenían sólo unos pocos transistores, pero como la mejora de la tecnología que se hizo más grande. Chips de circuitos integrados se fabrican en una oblea de silicio. La oblea se corta para producir los chips individuales, que luego son colocadas dentro de un tipo especial de paquete de chips. En 1970 se pudo poner en práctica toda la circuitería necesaria para realizar un microprocesador en un solo chip. A pesar de los primeros microprocesadores tenían capacidad de computación modesta para los estándares actuales, que abrió la puerta a la revolución de procesamiento de la información, proporcionando los medios para la aplicación de los ordenadores personales asequibles. Hace unos 30 años, Gordon Moore, presidente de Intel Corporation, señaló que la tecnología de circuito integrado se avanza a una velocidad asombrosa, duplicando el número de transistores que se podrían colocar en un chip cada 1,5 a 2 años. Este fenómeno, conocido informalmente como la ley de Moore, continúa hasta la actualidad (Wakerly, 2007)

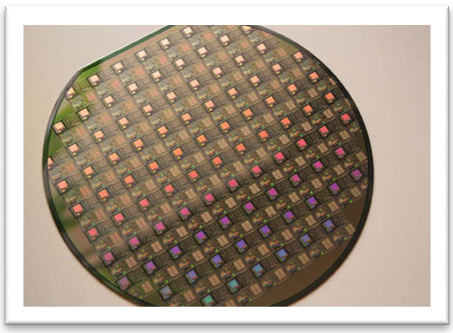


Figura 3.1: Oblea de silicio
Elaborado: sitio web (Ecolosfera)

3.1.1 CHIPS ESTANDAR

Numerosas fichas están disponibles que dan cuenta algunos circuitos lógicos comúnmente utilizados. Nos referiremos a ellos como los chips estándar, porque por lo general se ajustan a una norma acordada en términos de funcionalidad y la configuración física. Cada chip estándar contiene una pequeña cantidad de circuitos (por lo general implica menos de 100 transistores) y realiza una función simple.

Para construir un circuito lógico, el diseñador elige los chips que llevan a cabo las funciones de lo que sea necesario y luego se define la forma en que estos chips deben estar interconectados para realizar un circuito lógico más amplio. Fichas estándar eran muy populares para la construcción de circuitos lógicos hasta principios de 1980. Sin embargo, como la tecnología de circuito integrado mejorado, se convirtió en ineficaz de usar el espacio valioso en los PCB de los chips con baja funcionalidad. Otro inconveniente de los chips estándar es que la funcionalidad de cada chip es fija y no se puede cambiar

3.1.2 DISPOSITIVO LOGICO PROGRAMABLE

En contraste con los chips estándar que tienen funcionalidad fija, es posible construir chips que contienen circuitos que pueden ser configurados por el usuario para aplicar una amplia gama de Figura.

Figura 3.2: Chips FPGA
Elaborado: sitio web (Govoid)

Un chip de compuertas programables en campo array (por cortesía deAltera Corp.).

Diferentes circuitos lógicos. Estos chips tienen una estructura muy general e incluyen una colección de interruptores programables que permiten la circuitería interna en el chip para ser configurado de muchas maneras diferentes. El diseñador puede implementar cualquier funciones son necesarias para una aplicación particular, por la elección de una configuración adecuada de los interruptores. Los interruptores están programados por el usuario final, en vez de cuando el chip está fabricado. Estos chips se conocen como dispositivos de lógica programable (PLD). La mayoría de los tipos de PLDs se puede programar varias veces. Esta capacidad es una ventaja, porque un diseñador que está desarrollando un prototipo de un producto puede programar un PLD para realizar alguna función, pero más tarde, cuando el hardware prototipo está siendo probado, puede hacer las correcciones mediante la reprogramación del PLD.

La reprogramación puede ser necesario, por ejemplo, si una función diseñada no está del todo según lo previsto o si se necesitan nuevas funciones que no se contemplaron en el diseño original.PLDs están disponibles en una amplia gama de tamaños. Se pueden usar para lograr mucho más grandes circuitos lógicos de un chip estándar típico puede realizar. Debido a su tamaño y el hecho de que puede ser adaptada para satisfacer los requisitos de una aplicación específica, PLDs son ampliamente utilizados hoy en día.

Un dispositivo lógico programable o PLD es un nombre genérico para designar un circuito integrado digital capaz de ser programado para proporcionar diferentes funciones lógicas. Los dispositivos PLD más complejos pueden contener miles de compuertas lógicas y biestables. Por lo tanto un PLD sencillo puede reemplazar un gran número de circuitos integrados (roth, 2009)

Uno de los tipos más sofisticados de la PLD se conoce como un arreglo de compuertas programables en campo (FPGA). FPGAs que contienen varios cientos de millones de transistores están disponibles [2, 3]. Una fotografía de un chip FPGA se muestra en la Figura 3,2. El chip se compone de un gran número de pequeños elementos lógicos de circuitos, que pueden ser conectados entre sí mediante los interruptores programables. Los elementos del circuito de lógica están dispuestos en una ordinaria estructura bidimensional

3.1.3 CHIPS DISEÑADO A LA MEDIDA

PLD están disponibles como componentes fuera de la plataforma que se pueden comprar. Debido a que son programables, que pueden ser utilizados para poner en práctica la mayoría de los circuitos lógicos que se encuentran en hardware digital. Sin embargo, PLDs también tienen el inconveniente de que los interruptores programables consumir área del chip valioso y limitar la velocidad de funcionamiento de los circuitos implementados. Así, en algunos casos PLD no cumpla con el rendimiento deseado, o los objetivos de costo. En tales situaciones, es posible diseñar un chip desde cero, es decir, la circuitería lógica que debe ser incluida en el chip está diseñado primero y luego una tecnología apropiada es elegida para implementar el chip. Por último, el chip es fabricado por una empresa que cuenta con las instalaciones de fabricación. Este enfoque se conoce como diseño personalizado o semi-personalizadas, y estos chips son llamados chips personalizados o semi-personalizadas.

Estos chips están diseñados para su uso en aplicaciones específicas, y son a veces llamados específicos de la aplicación de circuitos integrados (ASIC).La principal ventaja de un chip es que su diseño puede ser optimizado para una tarea específica, por lo que por lo general conduce a un mejor rendimiento. Es posible incluir una cantidad mayor de circuitos lógicos en un chip personalizado que sería posible en otros tipos de fichas. El costo de producción de estos chips es alta, pero si se utilizan en un producto que se vende en grandes cantidades, entonces el costo por chip, amortiza en el número total de fichas fabricadas, puede ser menor que el coste total de fuera de la -plataforma chips que serían necesarias para poner en práctica la misma función (s). Además, si un solo chip se puede utilizar en lugar de múltiples chips para lograr el mismo objetivo, a continuación, un área más pequeña que se necesita en una PCB que alberga las virutas en el producto final.

Esto resulta en una reducción adicional de costo.Una desventaja del enfoque personalizado diseño es que la fabricación de un chip personalizado a menudo toma una cantidad considerable de tiempo, del orden de meses. En contraste, si un PLD se puede utilizar en su lugar, a continuación, las virutas

son programados por el usuario final y sin retrasos de fabricación están implicados

3.2 EL PROCESO DE DISEÑO

La disponibilidad de las herramientas informáticas ha influido grandemente el proceso de diseño en una amplia variedad de entornos de diseño. Por ejemplo, el diseño de un automóvil es similar en el enfoque general para el diseño de un horno o un ordenador. Algunos pasos en el ciclo de desarrollo debe llevarse a cabo si el producto final es cumplir con los objetivos especificados. Vamos a empezar con la introducción de un ciclo típico de desarrollo en los términos más generales. A continuación, nos centraremos en los aspectos particulares que atañen al diseño de circuitos lógicos. El diagrama de flujo en la figura se describe un proceso típico de desarrollo. Suponemos que el proceso es desarrollar un producto que cumpla con ciertas expectativas.

Los requisitos más evidentes son que el producto debe funcionar correctamente, que debe cumplir con un nivel de desempeño esperado, y que su costo no debe exceder de un objetivo determinado. El proceso comienza con la definición de las especificaciones del producto. Las características esenciales del producto son identificados, y un método aceptable de evaluar las características implementadas en el producto final es establecido.

Las especificaciones deben ser lo suficientemente apretado para asegurar que el producto desarrollado a cumplir con las expectativas generales, pero no debe ser innecesariamente restrictivo (es decir, las especificaciones no debe impedir que las opciones de diseño que pueden conducir a ventajas imprevistas). Desde un conjunto completo de especificaciones, es necesario definir la estructura general de un diseño inicial del producto. Este paso es difícil de automatizar. Se realiza generalmente por un diseñador humano, porque no hay una estrategia clara para el desarrollo de la estructura general de un producto que requiere una considerable experiencia en el diseño y la intuición.

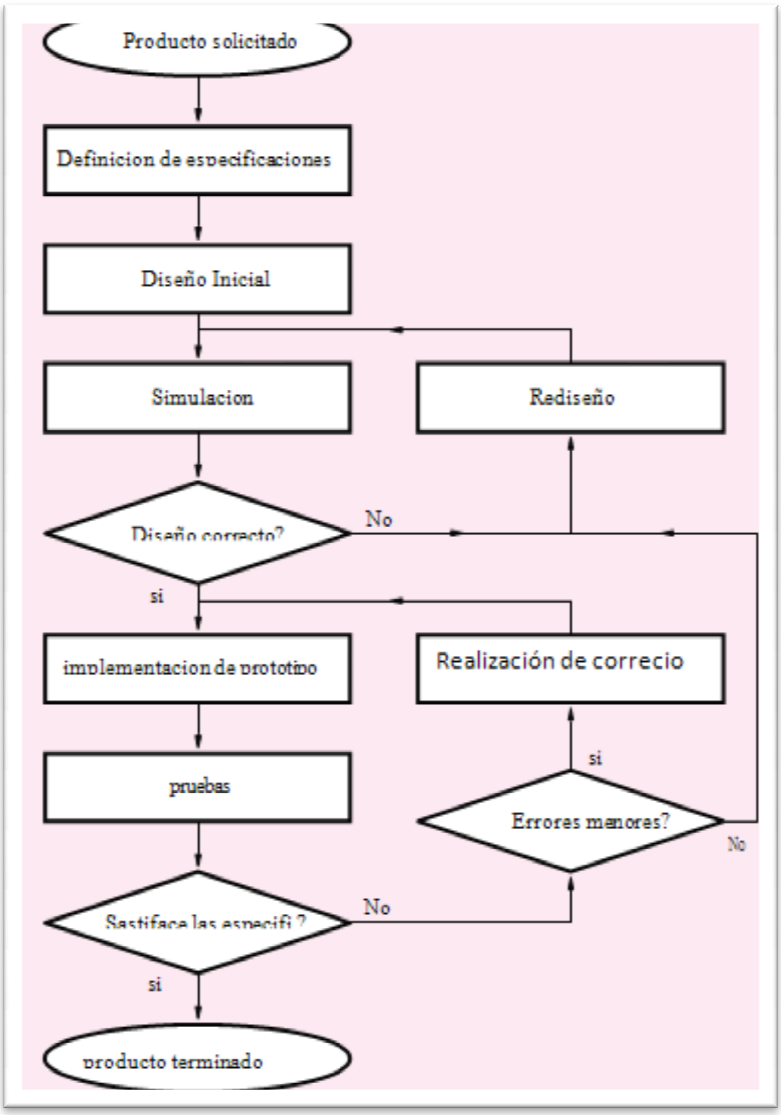


Figura 3.3: El proceso de desarrollo
Elaborado: El Autor

3.3 DISEÑO DE HARDWARE DIGITAL

Nuestra discusión anterior sobre el proceso de desarrollo es relevante en una forma más general. Los pasos descritos en la figura son plenamente aplicables en el desarrollo de hardware digital. Antes de discutir la secuencia completa de los pasos en este entorno de desarrollo, hay que destacar la naturaleza iterativa del proceso de diseño.

3.3.1 CICLO DE DISEÑO BASICO

Cualquier proceso de diseño comprende una secuencia básica de las tareas que se realizan en diversas situaciones. Esta secuencia se presenta en

la Figura Suponiendo que tenemos un concepto inicial de lo que debe conseguirse en el proceso de diseño, el primer paso consiste en generar un diseño inicial. Este paso a menudo requiere un gran esfuerzo manual, porque la mayoría de los diseños tienen algunos de los objetivos específicos que sólo se puede llegar a través del conocimiento del diseñador, habilidad, y la intuición. El siguiente paso es la simulación del diseño en la mano. (Tocci, Ronald.dj. Y Widmer,NEal.S., 2009)

Existen excelentes herramientas de CAD para ayudar en este paso. Para llevar a cabo la simulación con éxito, es necesario tener las condiciones adecuadas de entrada que se pueden aplicar al diseño que se está simulando y más tarde para el producto final que tiene que ser probado. La aplicación de estas condiciones de entrada, el simulador intenta comprobar que el producto diseñado se realiza como se requiere en las especificaciones del producto original. Si la simulación revela algunos errores, a continuación, el diseño debe ser cambiado para superar los problemas. La versión rediseñada es de nuevo simulado para determinar si los errores han desaparecido. Este bucle se repite hasta que la simulación indica un diseño exitoso. Un diseñador prudente dedica un esfuerzo considerable para poner remedio a los errores durante la simulación

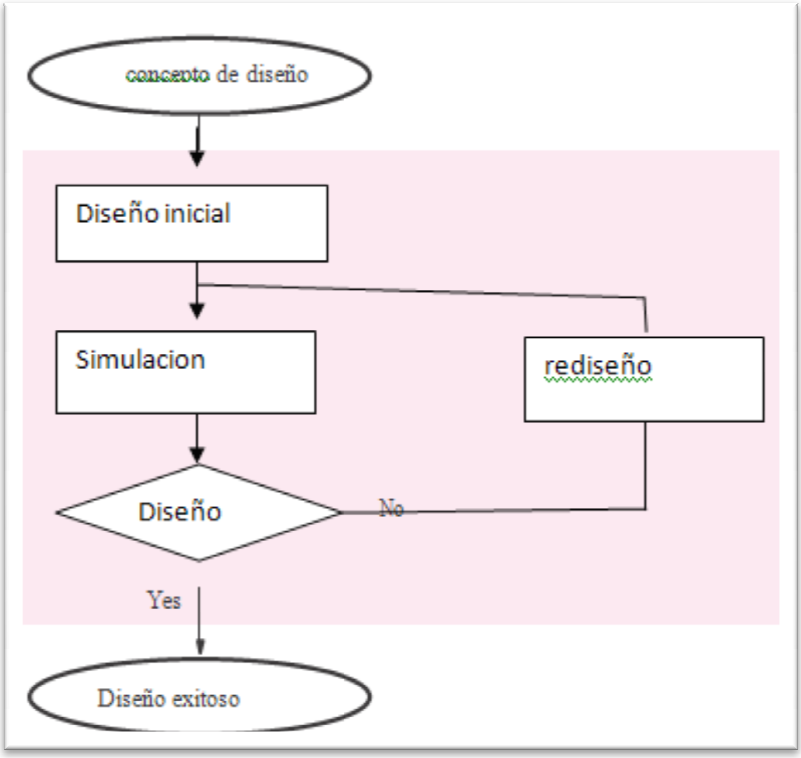


Figura 3.4: Ciclo de diseño básico

Elaborado: El Autor

Porque estos suelen ser mucho más difícil de corregir si se descubre tarde en el proceso de diseño aun así algunos de ellos no se detectan en la simulación por que hay que enfrentarlos en etapas posteriores del ciclo de desarrollo

3.3.2 ESTRUCTURA DE UNA COMPUTADORA

Para entender el papel que juegan los circuitos lógicos en los sistemas digitales, considerar la estructura de un equipo típico, como se ilustra en la Figura la carcasa del ordenador tiene un número de placas de circuito impreso (PCB), una fuente de alimentación, y (no se muestra en la figura) unidades de almacenamiento, como un disco duro y DVD o CD-ROM drives. Cada unidad está conectada a una placa principal, llamada de la placa base. Como se indica en la parte inferior de la figura, la placa base tiene varios chips de circuitos integrados, y que incluye ranuras para conectar los PCB, tales como audio, vídeo y tarjetas de red. La figura ilustra la estructura de un chip de circuito integrado. (Wakerly, 2007)

El chip comprende un número de subcircuitos, que están interconectadas para construir el circuito completo. Ejemplos de subcircuitos son aquellos que realizan operaciones aritméticas, almacenar datos, o controlar el flujo de datos. Cada uno de estos subcircuitos es un circuito lógico. Como se muestra en el centro de la figura, un circuito lógico comprende una red de puertas lógicas conectadas. Cada puerta lógica realiza una función muy sencilla, y las operaciones más complejas se realizan mediante la conexión de puertas juntas.

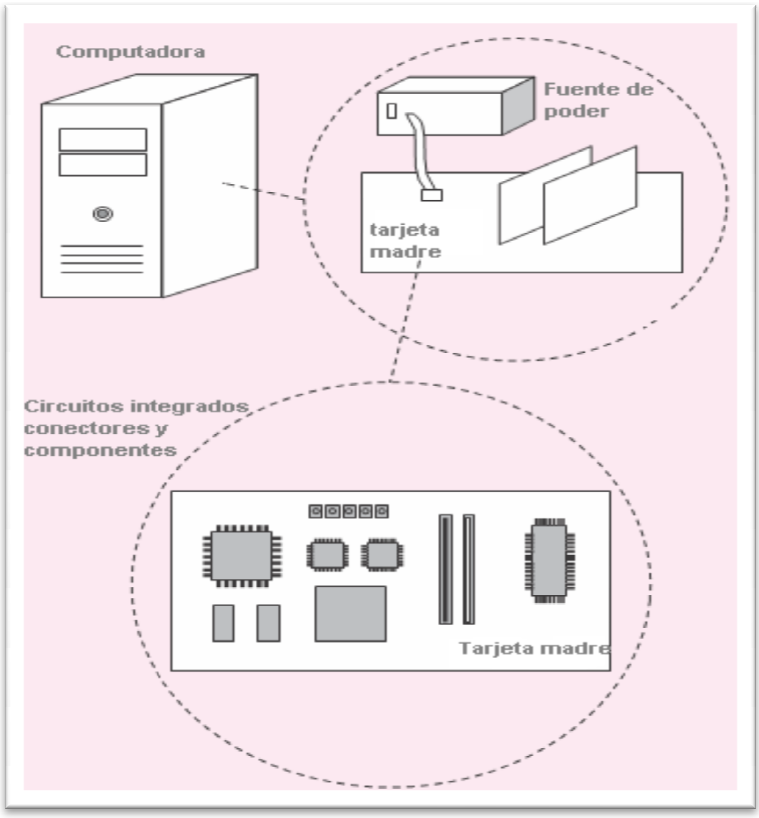


Figura 3.5: Hardware digital
Elaborado: autor y sitio web (blogspot)

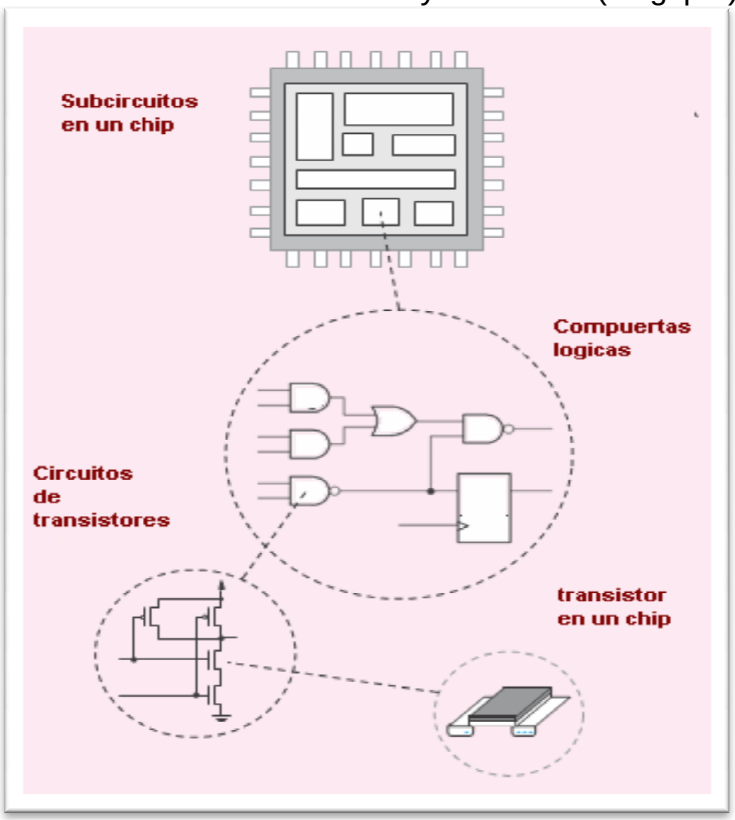


Figura 3.6: Hardware digital
Elaborado: autor y sitio web (blogspot)

Las puertas lógicas están construidas con transistores, que a su vez se implementan mediante la fabricación de varias capas de material sobre un chip de silicio. Explicamos cómo diseñar circuitos que realizan funciones importantes, tales como sumar, restar, multiplicar o los números, contar, el almacenamiento de datos, y controlar el procesamiento de la información. Se muestra cómo el comportamiento de estos circuitos se especifica, como los circuitos están diseñados para un costo mínimo o la velocidad máxima de operación, y cómo los circuitos pueden ser probados para garantizar su correcto funcionamiento. Asimismo, explicar brevemente cómo funcionan los transistores, y la forma en que se basan en chips de silicio

3.3.3 DISEÑO DE UNA UNIDAD DE HARDWARE DIGITAL

Como se muestra en la figura. Productos digitales de hardware suelen incluir uno o más PCB que contienen muchos chips y otros componentes. El desarrollo de estos productos se inicia con la definición de la estructura global. A continuación, los chips de circuitos integrados necesarios se seleccionan, y de los PCB que se casa y se conectan los chips están diseñados en conjunto. Si las fichas seleccionadas incluyen PLD o chips personalizados, a continuación, estos chips deben ser diseñados antes de que el diseño de PCB nivel se lleve a cabo. Dado que la complejidad de los circuitos implementados en los chips individuales y en las placas de circuito es generalmente muy alta, es esencial para hacer uso de buenas herramientas de CAD.

Una fotografía de una PCB se da en la figura. El PCB es una parte de un sistema informático grande diseñado en la Universidad de Toronto. Este ordenador, denominado NUMACHINE[4,5], es un multiprocesador, lo que significa que contiene muchos procesadores que se pueden utilizar para trabajar en una tarea particular. El PCB en la figura contiene un chip de procesador y chips de memoria diferentes y el apoyo. Circuitos lógicos complejos son necesarios para formar la interfaz entre el procesador y el resto del sistema. Un número de PLDs se utilizan para aplicar estos circuitos lógicos

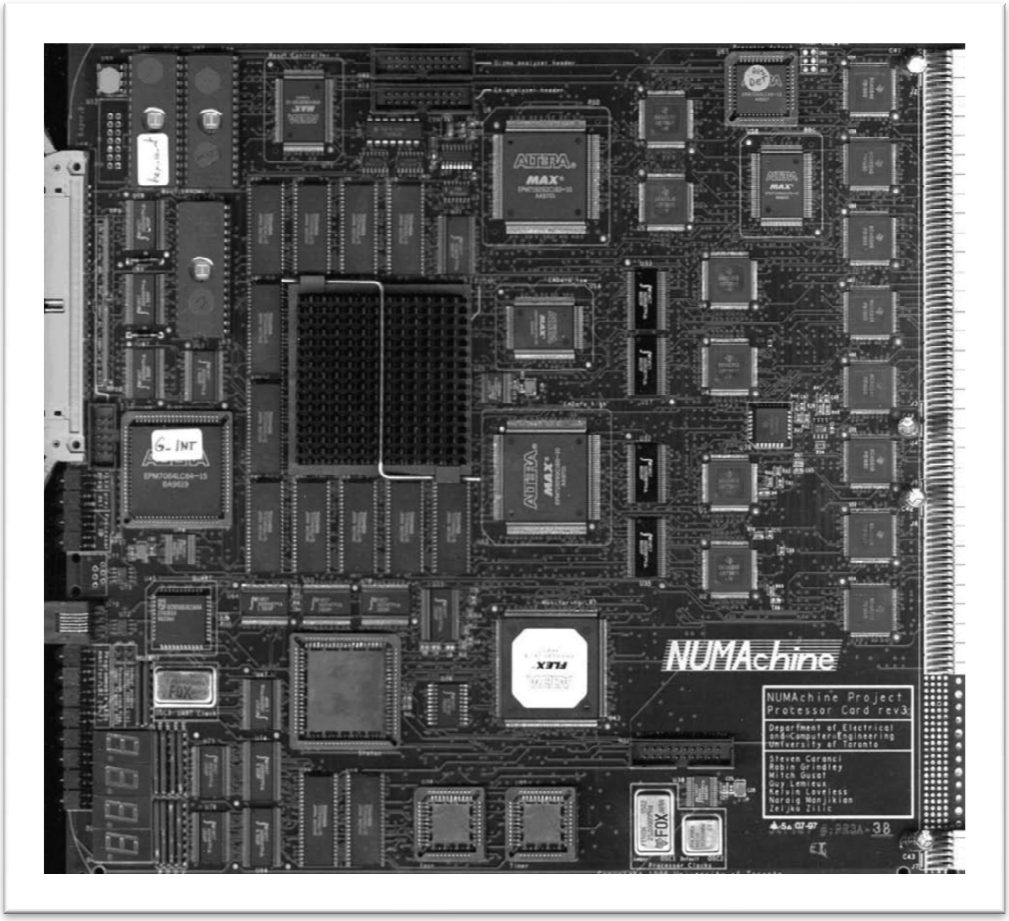


Figura 3.7: Tarjeta de circuito impreso
Elaborado: sitio web (blogspot)

El circuito global de gran tamaño en bloques no se ha hecho así, en cuyo caso la vía A está seguida. Esto puede ocurrir, por ejemplo, si ninguno de los bloques de implementar alguna funcionalidad necesaria en el circuito completo.

3.4 DISPOSITIVO LOGICOS PROGRAMABLES

3.4.1 ARREGLO LOGICO PROGRAMABLE (PLA)

Varios tipos de PLDs están disponibles comercialmente. La primera se desarrolló fue la matriz lógica programable (PLA). La estructura general de una PLA. Basado en la idea de que las funciones lógicas pueden realizarse en forma de suma de productos forma, una PLA comprende una colección de puertas Y que conjunto de puertas OR. Como se muestra en la figura, las entradas del EPL pase a través de un conjunto de buffers (que proporcionan tanto el valor verdadero y el complemento de cada entrada) en un bloque de circuito llamado un plano o matriz. El avión y produce un

conjunto de productos P1 términos,..., Pk. Cada uno de estos términos se puede configurar para poner en práctica toda y cada función de Los términos de productos sirven como entrada a un plano O, lo que produce la f1 salidas . Cada salida se puede configurar para realizar cualquier suma de y por lo tanto una función de suma de productos de la EPL entradas.

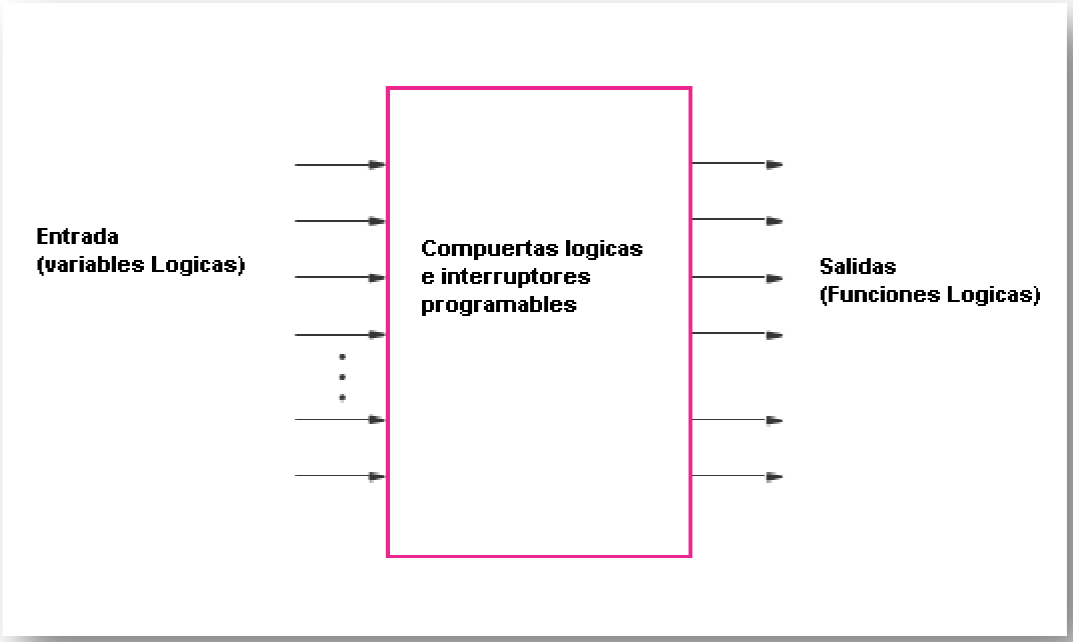


Figura 3.8: Caja negra (PLD)
Elaborado: el Autor

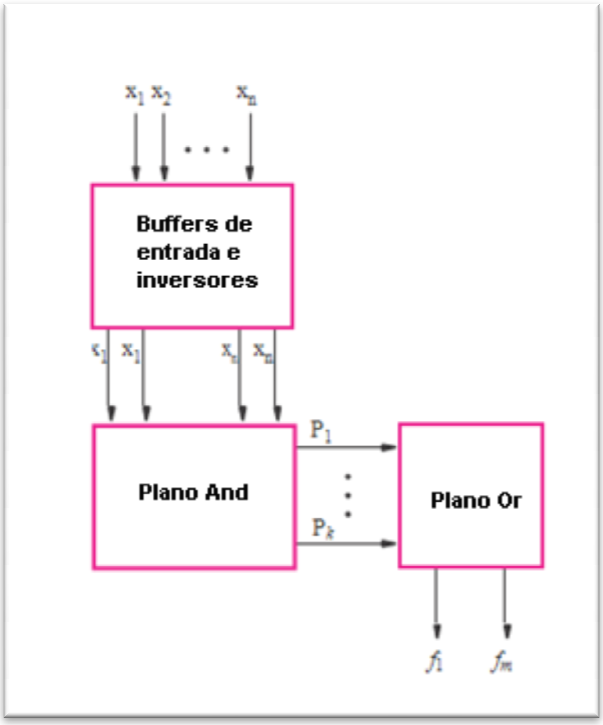


Figura 3.9: Estructura general de un PLA
Elaborado: el Autor

Un diagrama más detallado de una pequeña PLA se da en la figura 3,9, que muestra una PLA con tres entradas, cuatro términos de producto, y dos salidas. Cada puerta Y en el plano Y tiene seis entradas, correspondientes a las versiones verdaderas y complementado de las tres señales de entrada. Cada conexión a una puerta Y es programable; una señal que está conectada a una puerta Y se indica con una línea ondulada, y una señal que no está conectada a la puerta se muestra con una línea discontinua.

El circuito está diseñado de tal manera que cualquier puerta o entradas no afectan a la salida de la puerta AND. En PLA comercialmente disponibles, varios métodos de realizar las conexiones programables existe. Explicación detallada de cómo un PLA se puede construir utilizando transistores que produce P1 se muestra conectado a la entradas. Por lo tanto . Del mismo modo, Conexiones programables también existen para el avión o. F1 de salida está conectado al producto términos, la función Asimismo, la producción Por la programación de AND y OR aviones de manera diferente, cada uno de las salidas de podría poner en práctica las diversas funciones de

$x1, x2\ yx3$. La única limitación en las funciones que se pueden implementar es el tamaño, porque los términos de productos producen solo cuatro variables . PLA disponibles en el mercado vienen en tamaños más grandes de lo que hemos mostrado aquí. Los parámetros típicos son 16 entradas, 32 términos de productos, y ocho salidas.

3.4.2 LOGICA DE ARREGLO PROGRAMABLE

En una PLA tanto la AND y OR planos son programables. Históricamente, los interruptores programables presenta dos dificultades para los fabricantes de estos dispositivos: eran difíciles de fabricar correctamente, y se reduce la velocidad de rendimiento de los circuitos implementados en los PLA. Estos inconvenientes conducido al desarrollo de un dispositivo similar en el que el plano y es programable, pero el plano O se fija. Tal un chip que se conoce como una matriz lógica programable (PAL) del dispositivo. Debido a que son más fáciles de fabricar, y por lo tanto menos costosa que la PLA, y ofrecen un mejor rendimiento, PAL se han hecho populares en las aplicaciones prácticas.

Un ejemplo de una PAL con tres entradas, cuatro términos de producto, y dos salidas se da en la figura. El producto $P1\ y\ P2$ términos están conectadas a una puerta $OR, y\ P3\ y\ P4$ están programados para la otra puerta OR. El PAL se muestra programado para realizar las dos funciones lógicas $f1 = x1\ x2\ x3\ +\ x1\ x2\ x3\ y\ f2\ =\ x1\ x2\ +\ x1\ x2\ x3$

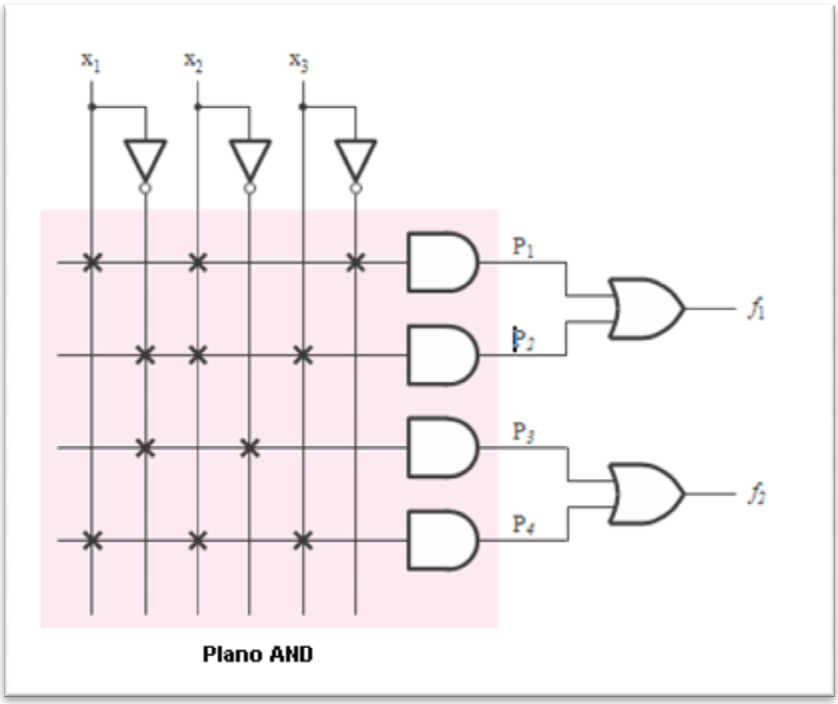


Figura 3.10: Esquema usual para el PLA
Elaborado: El Autor

En la que como las puertas OR en la PAL tienen sólo dos entradas. Para compensar la flexibilidad reducida, PALs esfabricado en una amplia gama de tamaños, con diversos números de entradas y salidas, y diferentes números de las puertas OR. Hasta ahora hemos asumido que el OR en un PAL, como en un PLA, se conectan directamente a los pines de salida del chip. En muchos PALs circuitería adicional se añade a la salida de cada puerta O para proporcionar flexibilidad adicional. Es habitual utilizar la macrocélula término para referirse a la puerta OR combinado con la circuitería adicional

3.4.3 PROGRAMACION DE PLA Y PAL

Cada Conexión Entre una Señal Lógica en la PLA o PAL y El y / o puertas es como una Muestra X. Se describe cómo estos interruptores se implementan utilizando transistores en la sección 3.10. Circuitos de los usuarios son implementadas en los dispositivos mediante la configuración, o programación, interruptores de estos. Comerciales contienen chips de unos pocos miles de interruptores programables, por lo que no es posible para un usuario de estos chips para especificar manualmente el estado deseado de programación de cada interruptor. En su lugar, los sistemas CAD se emplean

para este fin. Hemos introducido las herramientas CAD en el capítulo 2 y se describen métodos para la entrada de diseño y simulación de circuitos. Para los sistemas CAD que soportan la orientación de los circuitos a PLDs, las herramientas tienen la capacidad de producir automáticamente la información necesaria para la programación de cada uno de los interruptores en el dispositivo.

Por un cable a una unidad dedicada a la programación CAD. Una vez que el usuario ha completado el diseño de un circuito, el CAD genera un archivo, a menudo llamado un mapa de programación archivo o fusible, que especifica el estado que cada interruptor en el PLD debe tener, para realizar correctamente el circuito diseñado.

Figura 3.11: Unidad de programación PLD
Elaborado: sitio web (blogspot)

3.4.4 DISPOSITIVO LOGICOS PROGRAMABLE COMPLEJOS (CPLD)

PLA y PALS son útiles para la aplicación de una amplia variedad de pequeños circuitos digitales. Cada dispositivo se puede utilizar para implementar circuitos que no requieren más que el número de entradas, términos de productos, y las salidas que se proporcionan en el chip en particular. Estos chips están limitados a los tamaños relativamente modestos, típicamente apoyan un número combinado de entradas más salidas de no más

de 32. Para la implementación de los circuitos que requieren más entradas y salidas, ya sea PLA múltiples o PALS se puede emplear o bien un tipo más sofisticado de chip, llamado un complejo dispositivo de lógica programable (CPLD), se puede utilizar. Un CPLD comprende múltiples circuito de un solo chip, con cableado interno recursos para conectar los bloques de circuitos. Cada bloque de circuito es similar a una PLA o PAL;PAL como bloques. Un ejemplo de un CPLD se da en la figura.

Incluye cuatro como PAL-bloques que están conectados a un conjunto de cables de interconexión. Cada bloque PAL-como también está conectado a un subcircuitos etiqueta de E / S de bloques, que se ha unido a un número de entrada del chip y pines de salida.

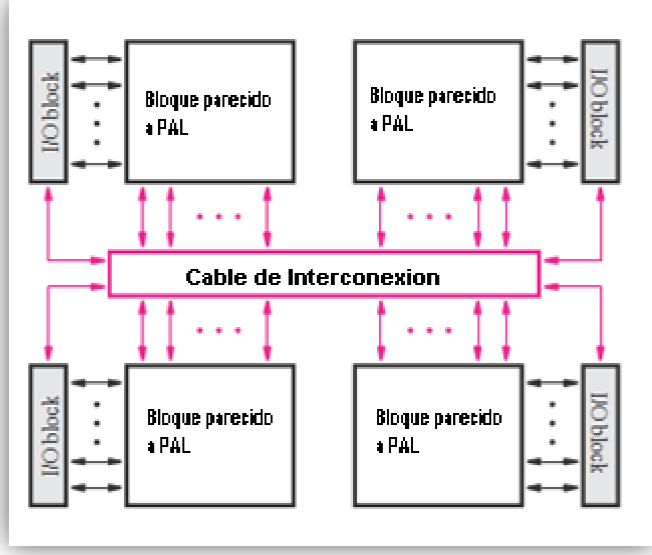


Figura 3.12 Estructura de un CPLD
Elaborado: Autor y web (blogspot-circuitoslogicos.geovanni, 2010)

La figura muestra un ejemplo de la estructura de cableado y las conexiones a un bloque PAL-como en un CPLD. El bloque de PAL-al igual que incluye 3 macrocélula (CPLDs reales suelen tener alrededor de 16 macrocélula en un bloque PAL), cada uno con una de cuatro entradas. Puerta OR (CPLDs reales suelen ofrecer entre 5 y 20 entradas de cada compuerta OR) La salida de la puerta O- está conectado a otro tipo de puerta lógica que todavía no han introducido. Se le llama una puerta OR exclusivo (XOR). Se discuten las puertas XOR en la sección. El comportamiento de una puerta XOR es la misma

ASFOR una puerta OR, excepto que si las dos condiciones de entrada 1, la puerta XOR

3.4.5ARREGLOS DE COMPUERTAS DE CAMPO PROGRAMABLE

Los tipos de chips descritos anteriormente, la serie 7400, SPLDs, y CPLDs, son útiles para la aplicación de una amplia gama de circuitos lógicos. A excepción de CPLDs, estos dispositivos son bastante pequeños y son adecuados sólo para aplicaciones relativamente simples. Incluso para CPLDs, sólo moderadamente grandes circuitos lógicos se pueden acomodar en un solo chip. Por razones de coste y rendimiento, es prudente implementar un circuito lógico deseado utilizando como chips de menor número posible, por lo que la cantidad de circuitos en un chip dado y su capacidad funcional son importantes. Una manera de cuantificar el tamaño de un circuito es suponer que el circuito se va a construir utilizando sólo puertas lógicas simples y luego estimar cómo muchas de estas puertas se necesitan.

Una medida comúnmente utilizada es el número total de dos entradas puertas NAND que sería necesario para construir el circuito; esta medida se llama a menudo el número de puertas equivalentes. Utilizando el equivalente métrico-puertas, el tamaño de un chip de 7400-serie es simple de medir porque cada chip contiene sólo puertas simples. Para SPLDs y CPLDs la medida típica utilizada es que cada macrocélula representa alrededor de 20 puertas equivalentes. Así, un PAL típica que tiene ocho macrocélula puede acomodar un circuito que necesita hasta alrededor de 160 puertas, y una CPLD grande que has500 macrocélula puede aplicar circuitos hasta aproximadamente 10.000 puertas equivalentes.

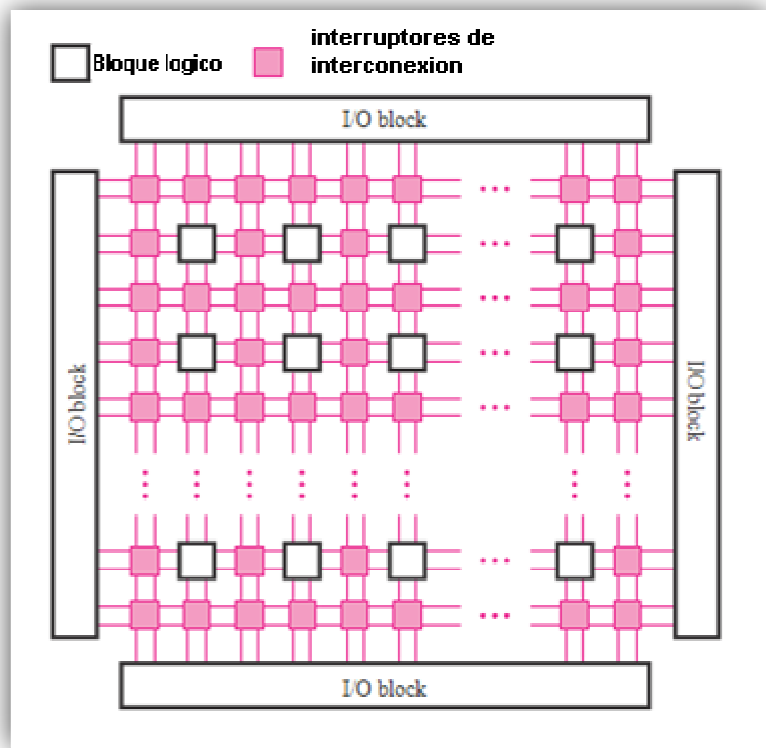


Figura 3.13: Estructura general (FPGA)
Elaborado: sitio web (blogspot)

Según el criterio moderno, un circuito lógico con 10.000 puertas no es muy grande. Para implementar grandes circuitos, es conveniente utilizar un tipo diferente de chip que tiene una capacidad lógica más grande. Un campo de arreglo de compuertas programables (FPGA) isa dispositivo lógico programable que apoye la implementación de circuitos lógicos relativamente grandes. FPGAs son muy diferentes de SPLDs CPLDs y FPGAs, porque no contienen AND y OR aviones. En cambio, los FPGAs proporcionan bloques lógicos para la ejecución de las funciones requeridas.

Contiene tres tipos principales de recursos: bloques lógicos, bloques I / O para conectar a los pines del paquete, y los cables de interconexión y los interruptores. Los bloques lógicos están dispuestos en una matriz de dos dimensiones, y los cables de interconexión están organizados como canales de encaminamiento horizontal y vertical entre las filas y bloques columnas lógicas. Los canales de encaminamiento contienen bus interruptores programables que permiten que los bloques lógicos para ser interconectados de muchas maneras. Las cajas azules adyacentes a los bloques lógicos tienen interruptores que conectan la entrada del bloque de la lógica y los terminales

de salida a los cables de interconexión, y las cajas azules que se encuentran en diagonal entre los bloques lógicos se conectan un cable de interconexión a otro (tal como un alambre vertical para un alambre horizontal). Conexiones programables también existen entre los bloques de E / S y los cables de interconexión.

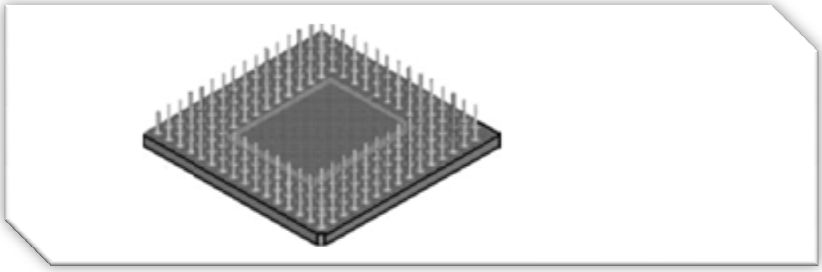


Figura 3.14: Paquete de arreglo (PGA)
Elaborado: sitio web (blogspot)

3.4.6 APLICACIONES DE LOS CPLD Y FPGA

CPLDs y FPGAs se utilizan hoy en día en muchas aplicaciones diversas, tales como productos de consumo como reproductores de DVD y de alta gama televisores, circuitos de controladores para las fábricas de automóviles y equipos de prueba, los routers de Internet de alta velocidad y switches de red, y gran equipo como el equipo cinta y sistemas de almacenamiento en disco.

En una situación de diseño dado un CPLD puede ser elegido cuando el circuito se necesita no es muy grande, o cuando el dispositivo realiza la función inmediatamente después de la aplicación de energía al circuito. FPGAs no es una buena opción para este último caso porque, mencionado antes, que se configuran los elementos de almacenamiento volátiles que pierden sus contenidos almacenados cuando el poder está apagado. (Tocci, Ronald.dj. Y Widmer,NEal.S., 2009)

Esto se traduce en un retraso de propiedad antes de que el chip FPGA puede realizar su función cuando se enciende. FPGAs son adecuados para la aplicación de los circuitos sobre una amplia gama de tamaño, desde aproximadamente 1000 a más de un millón de puertas lógicas equivalentes. Además del tamaño de un diseñador tendrá en cuenta otros criterios, tales como la velocidad necesaria de funcionamiento de un circuito, las limitaciones de potencia de disipación, y el coste de los chips. Cuando FPGAs no cumplen

con uno o más de los requisitos, el usuario puede optar por crear un chip fabricado como se describe a continuación.

CAPÍTULO 4: DESARROLLO EXPERIMENTAL DE LA HERRAMIENTA FPGA PARA DIAGRAMA DE BLOQUES Y VHDL

En el presente capítulo empezamos a utilizar el software Quartus II de Altera, que permite diseñar hardware mediante captura esquemática y programación en VHDL, así como también implementándolo en una tarjeta entrenadora, tanto el software como el entrenador están disponibles en el laboratorio de electrónica de la FETD. En esta parte los alumnos de las carreras de Ingeniería en Telecomunicaciones como Electrónica en Control y Automatismo que se encuentren cursando la asignatura de Digitales I podrán entender la teoría con esta herramienta de simulación y servirá para futuros proyectos de investigación.

4.1. EXPERIENCIA 1: CAPTURA ESQUEMÁTICA Y COMPILACIÓN

Antes de iniciar con el trabajo práctico los estudiante deberán revisar el *Manual de Usuario de la tarjeta lógica programable DE1* (ver Anexo 1). En la asignatura de Digitales I (ver Anexo 2, programa de estudios) se realizan operaciones con compuertas lógica en el diseño combinacional para emplear multiplexores, demultiplexores, codificadores, decodificadores, flip-flops que son tópicos que los docentes enseñan a los estudiantes de las carreras ya mencionadas con anterioridad.

En la actualidad se diseñan prácticas de pequeños circuitos lógicos que posteriormente se implementan en el protoboard del laboratorio de electrónica conectando entre sí varios circuitos integrados. El objetivo de esta experiencia 1 es que los alumnos van hacer lo mismo pero en vez de montar el circuito, diseñarán un circuito lógico digital mediante la herramienta de programación *Quartus II* y que entiendan la importancia del diseño jerárquico.

La experiencia consiste en diseñar un circuito digital de un número binario (0 a 9) mediante un display de 7 segmentos, es decir que la vamos a dibujar o realizar la captura esquemática y posteriormente dicho circuito se “descargará” en el dispositivo programable DE1 de Altera, donde se verificará su correcto funcionamiento.

Esta herramienta permite hacer cualquier circuito que se desee, no tenemos más utilizar compuertas lógicas en el área de trabajo (como un protoboard) del software *Quartus II* y realizar las conexiones mediante líneas que simulan a los cables. Para mantener un correcto ordenamiento de los archivos por parte de los estudiantes de Digitales I, será necesario crear un directorio de trabajo, donde almacenaremos los archivos de práctica que se trata en la materia. Por ejemplo, se puede crear un directorio C:\DigitalesI\Practicas.

A continuación se detalla el paso a seguir para capturar y compilar un esquema con Quartus II y finalmente procederemos a la simulación del circuito y la respectiva configuración del FPGA (dispositivo de lógica programable) de *Altera*. Vamos a crear un componente (bina7seg) de un número de 4 bits en binario natural, que permita generar el mismo número pero en sistema hexadecimal (ver figura 4.1) y visualizar en un display de 7 segmentos.

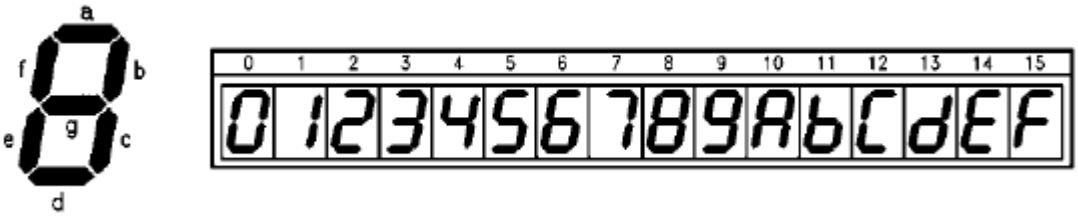


Figura 4. 1: Numeración en sistema hexadecimal para un display de 7 segmentos.
Fuente: El autor

4.1.1. Iniciar *Quartus II* de Altera

Para arrancar con la herramienta de programación *Quartus II* hay que elegir la opción Quartus II 7.2 Web Edition (32-Bit) en el escritorio de las computadoras del laboratorio de electrónica en la FETD, o ir al Botón de Inicio, buscar en todos los Programas > Altera > Quartus II 7.2 (ver figura 4.2).

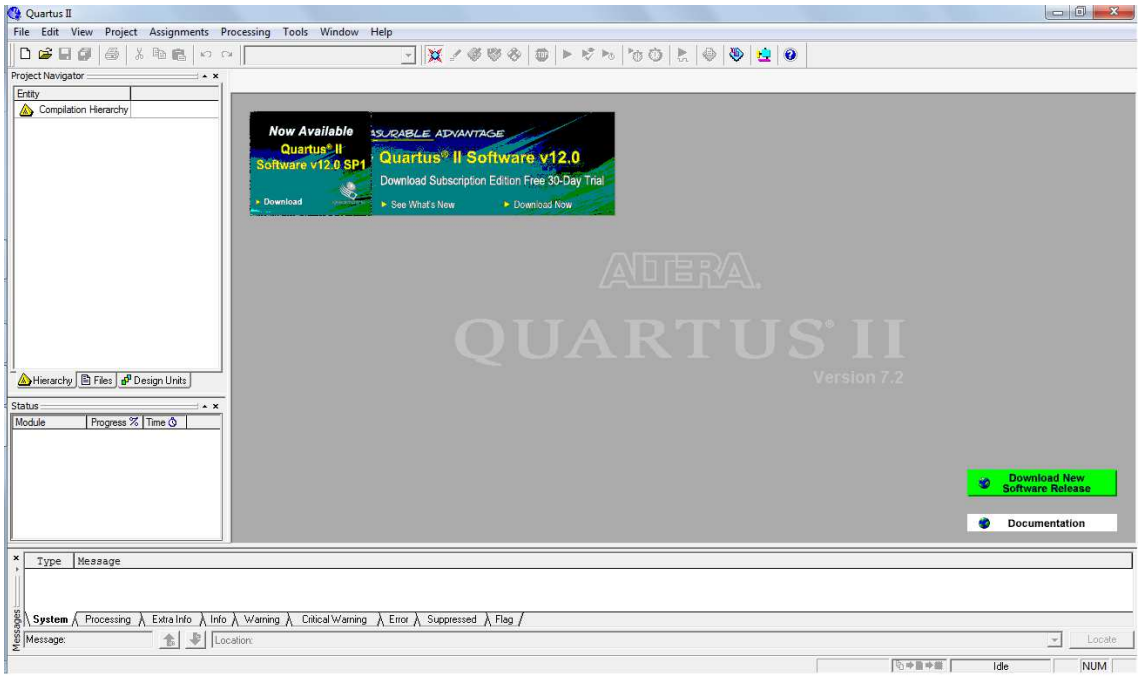


Figura 4. 2: Ventana de inicio de Quartus II 7.2 Web Edition.
Fuente: El autor

4.1.2. Creando nuevo proyecto en *Quartus II* de Altera

Una vez presentado el inicio de *Quartus II* seleccionamos *New Project Wizard* desde el menú *File* como se muestra en la figura 4.3. El mismo nos permite diseñar proyectos de circuitos digitales que para el presente proyecto sólo trabajaremos con la captura esquemática y programación en VHDL.

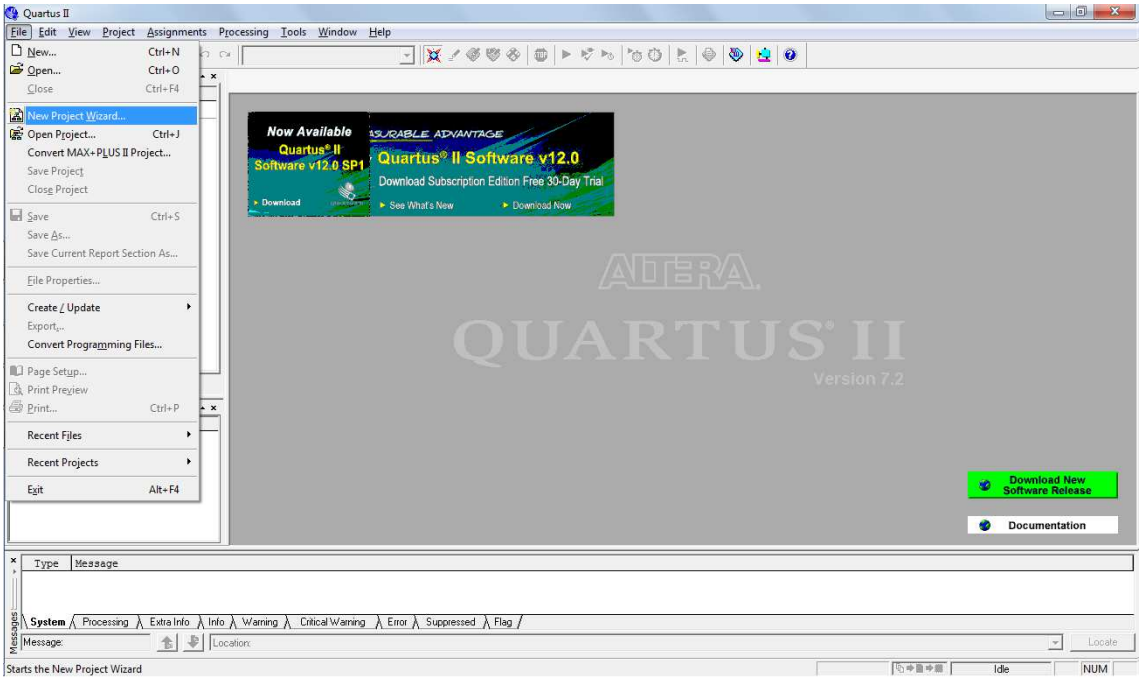


Figura 4. 3: Selección de un nuevo proyecto para diseño de circuitos digitales.

Fuente: El autor

A continuación, después de pulsar *New Project Wizard*, aparecerá la ventana mostrada en la figura 4.4, en la que crearemos un directorio en el que se va a trabajar (C:\Users\pc\Desktop\Electrónica\VHDL-Lab-Digitales\bin7seg); el nombre del proyecto (bin7seg); y el nombre del archivo principal de la jerarquía (bin7seg). Para evitar confusiones se da el mismo nombre en los dos campos.

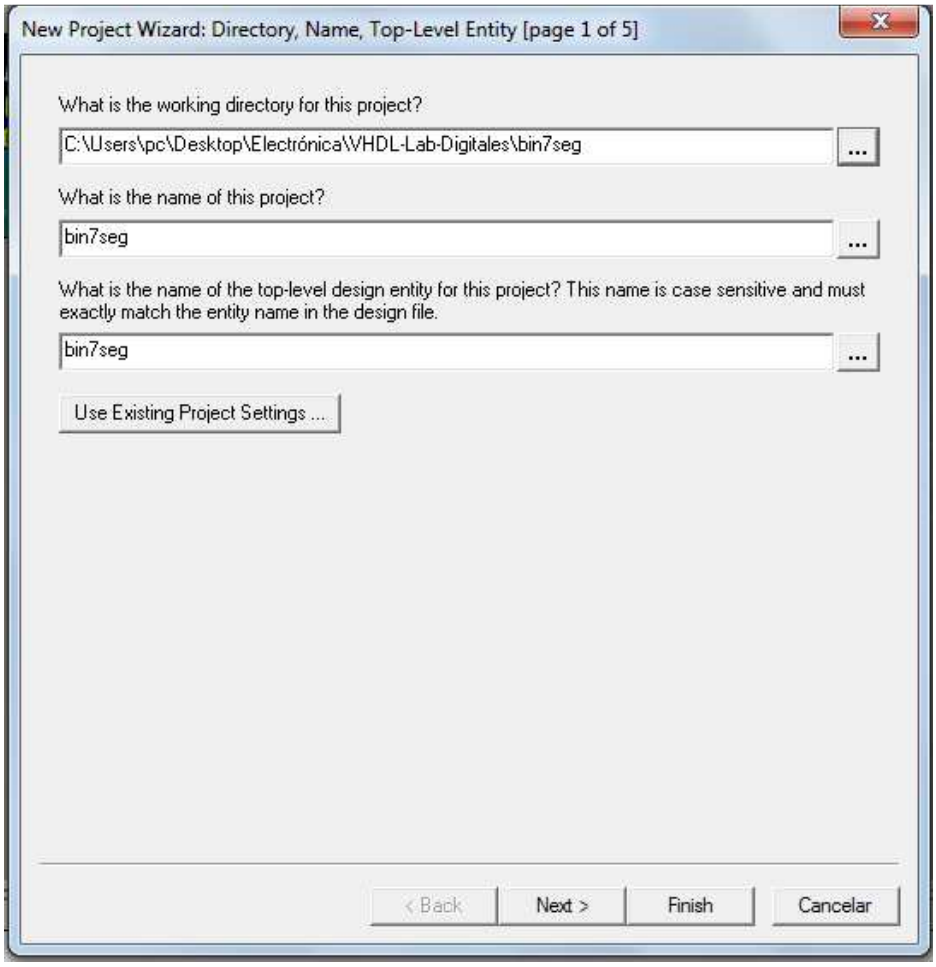


Figura 4. 4: Ventana para crear un nuevo proyecto.
Fuente: El autor

En la figura 4.5 se muestra una ventana para saber si se desea agregar archivos que requiera el proyecto, pero para esta experiencia no hay que hacer nada, simplemente pulsaremos *Next*.

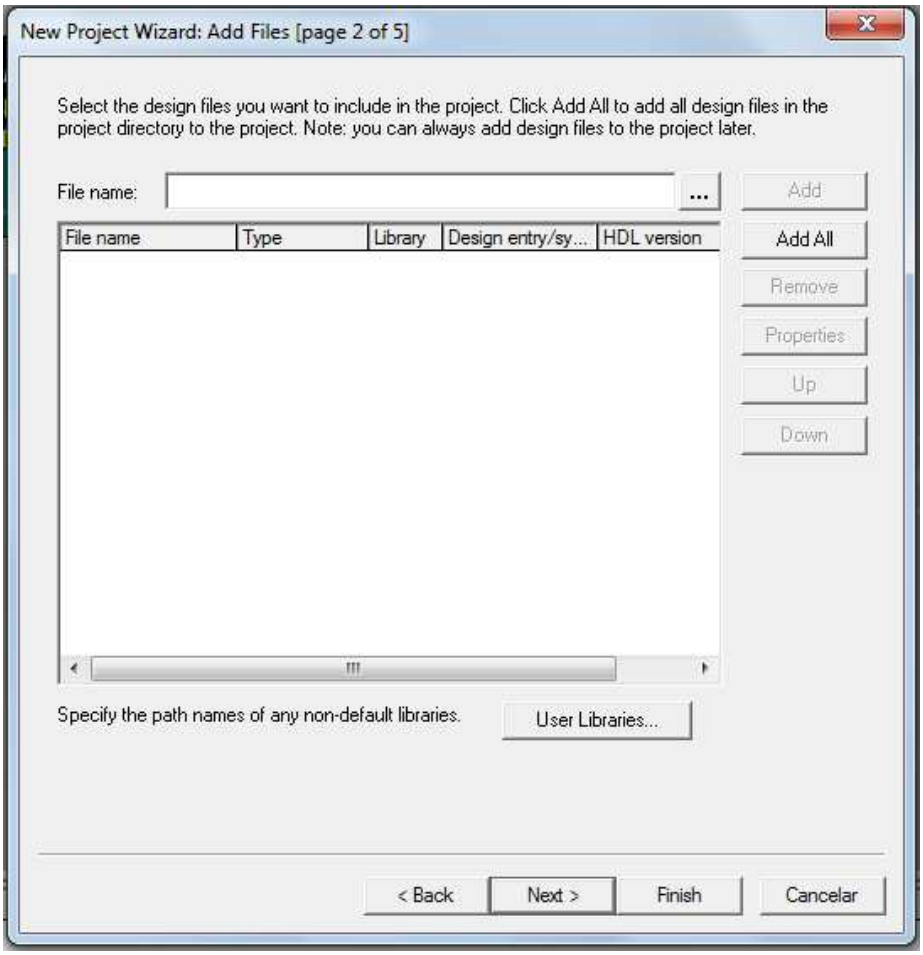


Figura 4. 5: Selección de un nuevo proyecto para diseño de circuitos digitales.
Fuente: El autor

En la figura 4.6 se muestra la ventana que permite seleccionar al dispositivo electrónico (familia y modelo) con el que trabajaremos de aquí en adelante para la asignatura de Digitales I y posteriormente en Digitales II y Laboratorio de Digitales. Por lo tanto la familia del dispositivo es Cyclone II, y después aparecerá una lista de dispositivos disponibles de la parte de abajo, seleccionamos el dispositivo EP2C20F484C7, que se encuentra montado en la tarjeta entrenadora DE1 del laboratorio de electrónica. Finalmente tenemos todo lo necesario para declarar el proyecto, por lo que, para terminar, hay que pulsar el botón *Finish*.

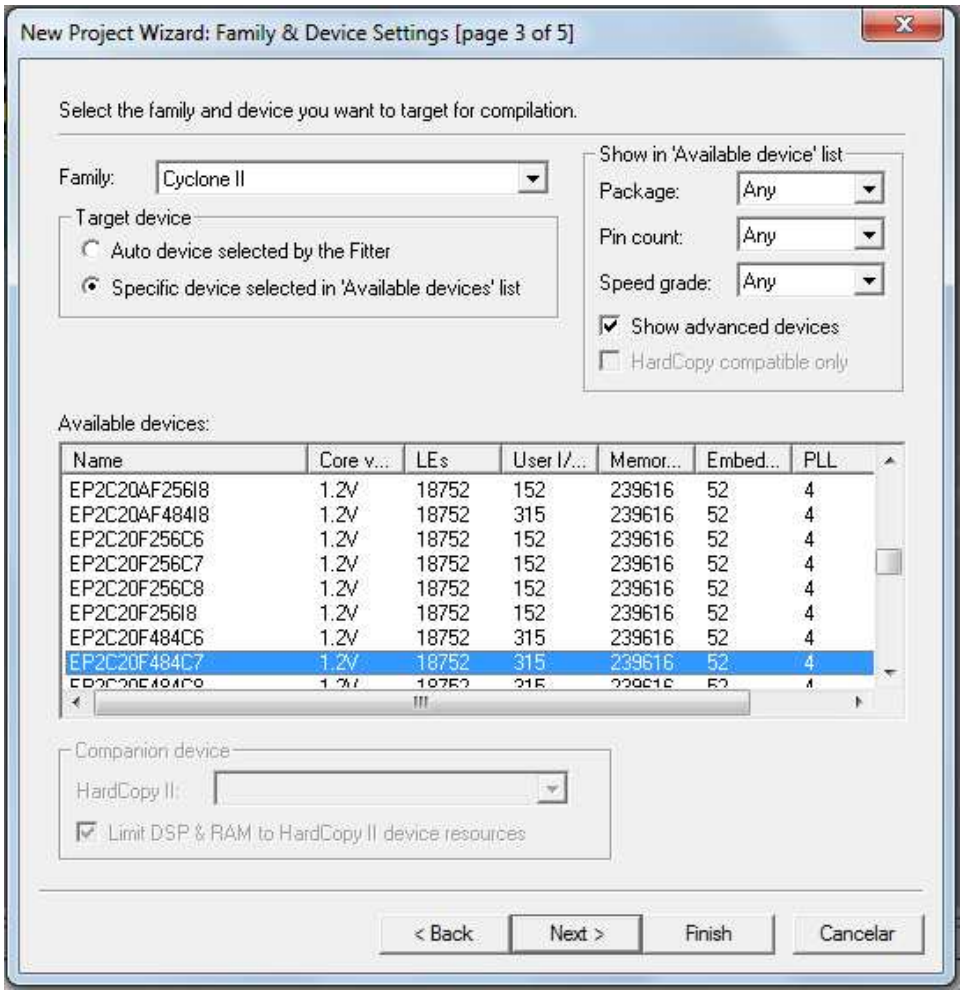


Figura 4. 6: Selección del dispositivo EP2C20F484C7.
Fuente: El autor

4.1.3. Captura esquemática de *Quartus II* de Altera

Una vez cumplido con los pasos de creación de un nuevo proyecto, ahora vamos a diseñar el esquemático que contendrá el circuito del componente, para eso en el menú *File* escogemos la opción *New* en la que aparece la ventana mostrada por la figura 4.7, permitiendo elegir el tipo de archivo que se desea diseñar, seleccionamos *Block diagram / Schematic File*.

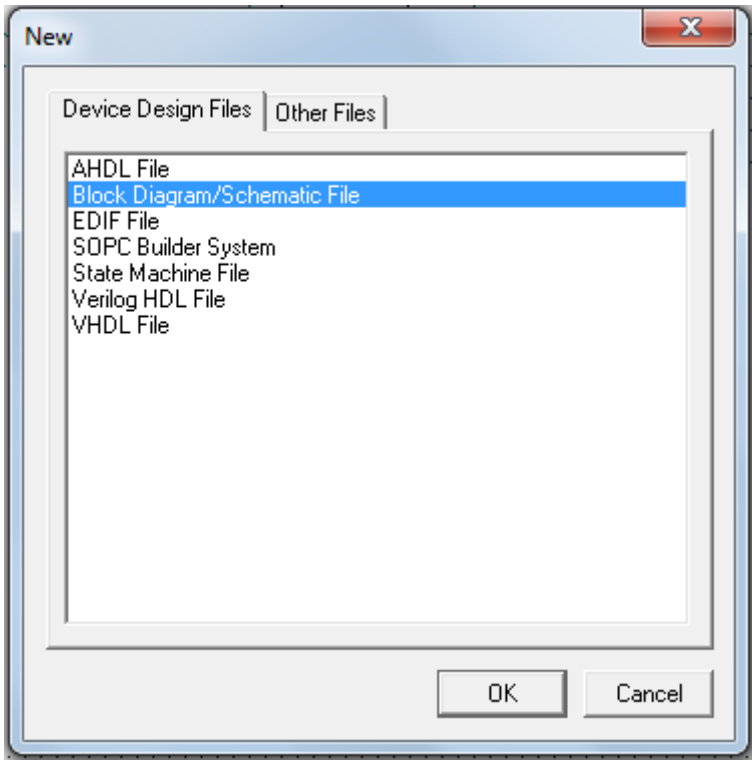


Figura 4. 7: Nueva captura esquemática.
Fuente: El autor

Existen dos grupos diferentes de archivos: *Device Design Files* (Archivos de Diseño) y *Other Files* (otros tipos de archivos complementarios). En el presente proyecto utilizaremos principalmente: captura esquemática *(Block Diagram/Schematic File)*, archivo de símbolo *(Block Symbol File)* para el componente que se ha de crear; archivo de forma de onda *(Vector Waveform File)*, para describir la evolución temporal de las entradas con vistas a la simulación de un circuito previamente creado y *VHDL File* para la programación de alto nivel.

La opción a seleccionar es *Block Diagram/Schematic File* que se mostró en la figura 4.7, donde al crearse el archivo tendrá como extensión .bdf e inmediatamente pulsando OK aparece la ventana (ver figura 4.8) de captura de esquemática. Antes de nada conviene guardar el archivo en el directorio de trabajo con el nombre bina7Seg.bdf, mediante la opción *Save As* del menú *File*. Tenga cuidado con el nombre, pues, por defecto, aparece el nombre del proyecto (bin7seg).

4.1.4. Introducción de componente en la captura esquemática.

El esquema mostrado por la figura 4.8 es el que procederemos a diseñar para posteriormente crear un símbolo que servirá como un decodificador de 7 segmentos. Para introducir un componente en la ventana de captura hay que pulsar el botón, que está en la barra de botones de la izquierda, o hacer doble clic con el ratón en la hoja donde se desea insertar, tras lo cual se presentará una ventana (ver figura 4.9), de esta manera se le puede decir al programa qué componente, de los que dispone en su base de datos, se quiere introducir.

En la figura 4.9 se muestra la ventana con las librerías disponibles, no son más que directorios dentro del ordenador con una serie de símbolos que se pueden usar de aquí en adelante para visualizar números a través de del decodificador de 7 segmentos en los circuitos diseñados. Para seleccionar el componente que se desea introducir, existen dos alternativas: teclearlo directamente de *Name*, o seleccionarlo en la ventana *Libraries*, buscando en la librería adecuada.

Primeramente introduciremos en la hoja de trabajo esquemático el componente *input* (señal de entrada al circuito de la hoja activa). Al teclear su nombre en el campo *Name* y pulsando en OK para la introducción del componente. Como podemos comprobar, dicho componente ha aparecido en la ventana de dibujo y haciendo clic en el lugar donde se quiera insertarlo.

Si el sitio donde se ubica el componente no es de su agrado, se puede arrastrar con el ratón o algo muy útil es copiar y pegar para introducir componentes iguales que previamente han sido insertados en la hoja de trabajo, sin tener que recurrir a la ventana para seleccionar otro tipo de componente. Finalmente, si ya ha ubicado correctamente el puerto de entrada *(input)*, introduciremos el componente o compuerta *not*, que se ha de situar por debajo y a la derecha del componente anterior, tal y como se muestra en la figura 4.10.

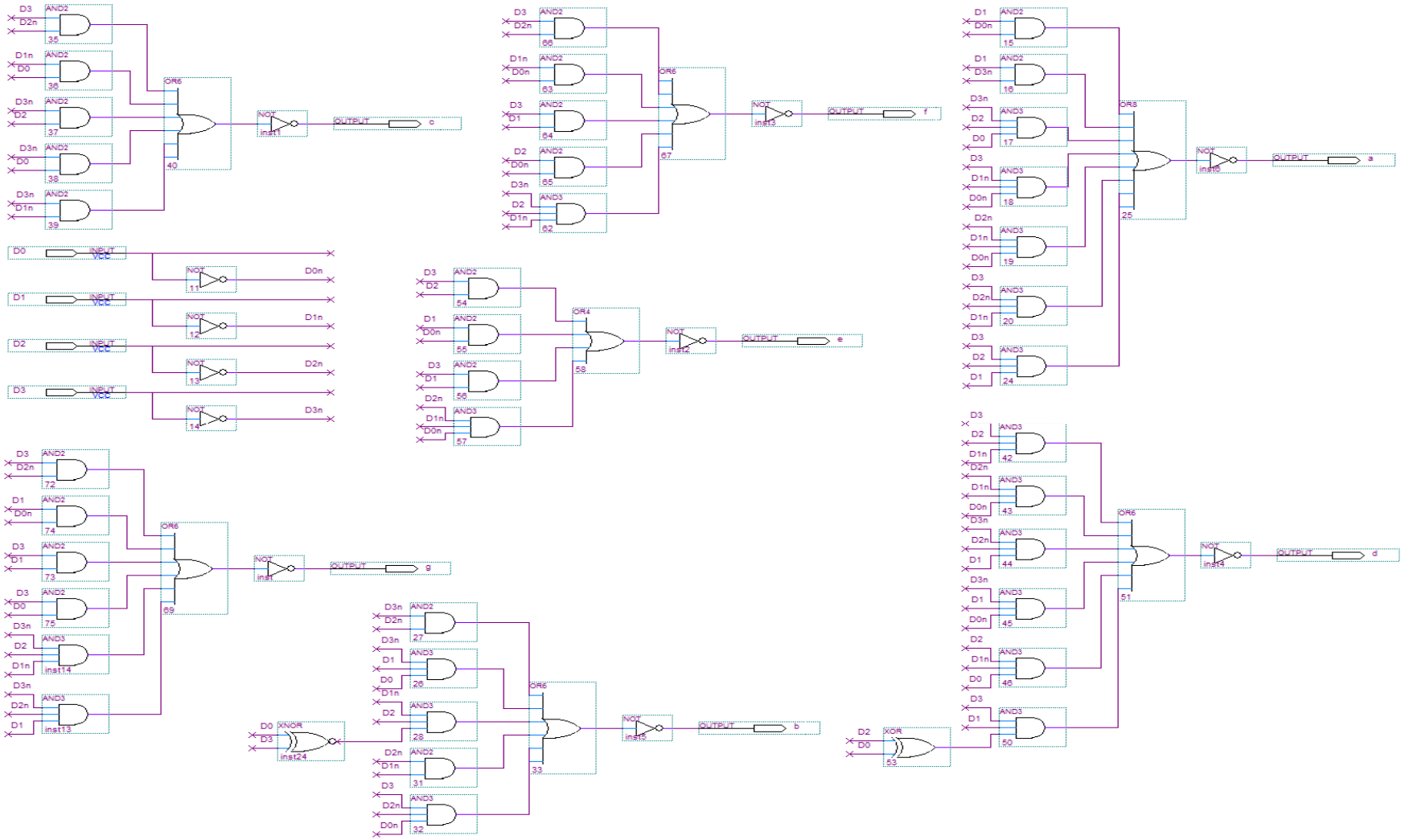


Figura 4. 8: Captura esquemática del binario de 7 segmentos.

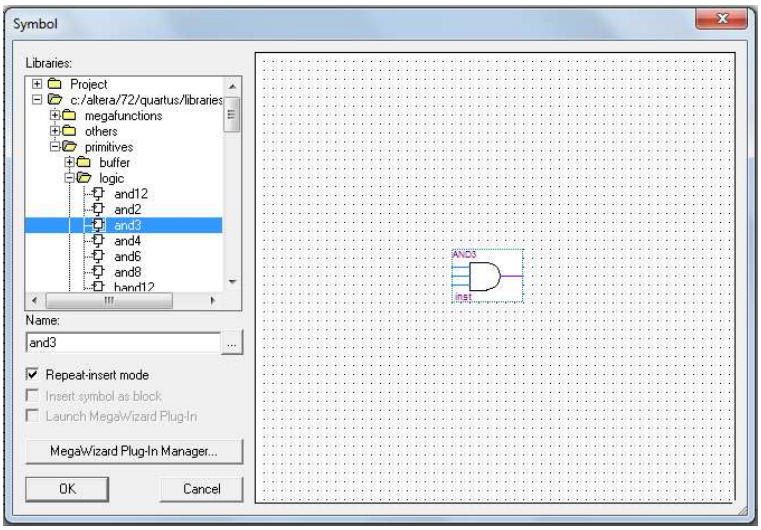


Figura 4. 9: Captura esquemática del binario de 7 segmentos.
Fuente: El autor

Para la unión de los componentes que se han introducido en la hoja, emplearemos el cableado mediante el cursor del ratón a una patilla del componente y, cuando la flecha cambie a una cruz (+), arrastraremos el cable al pin o patilla del componente a unir, obviamente sin soltar el botón del ratón, salvo mejor criterio de que se desee realizar más de una conexión, para ese caso basta con soltar y volver a pulsar el botón del ratón, permitiendo hacer un nodo. En la Figura 4.11 se observa la unión de dos cables que se representa por un punto grueso.

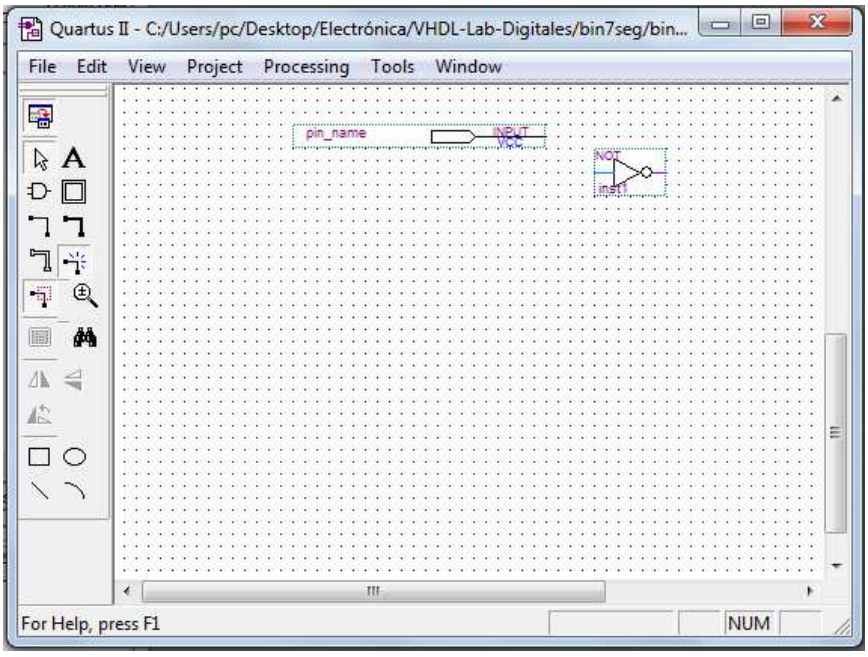


Figura 4. 10: Captura esquemática del binario de 7 segmentos.
Fuente: El autor

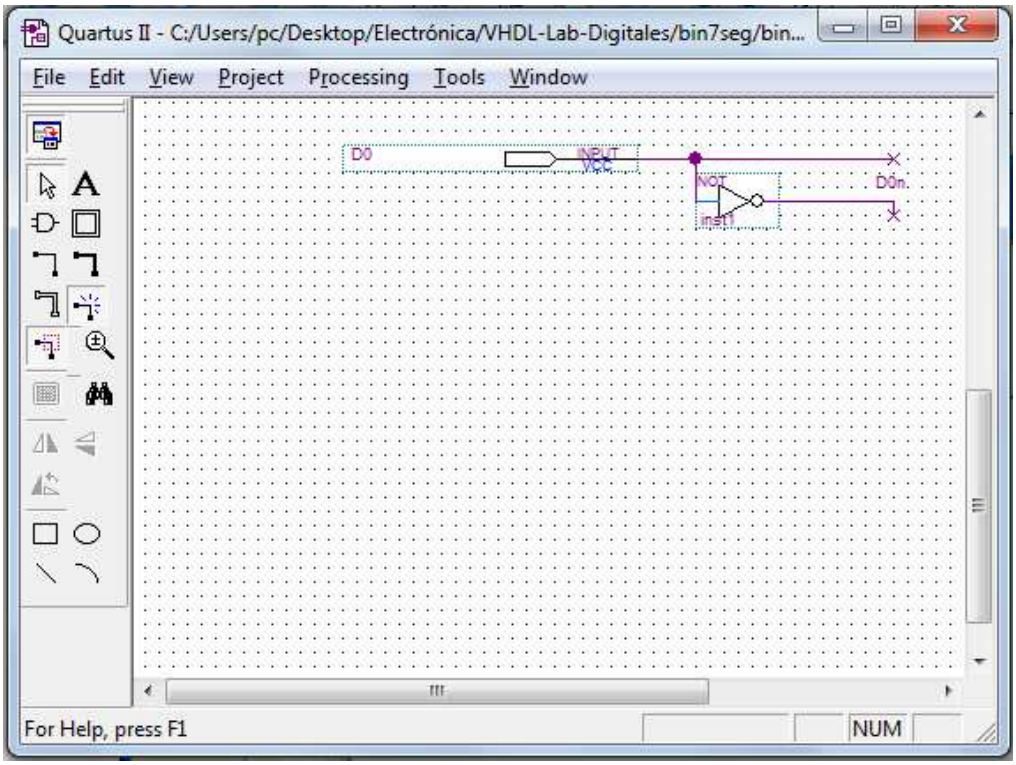


Figura 4. 11: Captura esquemática del binario de 7 segmentos.
Fuente: El autor

Finalmente este procedimiento lo repetiremos de acuerdo al esquema propuesto en la figura 4.8 para crear el símbolo que nos permita decodificar el número en hexadecimal.

4.1.5. Creación del símbolo mediante captura esquemática.

Una vez capturado el esquema mostrado en la figura 4.8, procedemos a la creación del símbolo asociado al esquema que se acaba de crear, de tal forma que se pueda utilizar dicho componente en futuros esquemas, que permitirán visualizar al sistema de numeración hexadecimal mediante un display de 7 segmentos. Para el nuevo símbolo usamos el comando *Create Symbol Files for Current File*dentro de la opción *Create/Update* del menú *File*, dicha acción muestra la ventana (ver figura 4.12) con el nombre del símbolo (bina7Seg.bsf), que resulta tener igual nombre del esquemático, esto creará el símbolo.

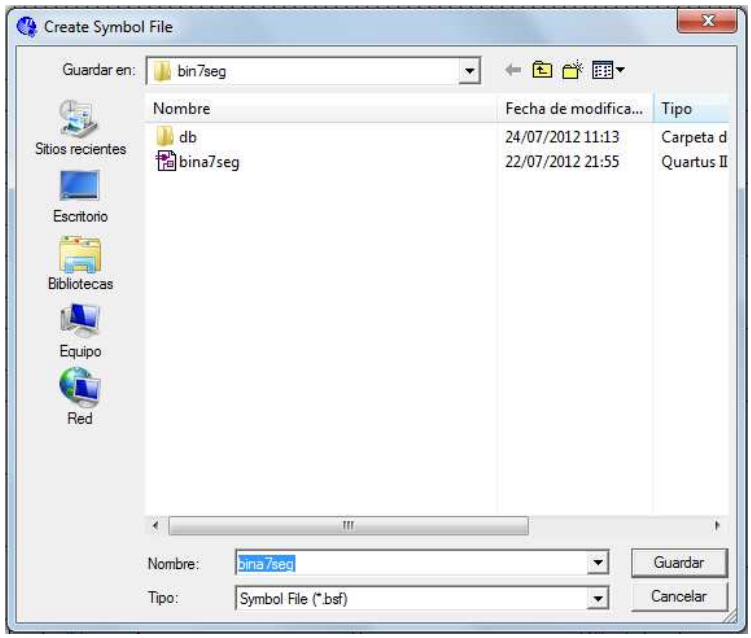


Figura 4. 12: Ventana para guardar el nuevo símbolo binario de 7 segmentos.
Fuente: El autor

Ahora que se ha grabado el nuevo símbolo, podemos inclusive editar el símbolo,abriendo el archivo *bina7Seg.bsf* con laopción *Open* del menú *File*. En la ventana abierta, se debe indicar el tipo dearchivo a buscar, qué es *Graphic Files(*.gdf, *.bdf, *.bsf, *.sym)* yluego indicarle el nombre del archivo con la extensión *.bsf* (véase la figura 4.13).

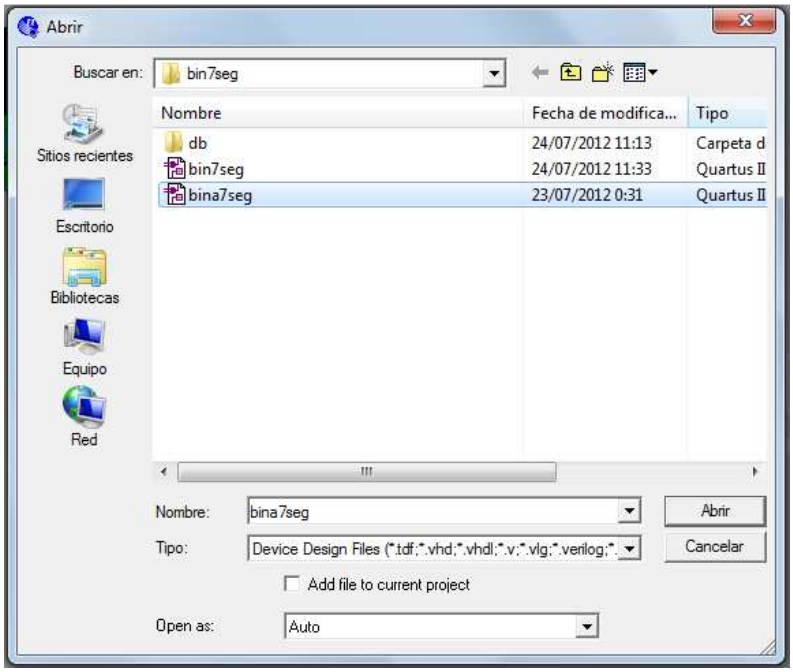


Figura 4. 13: Ventana para abrir el símbolo binario de 7 segmentos.
Fuente: El autor

Una vez que se abra el símbolo puede ocurrir que las entradas y salidas no guardan el respectivo orden, él mismo nos permite cambiar el orden de las entradas y salidas del nuevo símbolo, basta con dar doble clic sobre el nombre tanto de entrada como salida y escribir el nuevonombre de entrad o salida. No es correcto arrastrar los nombres de las señales.El símbolo final se muestra en la figura 4.14, y una vez que se tengan ordenadas las entradas y las salidas, se guardan los cambios.

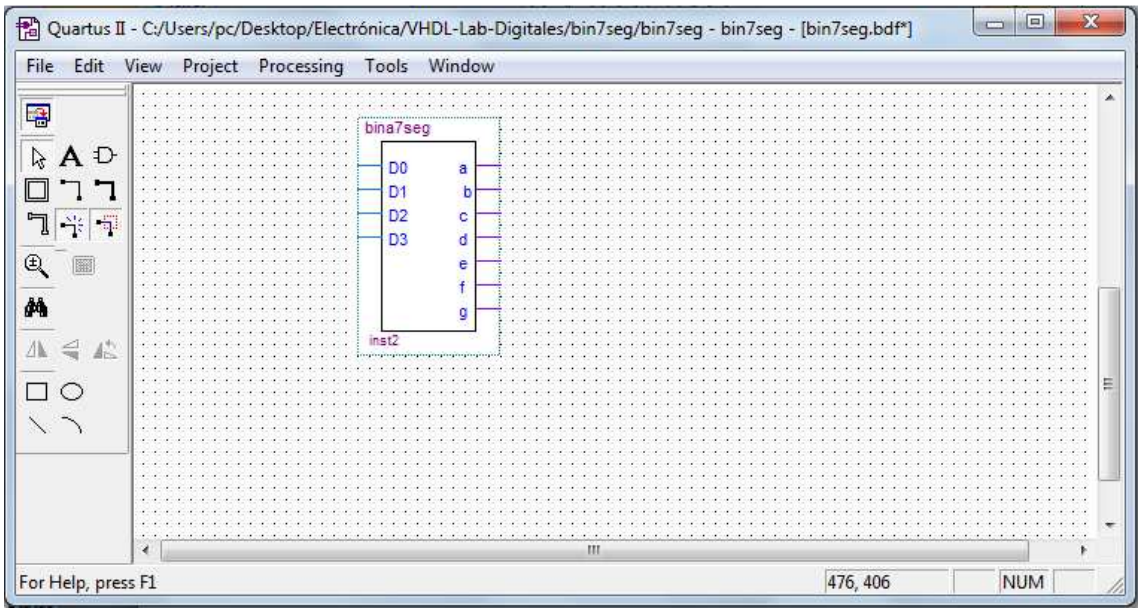


Figura 4. 14: Ventana para abrir el símbolo binario de 7 segmentos.
Fuente: El autor

Una vez finalizado el diseño de bina7segse va a realizar un circuito que muestre mediante un display (HEX0)de los cuatro disponible de la tarjeta (Cyclone II de Altera) entrenadora de lógica programable, el número codificado enbinario (D0 a D3) se los ingresamediante los switch SW3 al SW0 (ver Anexo B, donde encontrarán la configuración de los pines de la placa de LógicaProgramable DE1). El esquema final,tras realizar los pasos que se describen más adelante, debe quedar como el que se muestra enla figura 4.15.

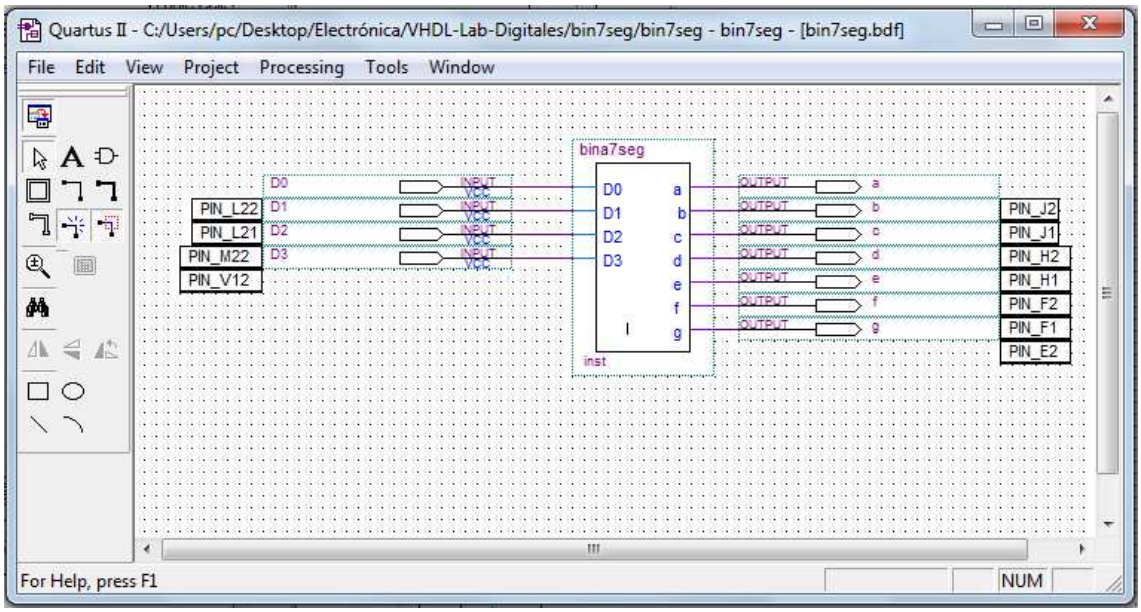


Figura 4. 15: Diseño esquemático del binario de 7 segmentos.
Fuente: El autor

En la figura 4.15 podemos apreciar el esquemático diseñado, el mismo que tiene cuatro puertos de entrada*(input)* yocho puertos de salida*(output)*. Tanto las entradas como salidas se conectan mediante un esquema sencillo, donde interviene el diseño bina7Seg.bdf. El dicho no consistía en tener el símbolo con las conexiones de entradas y salidas debido a que el componente (bina7Seg) está disponible para cualquier diseño del tipo esquemático, solamente insertarlo y conectar correctamente sus terminales.

Si el componente bina7seg no se creaba, tendríamos inconvenientes cada vez que necesite usar undecodificador de binario a siete segmentos, es decir sería una desventaja tener que dibujar todo el circuito de la figura 4.8 para cada display que se desee activar.Adicional podemos darnos cuenta y comprobado, que no es nada simple.

El diseño esquemático o de bloques ha sido organizado mediante diagramas de bloques, denominado diseño jerárquico, puesto que elcircuito forma una jerarquía de bloques partiendo de un nivel superior, en el quese representa el circuito con un alto nivel de abstracción, para ir aumentando el nivel dedetalle conforme se desciende en la jerarquía.

4.1.6. Compilación del circuito inversor de binario a siete segmentos

Una vez que se ha terminado con la tarea de capturar todos los esquemas de la jerarquía, procedemos a la compilación del proyecto, y por tanto el mismo verifica todos los archivos de los que consta el mismo,para verificar que el circuito está libre de errores.

4.1.7. Compilación de la experiencia práctica.

Previamente a la compilación, debemos guardar el proyecto con todos los esquemas diseñados siempre que exista algún cambiode última hora, y posteriormente ejecutar *Compiler Tool* del menú *Processing* como se muestra en la figura 4.16.

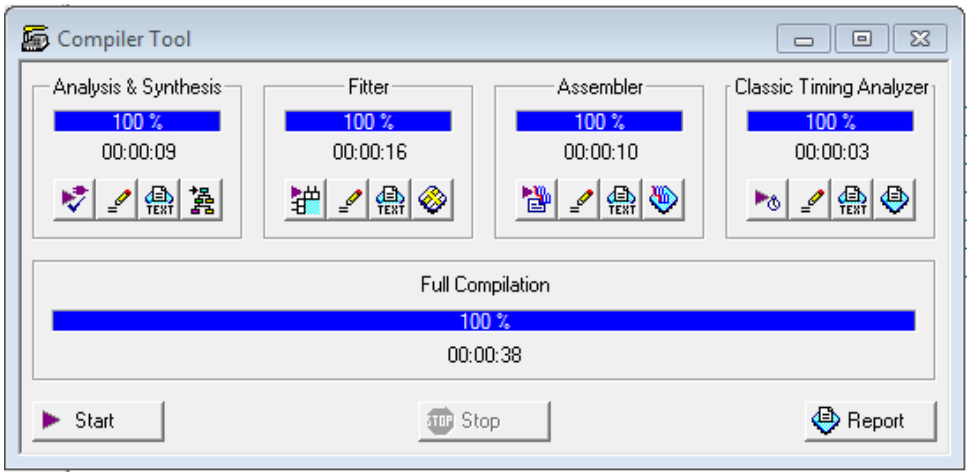


Figura 4. 16: Compilación del diseño del binario de 7 segmentos.
Fuente: El autor

Al terminar el proceso de compilación el programa nos informará el número de avisos (warnings), que por lo general si todo el diseño es correcto sóloaparecerán avisos que se muestran en la figura 4.17, así como también los mensajes deinformación del propio programa. En el caso de que problemas en la compilación debemos verificar el listado de inconvenientes para dar solución al inconveniente, muchas de las veces los alumnos no identifican el problema, por lo tanto deben de recurrir al docente para que los asesore en posibles soluciones.

Figura 4. 17: Resultado de compilación exitoso.
Fuente: El autor

4.1.8. Asignación de los pines del FPGA DE1 de Altera a las terminales de I/O.

Si la compilación realizada no se muestran errores en el diseño, a pesar de tener advertencias de no tener conectadas las entradas y salidas, para lo cual procedemos a la asignación de los pines del dispositivo FPGA DE1 hasta las terminales ya mencionadas del esquema propuesto. Para la asignación de pines debemos ir al menú *Assignments* y escoger la opción *Pin Planner*. En la figura 4.18 se muestra la designación de pines (ver tabla 4.1) tanto para la señales de entrada (D0, D1, D2 y D3) y las salidas (a, b, c, d, e, f, g) que se conectarán al display (Hex0) de la placa entrenadora DE1 de Altera. Como guía debemos seguir la figura 4.15 que muestra dicha asignación.

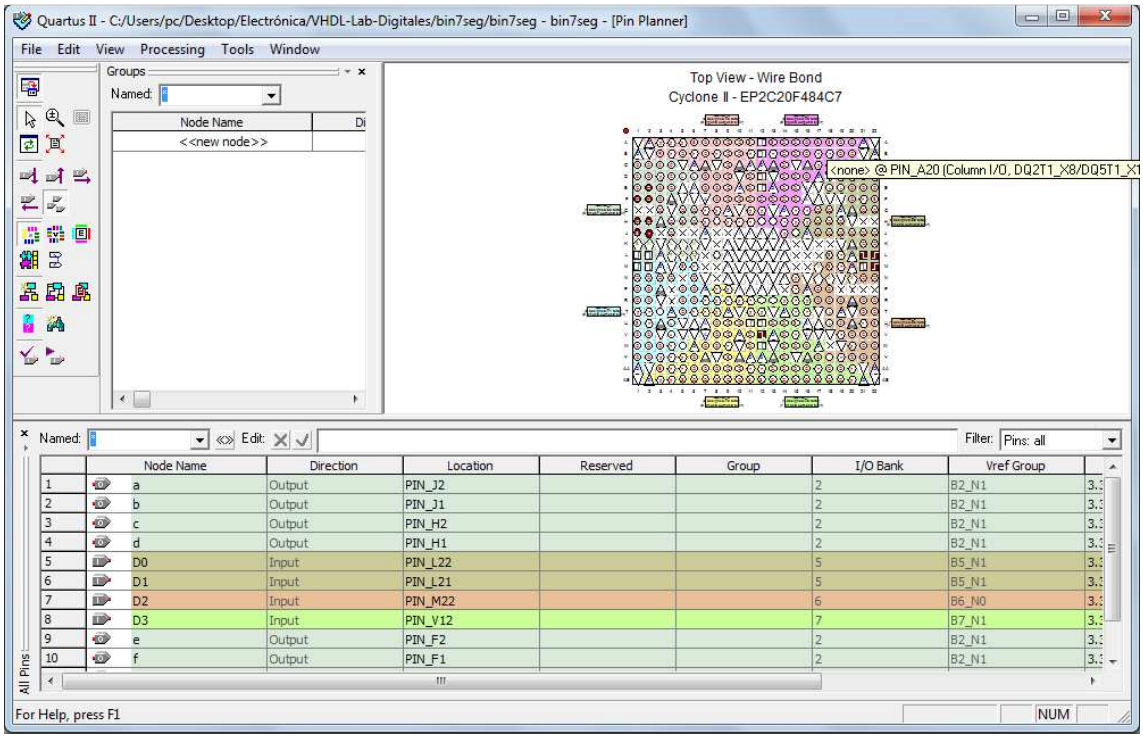


Figura 4. 18: Resultado de compilación exitoso.
Fuente: El autor

Tabla 4. 1: Asignación de pines para el diseño propuesto.

Señal	Tipo	Pines FPGA	Componente
D0	Entrada	PIN_L22	SW0
D1	Entrada	PIN_L21	SW1
D2	Entrada	PIN_M22	SW2
D3	Entrada	PIN_V12	SW3
a	Salida	PIN_J2	HEX0 [0]
b	Salida	PIN_J1	HEX0 [1]
c	Salida	PIN_H2	HEX0 [2]
d	Salida	PIN_H1	HEX0 [3]
E	Salida	PIN_F2	HEX0 [4]
F	Salida	PIN_F1	HEX0 [5]
G	Salida	PIN_E2	HEX0 [6]

Fuente: DE1 de Altera (ver Anexo B)

4.1.9. Resultado obtenido en la tarjeta DE1 de Altera.

Finalmente después de diseñar, compilar y asignar los pines, procedemos a la comprobación experimental en la tarjeta entrenado DE1 de Altera disponible en el laboratorio de electrónica de la FETD. Para esto la tarjeta debe estar conectado al PC mediante el cable USB, inmediatamente la tarjeta es reconocida por su controlador USB, en el menú *Tools* opción *Programmer* aparecerá la ventana que se muestra en la figura 4.19.

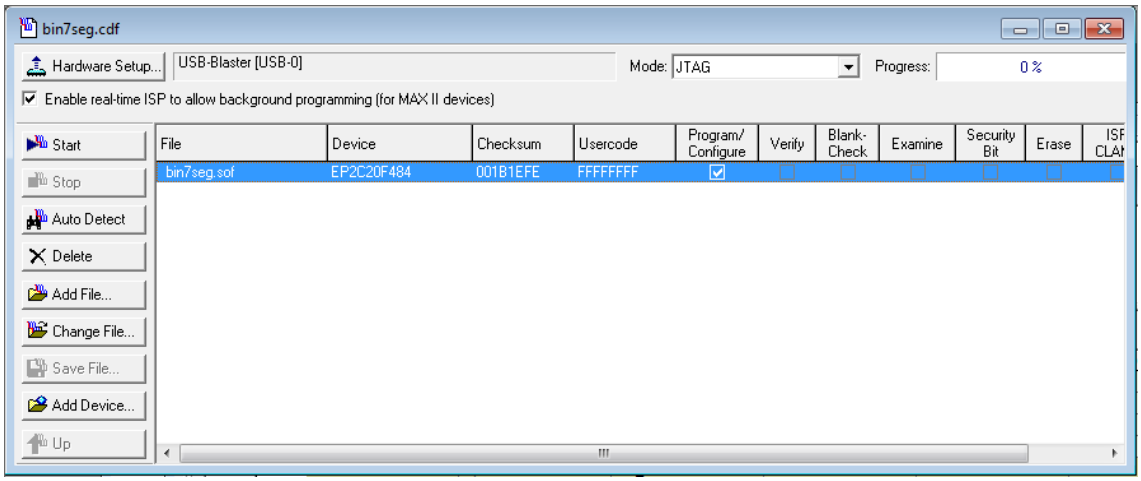


Figura 4. 19: Ventana para la programación en la tarjeta DE1.
Fuente: El autor

Al proceder con la programación hacia la tarjeta DE1 de Altera, el resultado esperado es el que se muestra en las figuras 4.20 y 4.21.

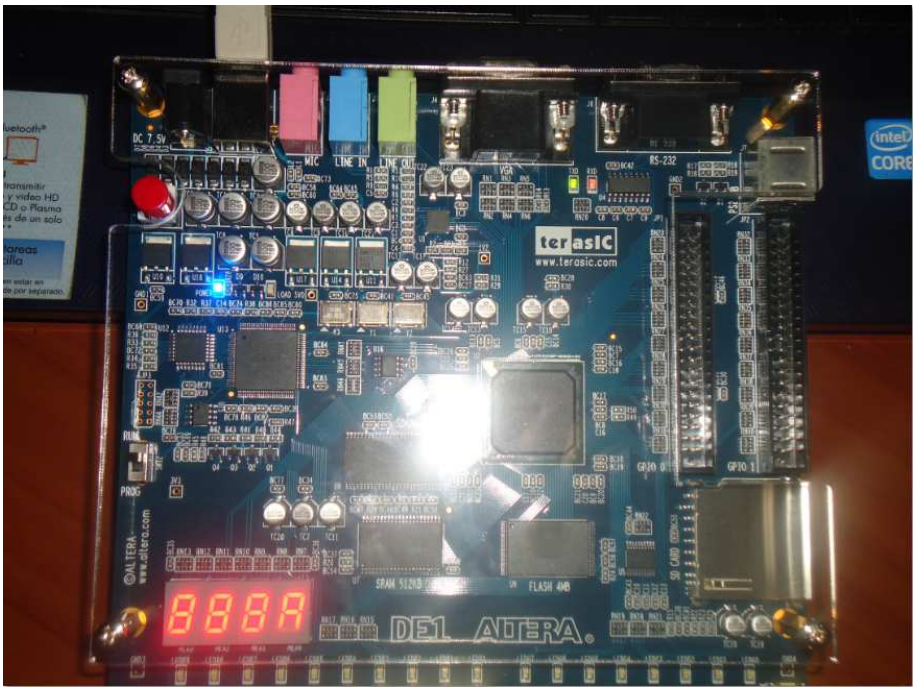


Figura 4. 20: Resultado obtenido en la tarjeta DE1 que muestra el 10h (A).
Fuente: El autor

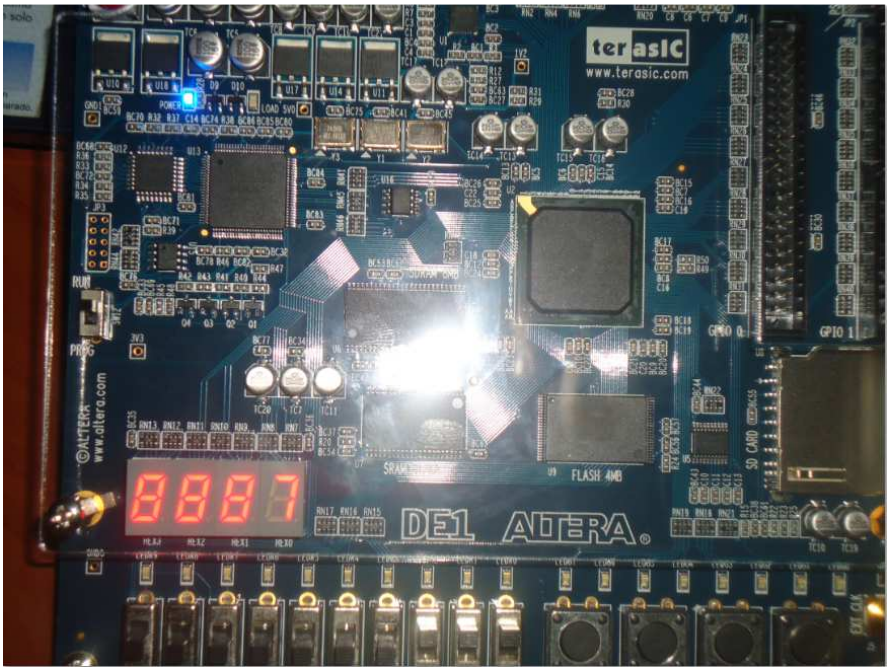


Figura 4. 21: Resultado obtenido en la tarjeta DE1 que muestra el 7h (7)
Fuente: El autor

4.2. EXPERIENCIA 2: PROGRAMACIÓN VHDL, COMPILACIÓN Y FUNCIONAMIENTO EN LA DE1.

Para la presente experiencia desarrollaremos la programación VHDL de operaciones con compuertas lógicas NOT, AND, OR, NAND, NOR, XOR y XNOR. Empezamos con nuestro primer diseño utilizando los componentes ya mencionados, son circuitos sumamente sencillos que se aborda en el programa de estudios de Digitales I, para así poder simular en VHDL. Es necesario para todo programa en VHDL emplear la librería ieee y el paquete std_logic_1164.

4.2.1. Programación VHDL a nivel de compuertas lógicas.

De acuerdo al circuito lógico a nivel de compuerta que se muestra en la figura 4.22, procederemos a realizar la programación en VHDL el mismo que no requiere de muchas líneas. Como podemos apreciar de la figura 4.22 tenemos tres entradas (a, b, c) y 3 salidas (x, y, z).

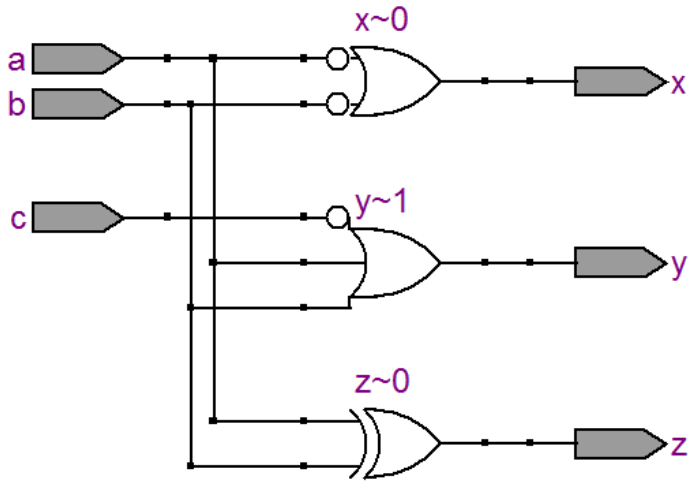


Figura 4. 22: Circuito topológico básico a nivel de compuertas lógicas.
Fuente: El autor

Del circuito topológico mostrado, aplicamosel algebra booleana o por simple deducción obtendremos las salidas x, y, z. Estas expresiones booleanas serán de gran ayuda para programar en VHDL, las salidas se expresan de la siguiente manera:

$$x = \bar{a} + \bar{b}$$

$$y = a + b + \bar{c}$$

$$z = a \oplus b$$

De manera similar a lo aprendido para el diseño de un nuevo proyecto, el procedimiento sigue siendo el mismo, la diferencia es que cuando está creado el proyecto, debemos crear un nuevo archivo desde el menú *File* en la opción *New*, en la figura 4.23 se muestra la opción a escoger, que sería *VHDL File.*

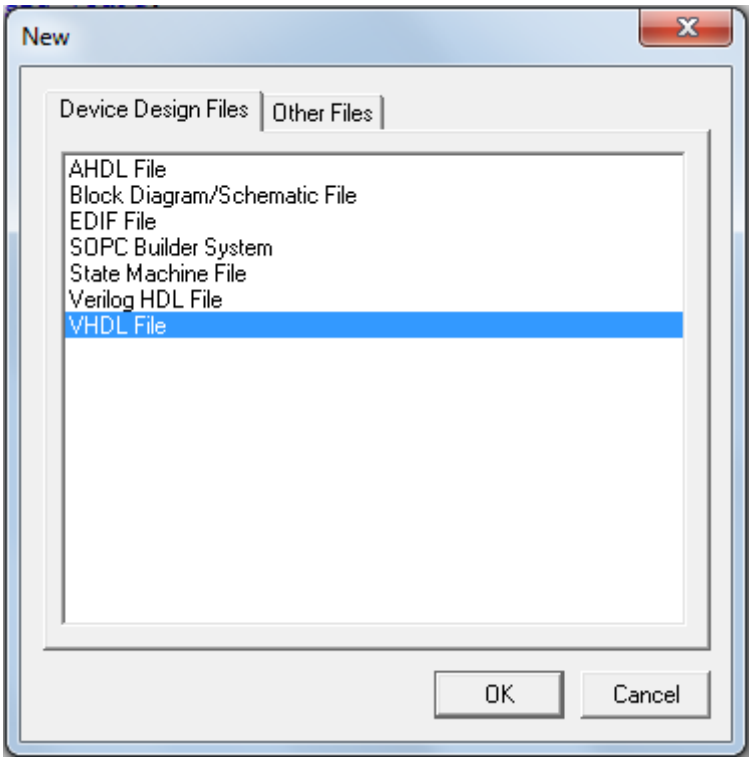


Figura 4. 23: Ventana para escoger diseño en formato VHDL.
Fuente: El autor

Antes de iniciar con la programación hay que tener en cuenta como se esta constituida la ventana de programación en VHDL denominada declaración de entidad (ver figura 4.24) la misma que consta de ***entity,*** **identificador de la entidad (sumador), port (declarar los puertos I/O), modo de operación (I/O), tipo de dato y *end* (fin de la declaración)**

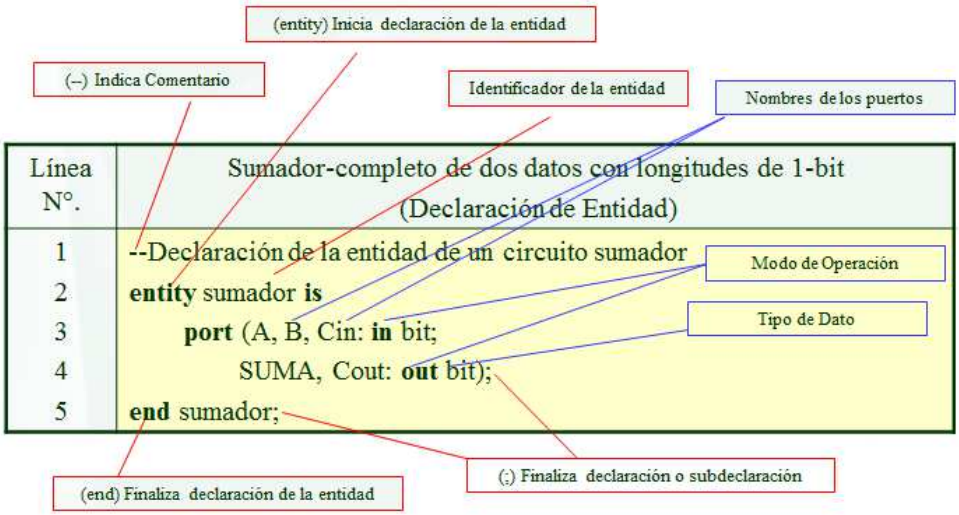


Figura 4. 24: Declaración de la entidad para programar en VHDL.
Fuente: El autor

Como ya se explicó acerca de como crear el *VHDL File*, finalmente programaremos en VHDL del circuito propuesto en la figura 4.22. El archivo inicia con un comentario (color verde) de la Tesis VHDL, también se llama la librería ieee para su respectivo uso, se ha declarado la entidad con el mismo nombre del proyecto (no olvidarse de esta parte), se configuran los puertos de entrada y salida. En la figura 4.25 se muestra la programación VHDL.

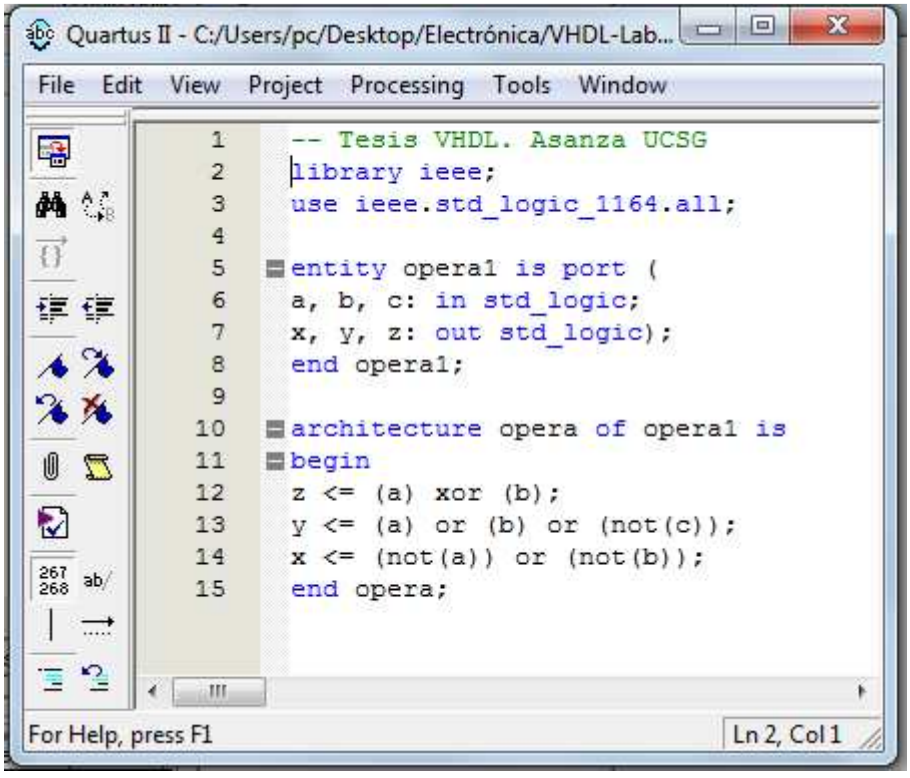


Figura 4. 25: Programa en VHDL para compuertas lógicas.

Fuente: El autor

No debemos olvidar de la importancia de compilar el archivo que se programó en VHDL, en la anterior práctica se aplicó el proceso que se mantiene para cualquier tipo de archivo en la herramienta de programación Quartus II.Las funciones booleanas de salida (x, y, z) del circuito topológico que fueron deducidas a partir de la figura 4.22, y que esas 3 ecuaciones de salida se muestra en la figura 4.25. Antes de mostrar el resultado en la tarjeta de entrenamiento DE1, debemos de configurar o asignar los pines de la DE1 hacia las entradas y salidas que se detallan en la figura 4.26.

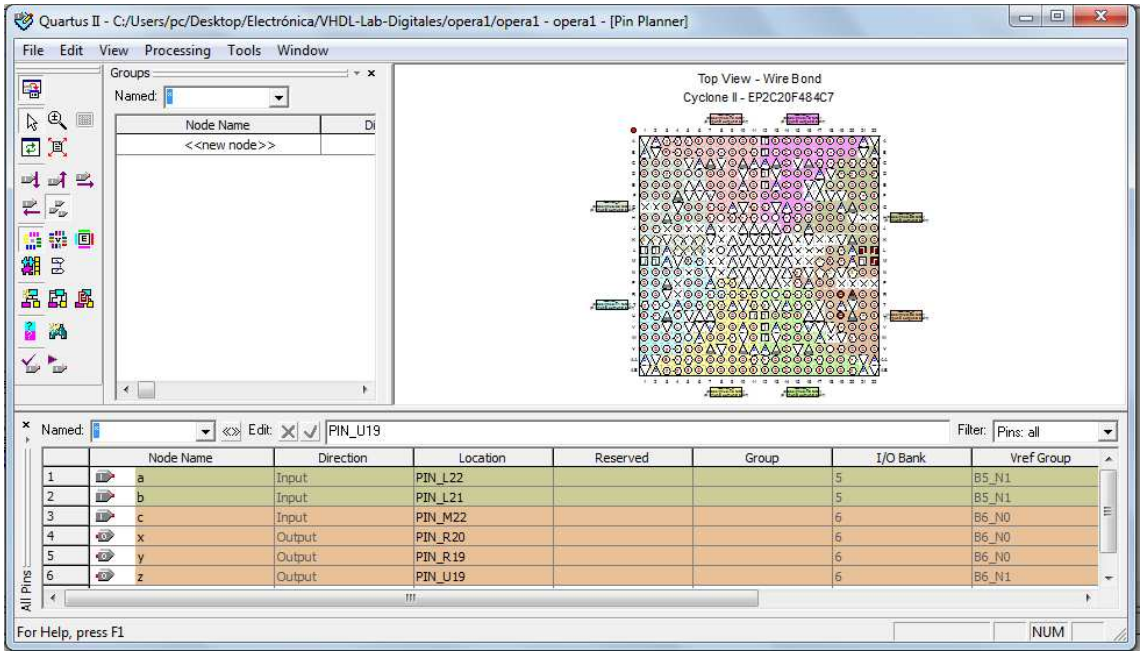


Figura 4. 26: Asignación de pines l de compuertas lógicas.

Fuente: El autor

La figura 4.27 muestra la tarjeta DE1 de Altera ya funcionando de acuerdo a la programación en VHDL, como se puede apreciar el diodo LEDR0 (salida x) se encuentra encendido de acuerdo a la tabla 4.2 de verdad de las funciones booleanas de salida (x, y, z). La salida ***"y"*** corresponde al diodo LEDR1, y ***"z"*** es el diodo LEDR2. Mientras que las señales de entrada a, b y c corresponden a SW0, SW1 y SW2 respectivamente para cada entrada.

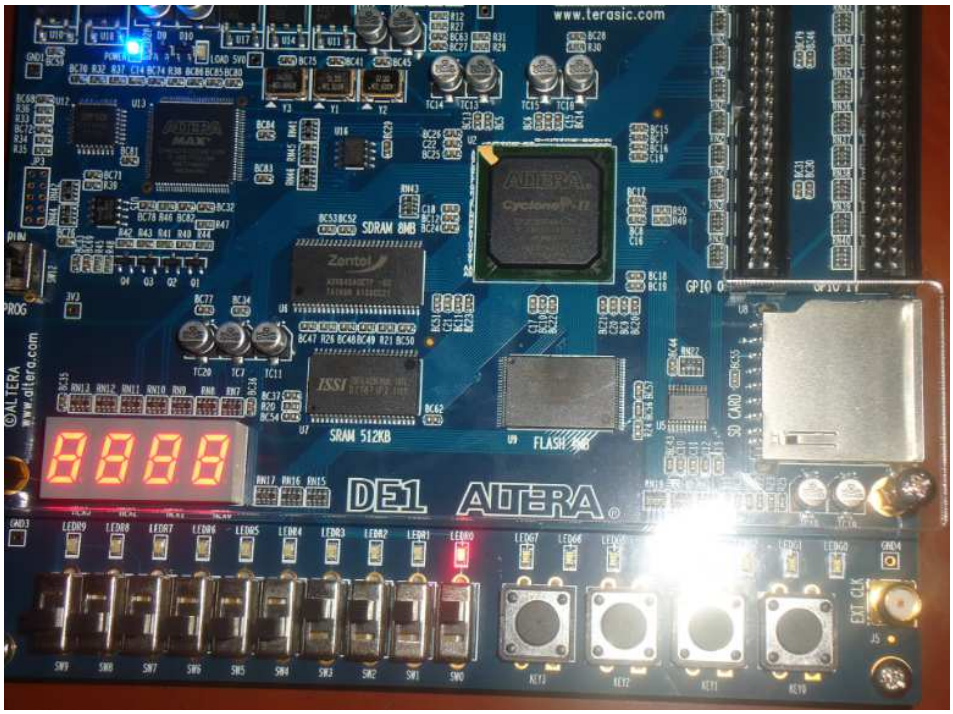


Figura 4. 27: Resultado obtenido del operador lógico..
Fuente: El autor

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1 CONCLUSIONES

- El estudio del diseño VHDL se vuelve importante en el estudiante de telecomunicaciones para contribuir a su aprendizaje significativo en la Facultad de Educación Técnica para el Desarrollo en la Universidad Católica de Santiago de Guayaquil.

- El diseño VHDL se ha convertido en una herramienta de mucho valor en la práctica mediante su estudio hace referencia en grandes empresas donde se encargan en simulación y comportamiento del circuito electrónico.

- En el estudio de este tema tutorial el estudiante de la carrera de telecomunicaciones estará capacitado en desarrollar circuitos electrónicos e implementar para sus casos en la vida profesional y estar preparado para sus competencias a futuro

5.2 RECOMENDACIONES

- El diseño VHDL presenta varias librerías que pueden dar uso a muchos dispositivos electrónicos por la cual en la implementación debemos hacer referencia que verificamos en el sistema Quartus II CAD de altera el comportamiento del circuito y su buen desarrollo

- Si bien hemos manifestado el estudio digital se encuentra en pleno proceso de desarrollo por la cual el catedrático y estudiante de la ingeniería de telecomunicaciones deben de estar en una constante investigación y actualización de los sistemas, memorias, dispositivos en la actualidad.

- Debemos indicar que el personal de desarrollo que esta a cargo de los laboratorios tenga arduos conocimiento en el tema digital en la cual se puede dar uso a un buen estudio de elementos electrónicos que se encuentra al alcance del estudiante de digitales 1 de la carrera de telecomunicaciones

REFERENCIAS BIBLIOGRÁFICAS

1. Blogspot. (s.f.). *taller de circuitos logicoas.* Recuperado el domingo de junio de 2012, de http://feagle.blogspot.com/2009/09/taller-3-pensamiento-logico.html

2. Blogspot-circuitoslogicos.geovanni. (2010). *Circuitos Logicos.* Recuperado el domingo de junio de 2012, de blogspot.com: http://geovanni-circuitoslogicos.blogspot.com/

3. *Circuitos logicos* . (s.f.). Obtenido de http://geovanni-circuitoslogicos.blogspot.com/

4. Dopico, A. G. (2009). *Distribucion de carga y aumento del grado de parlelismo en simulacion sincrona de lenguajes de descripcion hardware.* departamento de arquitectura y tecnologia de sistemas informaticos.

5. Ecolosfera, B. d. (s.f.). *ecolosfera*. Recuperado el miercoles de junio de 2012, de http://ecolosfera.com

6. Espinosa, R. D. (Febrero de 2009). *DISEÑO DIGITAL PARA INGENIERIA.* República de Colombia: UNITECNICA (Ingecómputo) Manizales.

7. Govoid. (s.f.). *el mundo de la Fpga*. Recuperado el miercoles de junio de 2012, de http://www.govoid.es/2011/04/programando-el-hardware-i-el-mundo-de-las-fpga/

8. M., M. E. (semestre 2011-A). *Logica Binaria y Algebra de Boole.* guayaquil : universidad catolica de santiago de guayaquil .

9. Muñoz Frías, J. D. (2012). *Introducción a los Sistemas Digitales: Un enfoque usando lenguajes de descripción de hardware.* California: Creative Commons.

10. Roth, C. h. (2009). *Fundamentos de diseños logicos .* Mexico, DF: Thonmson Learning.

11. Tocci, Ronald.dj. Y Widmer,NEal.S. (2009). *Sistema Digitales octava edicion.* Mexico: Pearson Educacion.

12. Victor P. Nelson ; H. Troy Nagle ; Bill D. CArroll ; J.David Irwin. (Mexico). *Analisis y Diseño de Circuito Logicos Digitales .* DOn Fowley.

13. Wakerly, j. F. (2007). *diseño digital principio y practicas .* mexico: pearson educacion.

14. G.Boole,*An InvestigationoftheLawsofThought*,1854,reprintedbyDoverPublications, NewYork, 1954.

15. C.E.Shannon, "ASymbolicAnalysisofRelayandSwitchingCircuits,"*Transactions ofAIEE*57 (1938), pp.713–723.

16. E.V.Huntington,"SetsofIndependentPostulatesfortheAlgebraof Logic"

17. TransactionsoftheAmericanMathematicalSociety5 (1904), pp.288–309.

18. S.BrownandZ.Vranesic, Fundamentals of Digital Logic with Verilog Design, 2nd ed.(McGraw-Hill:NewYork,2007).

19. Z.Navabi, *VHDL – AnalysisandModelingofDigitalSystems*, 2nd ed. (McGraw-Hill: NewYork, 1998).

20. D.L.Perry, VHDL, 3rded.(McGraw-Hill:NewYork,1998).

21. J.Bhasker, AVHDLPrimer, 3rded.(Prentice-Hall:EnglewoodCliffs,NJ,1998).

22. “International Technology Roadmap for Semiconductors,” http://www.itrs.net

23. Altera Corporation, “Stratix III Field Programmable GateArrays,”http://www.altera.com

24. Xilinx Corporation, “Virtex-5 Field Programmable Gate Arrays,”

25. S. Sedra and K. C. Smith, Microelectronic Circuits, 5th ed. (Oxford UniversityPress: New York, 2003).

26. J. M. Rabaey, Digital Integrated Circuits, (Prentice-Hall: Englewood Cliffs, NJ,1996).

27. Texas Instruments, Logic Products Selection Guide and Databook CD-ROM, 1997.

28. National Semiconductor, VHC/VHCT Advanced CMOS Logic Databook, 1993.

29. Motorola, CMOS Logic Databook, 1996.

30. Toshiba America Electronic Components, TC74VHC/VHCT Series CMOS Logic Databook, 1994.

31. Integrated DevicesTechnology, High Performance Logic Databook, 1994.

32. J. F. Wakerly, Digital Design Principles and Practices 3rd ed. (Prentice-Hall: Englewood Cliffs, NJ, 1999).

33. M. M. Mano, Digital Design 3rd ed. (Prentice-Hall: Upper Saddle River, NJ, 2002).

34. R. H. Katz, Contemporary Logic Design (Benjamin/Cummings:

Redwood City, CA,1994).

35. J. P. Hayes, Introduction to Logic Design (Addison-Wesley: Reading, MA, 1993).

36. D. D. Gajski, Principles of Digital Design (Prentice-Hall: Upper Saddle River, NJ,1997).

BIBLIOGRAFÍA COMPLEMENTARIA

http://www.visionlibros.com/detalles.asp?id_Productos=10086

http://atc2.aut.uah.es/~marcos_s/LabArqCom/ManualUsuarioVHDL.pdf

http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/rubio_s_d/capitulo3.pdf

http://redeya.bytemaniacos.com/electronica/tutoriales/PDF/vhdl.pdf