UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

Evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales

AUTOR:
Reyes Erazo, Cristhian Eduardo

Trabajo de Titulación previo a la obtención del grado de
INGENIERO EN TELECOMUNICACIONES

TUTOR:
Suarez Murillo, Efraín

Guayaquil, Ecuador
12 de Septiembre del 2016

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por el Sr. **Reyes Erazo, Cristhian Eduardo** como requerimiento para la obtención del título de **INGENIERO EN TELECOMUNICACIONES**.

TUTOR

_____

Suarez Murillo, Efraín

DIRECTOR DE CARRERA

_____

Heras Sánchez, Miguel Armando

Guayaquil, a los 12 del mes de Septiembre del año 2016

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Reyes Erazo, Cristhian Eduardo**

DECLARÓ QUE:

El trabajo de titulación "**Evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales**" previo a la obtención del Título de **Ingeniero en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 12 del mes de Septiembre del año 2016

EL AUTOR

_____

REYES ERAZO, CRISTHIAN EDUARDO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Reyes Erazo, Cristhian Eduardo**

Autorizó a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del Trabajo de Titulación: **"Evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

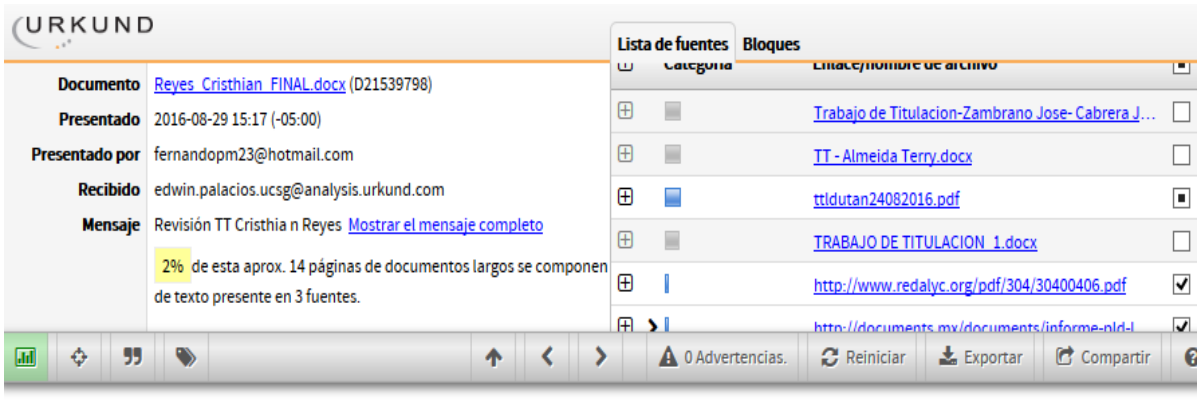
Guayaquil, a los 12 del mes de Septiembre del año 2016

EL AUTOR

_____

REYES ERAZO, CRISTHIAN EDUARDO

REPORTE DE URKUND



UNIVERSIDAD CATÓLICA DE SANTIAGO DE GUAYAQUIL FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO CARRERA DE INGENIERÍA EN TELECOMUNICACIONES TEMA:

Evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales AUTOR: Reyes Erazo, Cristhian Eduardo

Trabajo de Titulación previo a la obtención del grado de INGENIERO EN TELECOMUNICACIONES TUTOR: Suarez

Murillo, Efraín Guayaquil, Ecuador 12 de Septiembre del 2016

UNIVERSIDAD CATÓLICA DE SANTIAGO DE GUAYAQUIL FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN Certificamos que el presente trabajo fue realizado en su totalidad por el Sr. Reyes Erazo, Cristhian Eduardo como requerimiento para la obtención del

DEDICATORIA

Dedico de manera especial a mi madre María Erazo quien forjó en mí las bases de responsabilidad y deseos de superación, y a la vez una profesional con virtudes infinitas y de gran corazón que me llevan a admirarla cada día más.

A mi amada esposa Lissette Vásquez por su sacrificio y esfuerzo, por su motivación y apoyo constante, pero más que nada por su amor.

A mi hija Melina Reyes que siempre es fuente de inspiración en cada paso que doy.

A mis hermanos que siempre estuvieron pendientes y apoyándome con su gran cariño y calidez familiar

EL AUTOR

REYES ERAZO, CRISTHIAN EDUARDO

AGRADECIMIENTO

Agradezco a mi esposa Lissette Vásquez por tomarme de la mano cuando más lo necesite, por su aporte incansable y su amor incondicional es que hoy vivo este hermoso momento junto a ella, primero fue ella quien logró ser una profesional y ahora me tocó a mí que gracias a su apoyo constante es un sueño logrado para ambos.

Agradezco muy especialmente a mi querida madre María Erazo quien desde el inicio de esta etapa hizo que valga la pena todo el sacrificio vivido, por ser la principal promotora de mis sueños, gracias madre por cada día confiar y creer en mí y mis expectativas, gracias por cada consejo y cada una de las palabras que me guiaron durante mi vida, sin ti esto no hubiera sido posible.

Gracias de corazón a mi tutor Ing. Efraín Suárez por su paciencia, dedicación, motivación, criterio y aliento. Han hecho fácil lo difícil. Ha sido un privilegio contar con su guía y ayuda.

EL AUTOR

REYES ERAZO, CRISTHIAN EDUARDO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f. _____

EFRAÍN OSWALDO SUAREZ MURILLO

TUTOR

f. _____

MIGUEL ARMANDO HERAS SÁNCHEZ

DIRECTOR DE CARRERA

f. _____

NESTOR ARMANDO ZAMORA CEDEÑO

COORDINADOR DE AREA

Índice General

Índice de Figuras

Capítulo 2

Capítulo 3

Índice de Tablas

Capítulo 2:

Capítulo 3:

Resumen

El presente trabajo de titulación consiste en realizar la evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales. La idea del trabajo es fundamental y formativa, porque existe otra herramienta de simulación que no se ha considerado en el programa de estudios de la asignatura Sistemas Digitales I y II. Por lo general, en estas asignaturas los estudiantes utilizan Isis Proteus y Multisim, ambas plataformas son amigables. En la búsqueda de información se pudo constatar que Simulink de MatLab, también permite desarrollar simulaciones de sistemas digitales. Los estudiantes de V Ciclo de Telecomunicaciones, Electrónica en Control y Automatismo y Eléctrico-mecánico pueden hacer uso del presente trabajo como guía y así profundizar más en el tema usando Simulink. Quartus II, es una plataforma que permite realizar programación en VHDL, diseño esquemático y diseño por máquinas de estados y esto a su vez se implementa en la FPGA de Altera disponible en el laboratorio de electrónica. Cabe mencionar que tanto Simulink como Quartus II, realizan múltiples aplicaciones para aplicaciones en telecomunicaciones.

CAPÍTULO 1: DESCRIPCIÓN GENERAL DEL TRABAJO DE TITULACIÓN

En este capítulo se desarrolla parte esencial del trabajo de titulación, en donde lo más relevante, es la definición del problema, objetivos tanto general como específicos y el tipo de metodología utilizada.

1.1. Introducción.

En la última década, la industria de semiconductores ha experimentado una evolución difícil en la complejidad de los diseños de circuitos integrados digitales (IC): el aumento de la densidad de integración y tamaño de la oblea ha hecho posible el diseño de los chips con cientos de millones de transistores.

A lo largo de la carrera de Ingeniería en Telecomunicaciones, específicamente en la asignatura de Sistemas Digitales, el único libro que se utilizaba fue de Tocci. Aunque existían otros libros, casi ningún texto explica la simulación de circuitos digitales usando Simulink y tampoco Quartus II. Por ejemplo, el autor Perelroyzen (2006) trata de manera explícita la simulación de circuitos digitales usando Simulink.

Lo mismo ocurre, con Quartus II, hay libros que enseñan como programar en VHDL, pero no como utilizar el software Quartus II. Este programa, es muy versátil y amigable, lo que permite diseñar circuitos ya sea un esquemático, VHDL o diagramas de estados. Aunque, VHDL tiene una

historia interesante, pero al conocer esta historia probablemente no va a ayudar a escribir mejor un código VHDL.

Simulink y Quartus II son herramientas de simulación, que permiten desarrollar diversas aplicaciones en cada una de las áreas de la Ingeniería en Telecomunicaciones.

1.2. Antecedentes.

Durante la búsqueda de información en el repositorio de ingeniería en telecomunicaciones y electrónica, se pudieron encontrar algunos trabajos de pregrado y posgrado. Por ejemplo, Benalcázar P. & Andrade C. (2013) implementaron circuitos digitales usando el dispositivo FPGA Cyclone II y a su vez simulaciones en Quartus II, este proyecto permitió disponer de tarjetas FPGA, y que han sido de gran ayuda para las asignaturas Laboratorio de Digitales y Sistemas Digitales II.

Otro trabajo encontrado, fue de Sosa C. & Valdez J. (2015) en donde desarrollaron aplicaciones prácticas para Laboratorio de Digitales usando la tarjeta FPGA DE1 de Altera, este grupo también deje implementado los dispositivos para su uso en el Laboratorio. Mientras que Alvarado B. (2016) utilizo el dispositivo FPGA Cyclone IV incorporado en la tarjeta DE0 nano de Altera, este trabajo se desarrollaron algoritmos en VHDL que permitieron comprobar la robustez de la tarjeta FPGA.

1.3. Definición del Problema.

Necesidad de evaluar las plataformas de simulación Simulink y Quartus II que permita incrementar el aprendizaje y conocimientos de los estudiantes de la Carrera de Ingeniería en Telecomunicaciones de la Universidad Católica de Santiago de Guayaquil.

1.4. Justificación del Problema.

Evidenciar que no solamente Isis Proteus o Multisim son utilizados para simular circuitos digitales, y que conozcan de una excelente herramienta de simulación como Simulink de MatLab. Simulink también es compatible con programación VHDL y se puede cargar el programa en el compilador Quartus II de Altera.

1.5. Objetivos del Problema de Investigación.

1.5.1. Objetivo General.

Evaluar las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales

1.5.2. Objetivos Específicos.

- Fundamentar los sistemas combinatoriales y los dispositivos lógicos programables.
- Establecer las plataformas de simulación Simulink y de programación Quartus II.
- Diseñar sistemas combinatoriales usando Simulink.

- Diseñar aplicaciones de sistemas digitales usando la plataforma Quartus II.

1.6. Idea a defender.

Demostrar que Simulink y Quartus II utilizan modelos basados en FPGAs de Altera, esto permitirá que los estudiantes de Ingeniería en Telecomunicaciones cuenten con una nueva herramienta de simulación, y que puede generar proyectos formativos para futuros profesionales.

1.7. Metodología de Investigación.

Para el desarrollo del proyecto de titulación utilizamos los tipos de investigación descriptiva, comparativa y explicativa. El enfoque utilizado es cuantitativo. El trabajo trata de una investigación de carácter no experimental de diseño transeccional exploratorio. El método utilizado es de simulación virtual instrumental en Simulink y Quartus II para aplicaciones en los dispositivos electrónicos

CAPÍTULO 2: FUNDAMENTACIÓN TEÓRICA

2.1. Descripción general.

Para el presente capítulo, se abordarán temas relacionados a los circuitos o sistemas digitales, y que son tratados en el syllabus de la asignatura de V y VI Ciclo de la Carrera de Ingeniería en Telecomunicaciones. No se tratarán temas fundamentales, sino más bien se describirán el, diseño de circuitos lógicos combinacionales y secuenciales, dispositivos FPGA, programación VHDL y plataformas de simulación Quartus II y Simulink de MatLab.

La mayoría de información teórica fueron obtenidas de textos y trabajos de tesis, titulación o proyectos fin de grado.

2.2. Diseño de circuitos lógicos combinacionales.

En esta sección describiremos los circuitos sumadores, multiplexores (Mux), demultiplexores (Demux), codificadores y decodificadores. Un aspecto importante del diseño digital con circuitos MSI, es el diseño e implementación de circuitos aritméticos. Originalmente, los circuitos aritméticos básicos fueron diseñados utilizando componentes discretos, pero este método durante mucho tiempo ha sido sustituido por la introducción de circuitos MSI. Los sumadores de múltiples bits, unidades lógicas y aritméticas y otros circuitos son ahora fácilmente disponibles como los circuitos integrados de mediana escala.

2.2.1. Semisumador y sumador binario.

Una de las tareas más importantes realizadas por un ordenador digital es la operación de suma de dos números binarios. Pero para el diseñador de la lógica, la cuestión importante es cómo diseñar un sumador para aumentar la velocidad, independientemente del tipo de puerta utilizado. Puede ser que una mayor velocidad puede lograrse a costa de una mayor complejidad del circuito.

Supongamos que queremos construir un dispositivo que podría sumar dos bits binarios juntos. Dicho dispositivo se conoce como un semi-sumador *(Half-Adder, HA)*, y su circuito a nivel de compuertas lógicas se muestra en la figura 2.1.

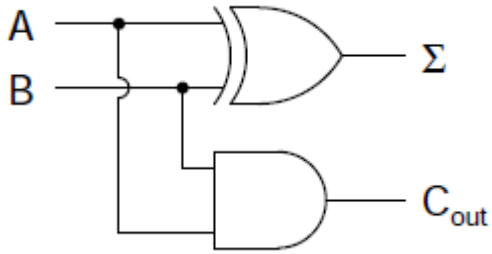


Figura 2. 1: Esquemático de un circuito semi-sumador.
Fuente: (Floyd, 2015)

El símbolo Σ representa la suma de “salida" del semi-sumador, es decir, el bit menos significativo *(Least Significant Bit, LSB)* de la suma. Para el caso de acarreo, Cout representa a la salida "llevar" del semi-sumador, siendo el bit más significativo *(Most Significant Bit, MSB)*. En forma general, un circuito semi-sumador se comporta como un circuito aritmético básico y

que es muy utilizado en cualquier aplicación relacionada a las Telecomunicaciones.

De la figura 2.1 podemos expresar la función de la suma (XOR) y acarreo (AND) de la siguiente manera:

$$\Sigma = A \oplus B$$

$$C_{out} = A \cdot B$$

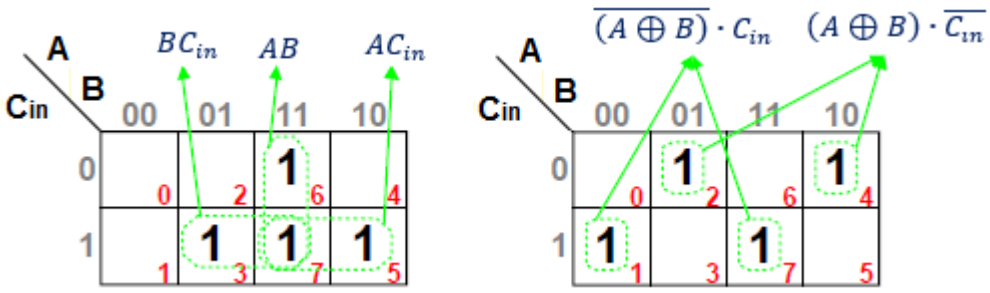
Un enfoque alternativo para la adición de dos números de n bits, es utilizar un circuito separado para cada par de bits. Es decir, que el circuito aceptaría los 2 bits que se añaden, junto con el acarreo C_{in} resultante de la adición de los bits menos significativos. Sería producir como salida a la suma de 1 bit y acarreo de 1 bit como el bit más significativo. En la tabla 2.1 se muestra los valores de verdad de un sumador completo.

Tabla 2. 1: Tabla de verdad de un sumador completo.

A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Elaborado por: Autor

De la tabla 2.1 se pueden obtener las expresiones de suma de productos (SOP) aunque tendríamos un circuito extenso, entonces, a partir de la salida de suma (S) y acarreo (Cout) se realizará la minimización por mapas de Karnaugh.



En la figura 2.2 se muestra el diseño esquemático de un sumador completo a nivel de compuertas lógicas, de acuerdo a la siguiente función lógica:

$$C_{out} = AB + AC_{in} + BC_{in} = AB + (A + B)C_{in}$$

$$S = \overline{(A \oplus B)} \cdot C_{in} + (A \oplus B) \cdot C_{in}$$

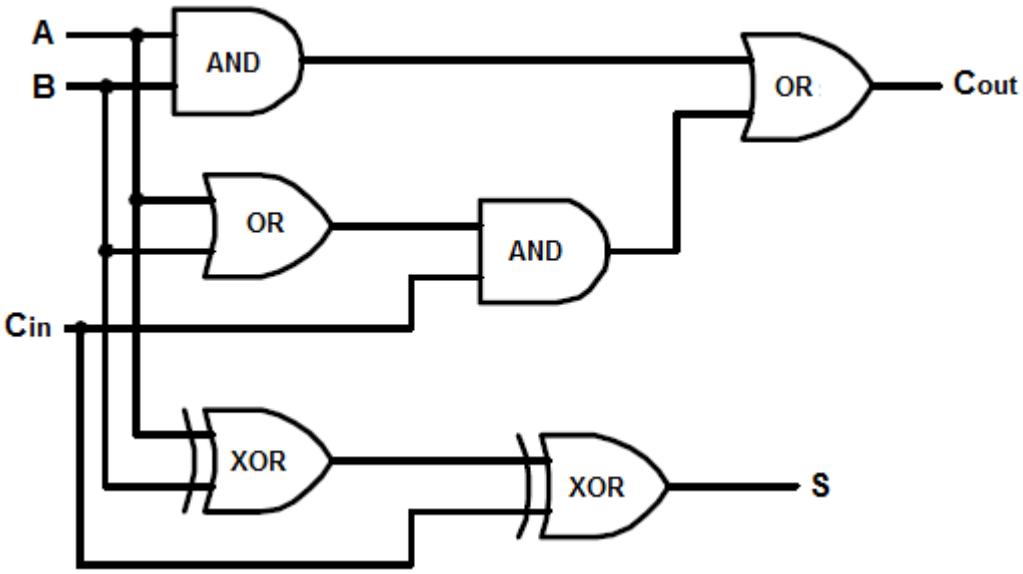


Figura 2. 2: Esquemático de un circuito sumador completo.
Fuente: (Floyd, 2015)

2.2.2. Sumador-restador binario.

La resta de dos números binarios se puede lograr mediante la adición de complemento 2 del sustraendo al minuendo y sin tener en cuenta el acarreo final, si la hay. Si el bit MSB en el resultado de la suma es un '0', entonces el resultado de la suma es la respuesta correcta. Si el bit MSB es un '1', esto implica que la respuesta tiene un signo negativo. La magnitud en este caso está dada por el complemento 2 del resultado de un sumador completo.

Es conocido, que los sumadores completos pueden realizar la resta, siempre y cuando tengamos el hardware adicional necesario para generar el complemento 2 del sustraendo y no tener en cuenta el acarreo final o desbordamiento. En la figura 2.3 se muestra las disposiciones de hardware en el cual vemos cómo se lleva a cabo la resta de dos números binarios de cuatro bits.

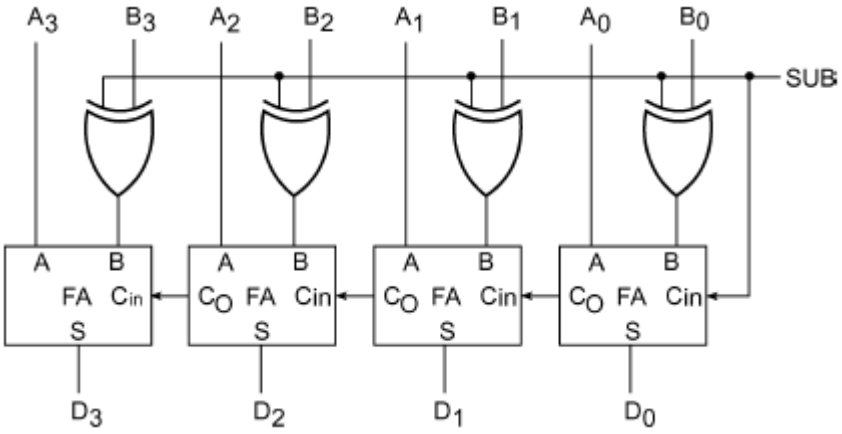


Figura 2. 3: Esquemático de un circuito sumador-restador de 4 bits.
Fuente: (Maini, 2007)

Una mirada cercana en el diagrama revelaría que es la disposición de hardware para un sumador binario de cuatro bits, con la excepción de que los bits de uno de los números binarios son alimentados a través de inversores controlados (Maini, 2007). La entrada de control es referida como la entrada SUB, cuando la entrada SUB está en estado lógico "0", los 4 bits del número binario $(B_3\ B_2\ B_1\ B_0)$ se transmiten como tales a las entradas B de los sumadores completos correspondientes.

Las salidas de los sumadores completos en este caso dan el resultado de la adición de los dos números. Cuando la entrada SUB está en estado lógico "1", cuatro bits de uno de los números, $(B_3\ B_2\ B_1\ B_0)$ en el presente caso, queda complementado. Si el mismo '1' se alimenta también al acarreo (Cin) del LSB en el sumador completo, lo que finalmente alcanzamos es la adición de complemento a 2 y no de complemento a 1.

Por lo tanto, en la disposición mostrada por la figura 2.3 logra realizar la operación A-B, podemos ver que estamos añadiendo básicamente complemento 2 de $(B_3\ B_2\ B_1\ B_0)$ a $(A_3\ A_2\ A_1\ A_0)$. Las salidas de los sumadores completos en este caso dan el resultado de la resta de los dos números. Podemos ver que el acarreo (acarreo de salida del MSB del sumador completo) se ignora si no se visualiza. Para implementar un sumador-restador de ocho bits, se requerirá ocho sumadores completos y ocho XOR de dos entradas. En la figura 2.4 se muestra la implementación

de un sumador de cuatro bits que utiliza comúnmente los integrados 7483 (sumador) y 7486 (XOR) de dos entradas.

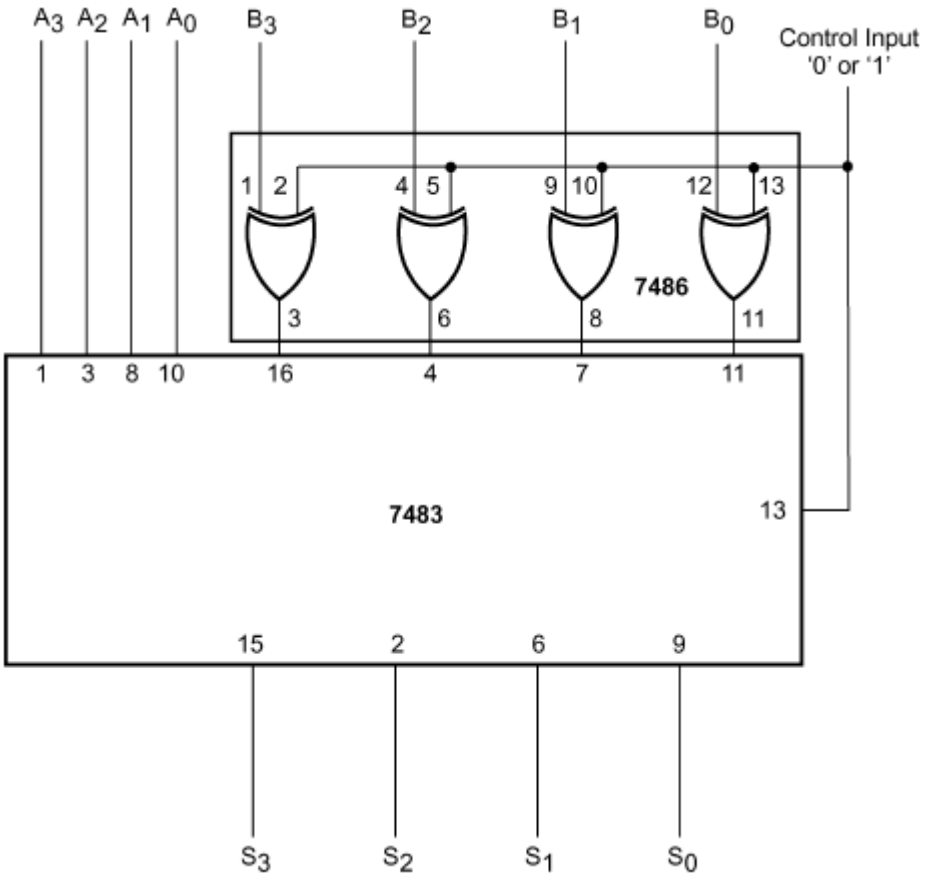


Figura 2. 4: Configuración nueva para un circuito sumador-restador de 4 bits.
Fuente: (Maini, 2007)

2.2.3. Multiplexores.

Muchas de las tareas en las comunicaciones, control y sistemas informáticos pueden ser realizadas por circuitos lógicos combinacionales. Cuando un circuito ha sido diseñado para realizar alguna tarea en una aplicación, que a menudo encuentra uso en una aplicación diferente. De esta manera, adquiere diferentes nombres de sus diversos usos. En esta y en las siguientes secciones, describiremos una serie de tales circuitos y sus

utilidades. Discutiremos sus principios de funcionamiento, especificando sus implementaciones de MSI o LSI. Una tarea común se ilustra en la figura 2.5.

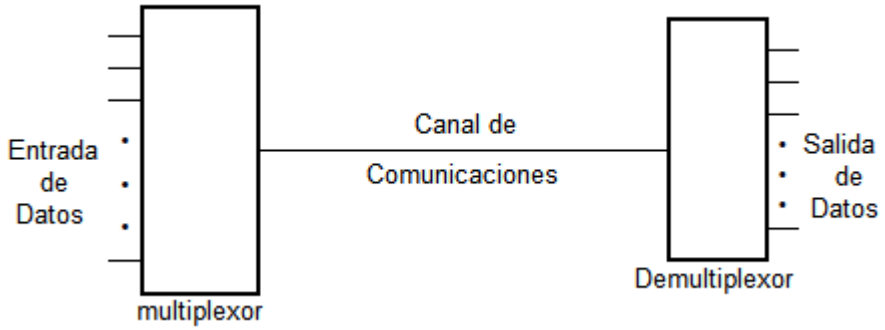


Figura 2. 5: Aplicación común de multiplexor y demultiplexores en un sistema de comunicaciones.
Fuente: (Maini, 2007)

Una aplicación muy utilizada en sistemas de telecomunicaciones, es el multiplexor (ver figura 2.5). Es decir, que los datos generados en un solo lugar se utilizarán en otro lugar, para lo cual se requiere de un método para transmitir de un lugar a otro a través de algún canal de comunicaciones. Los datos están disponibles, en paralelo, en muchas líneas diferentes, pero debe ser transmitida a través de un único enlace de comunicaciones.

Adicionalmente, se requiere de un mecanismo para seleccionar cuál de las muchas líneas de datos para activar secuencialmente a cualquier momento de manera que los datos de esta línea de acarreo se pueden transmitir en ese momento y a este proceso se denomina multiplexación. Otra aplicación práctica de la multiplexación en telecomunicaciones, es la conversación en sistemas telefónicos. Una serie de conversaciones

telefónicas se cambia de forma alterna a la línea telefónica varias veces por segundo.

En el otro extremo del enlace de comunicaciones se necesita un dispositivo que va a deshacer la multiplexación, que se conoce como demultiplexor. Tal dispositivo debe aceptar los datos en serie entrantes y dirigirla en paralelo a una de las muchas líneas de salida. Los fragmentos intercalados de conversaciones telefónicas.

Un multiplexor digital, es un circuito con 2^n líneas de datos de entrada y una línea de salida. También debe tener una forma de determinar la línea de entrada de datos específico que se selecciona al mismo tiempo. Esto se realiza con otras n líneas de entrada, denominadas selector o selectores de entradas, cuya función es la de seleccionar una de las 2^n entradas de datos para la conexión a la salida.

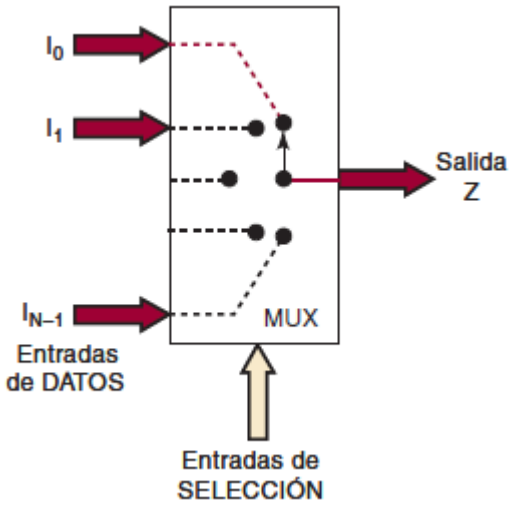
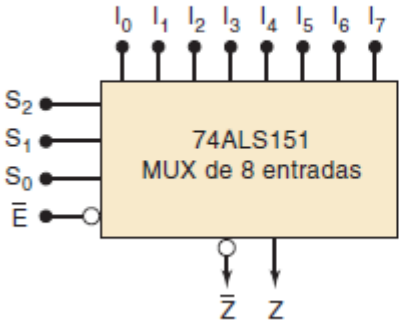


Figura 2. 6: Representación funcional de un multiplexor.
Fuente: (Tocci, Widmer, & Moss, 2011)

Tocci, et al. (2011) muestra en la figura 2.7 el circuito integrado 74ALS151 y su tabla de verdad, donde n=3 es el selector de datos y debe tener $2^n = 8$ combinaciones de valores que constituyen números binarios seleccionados. En la figura 2.8 se muestra el diseño lógico del circuito integrado 74ALS151.



Entradas				Salidas	
$\bar{E}$	S_2	S_1	S_0	$\bar{Z}$	Z
H	X	X	X	H	L
L	L	L	L	$\bar{I}_0$	I_0
L	L	L	H	$\bar{I}_1$	I_1
L	L	H	L	$\bar{I}_2$	I_2
L	L	H	H	$\bar{I}_3$	I_3
L	H	L	L	$\bar{I}_4$	I_4
L	H	L	H	$\bar{I}_5$	I_5
L	H	H	L	$\bar{I}_6$	I_6
L	H	H	H	$\bar{I}_7$	I_7

Figura 2. 7: Representación del circuito multiplexor 74ALS151 de 8 entradas.
Fuente: (Tocci et al., 2011)

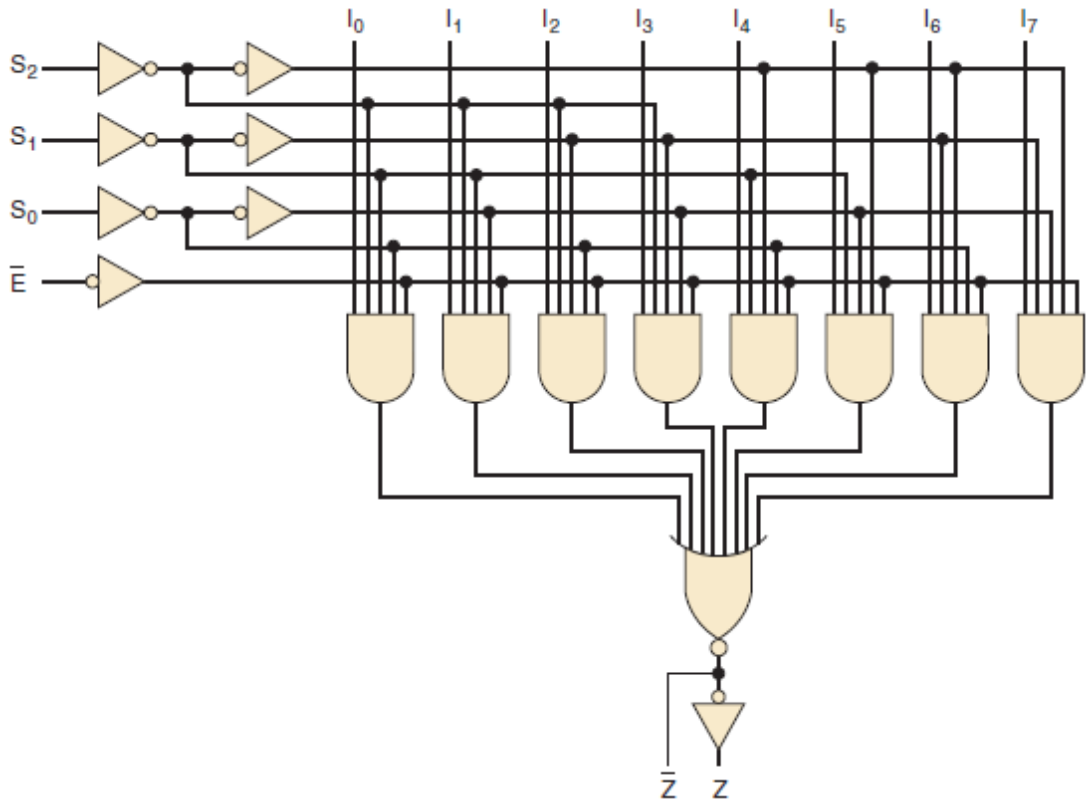


Figura 2. 8: Circuito digital para el multiplexor 74ALS151.
Fuente: (Tocci et al., 2011)

2.3. Visión general de los dispositivos lógicos programables.

Hay muchos tipos de dispositivos lógicos programables (PLDs), distinguibles unas de otras en cuanto a la arquitectura, la capacidad lógica, capacidad de programación y algunas otras características. Para Ballesteros L. & Piraján A. (2004) un PLD dispone de una arquitectura general predefinida, donde el usuario puede configurar a partir de un conjunto de herramientas de desarrollo, dentro de estos dispositivos encontramos: PLAs, PROMs, PALs, GALs, CPLDs y FPGAs según orden de complejidad y versatilidad

A continuación, se describirán brevemente los PLDs comúnmente utilizados y sus características más destacadas.

2.3.1. Memorias ROMs programables.

Las memorias PROM (Programmable Read Only Memory) y EPROM (Erasable Programmable Read Only Memory) pueden ser consideradas como los precursores de PLDs. La arquitectura de una ROM programable permite al usuario hardware implementar una función combinacional arbitraria de un número determinado de entradas.

La arquitectura de los circuitos programables PROM implica la programación de las conexiones con la compuerta OR. Las compuertas AND se utilizan para decodificar todas las posibles combinaciones de las variables de entrada, tal como se muestra en la figura 2.9 (Tocci et al., 2011).

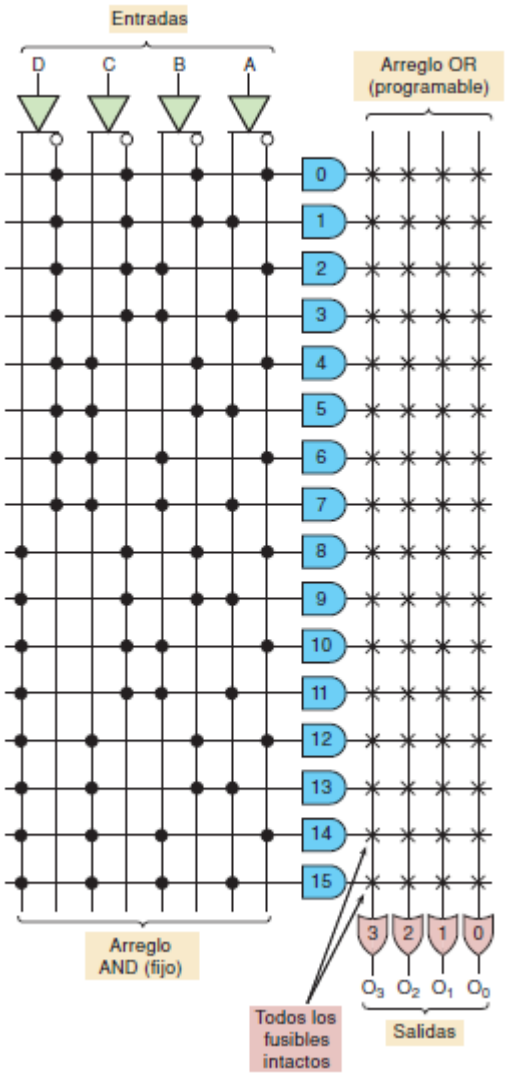


Figura 2. 9: Arquitectura de una memoria programable – PROM.
Fuente: (Tocci et al., 2011)

2.3.2. Matriz lógica programable - PLA.

Una matriz lógica programable *(Programmable Logic Array, PLA)* tiene un conjunto de compuertas AND programables, que une a una matriz programable de compuertas OR, que luego son condicionalmente complementadas para producir una salida (Parmar, Kumar, & Indubala, 2015).

Similar a Parmar et al. (2015), Maini (2007) coincide que la arquitectura PLA se diferencia de la de una PROM en los siguientes aspectos; primero que tiene una matriz programable AND en lugar de una matriz cableada AND. El número de compuertas AND en una PROM de m-entradas es siempre igual a 2^m. En la figura 2.10 se muestra la arquitectura de una PLA.

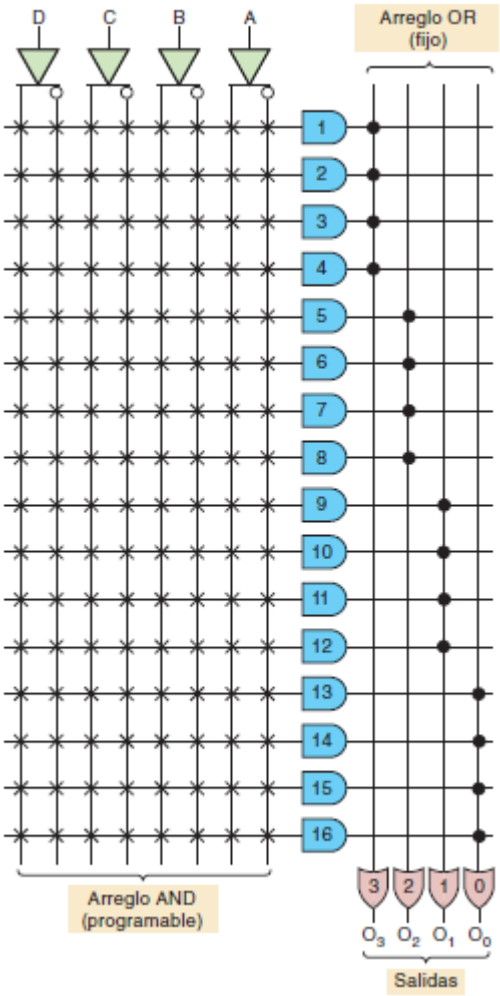


Figura 2. 10: Arquitectura de una matriz lógica programable – PLA.
Fuente: (Tocci et al., 2011)

2.3.3. Matriz Lógica Genérica – GAL.

Un dispositivo de matriz lógica genérica (GAL), de acuerdo a Tocci, et al. (2011) es similar a un dispositivo PAL y fue inventado por Lattice

Semiconductor. En la figura 2.11 se muestra el diagrama de bloques funcional de una GAL, se diferencia de un dispositivo PAL en el que la matriz programable y de un dispositivo GAL puede ser borrado y reprogramado. Además, tiene lógica de salida reprogramable. Esta característica lo hace particularmente atractivo en la fase de creación de prototipos de dispositivos, cualquier error en la lógica pueden ser corregidos mediante la reprogramación.

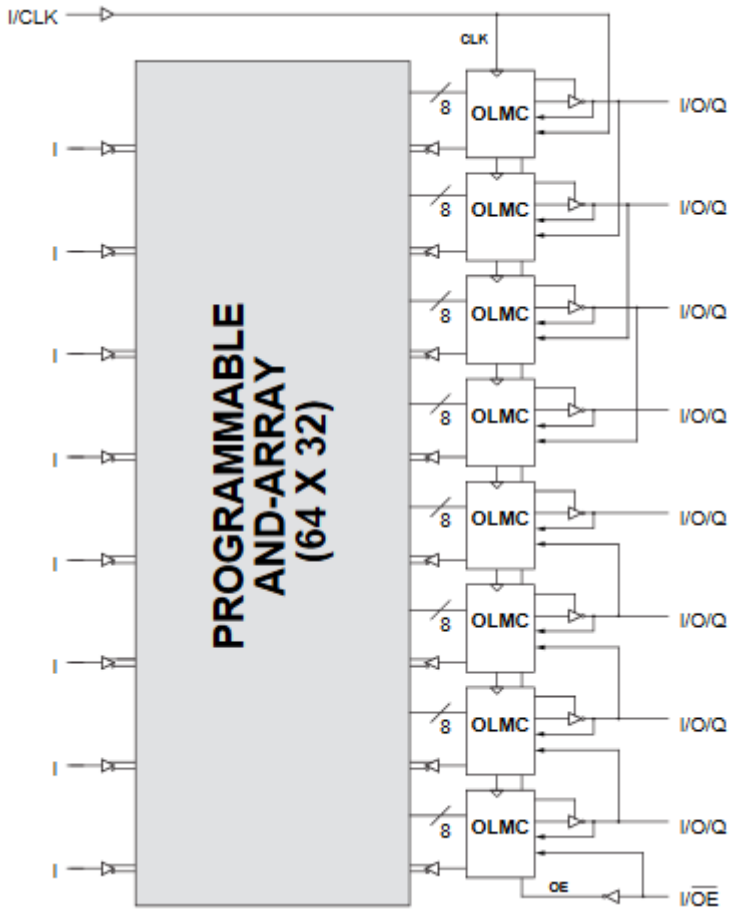


Figura 2. 11: Diagrama de bloques de una matriz lógica genérica – GAL.
Fuente: (Lattice, 2013)

En la figura 2.12 se muestran tres configuraciones para el circuito integrado GAL 16V8, que son: PLCC, DIP y SOIC.

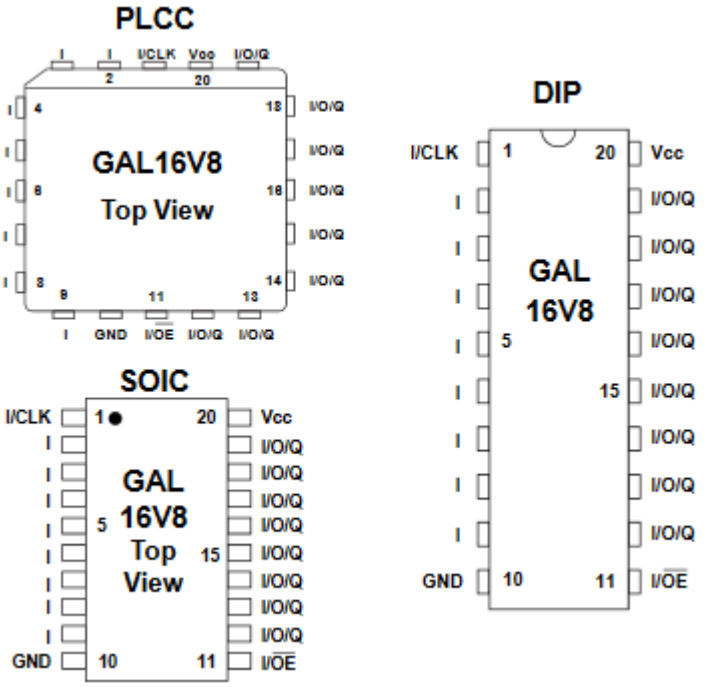


Figura 2. 12: Configuraciones de una matriz lógica genérica – GAL16V8.
Fuente: (Lattice, 2013)

2.3.4. Dispositivo Lógico Programable Complejo.

Los dispositivos lógicos programables, tales como PLA, PAL, GAL y otros dispositivos parecidos a un PAL a menudo se agrupan en una sola categoría llamados dispositivos lógicos programables simples (SPLDs) para distinguirlos de los que son mucho más complejas. Un dispositivo lógico programable complejo (CPLD), como su nombre indica, es un dispositivo mucho más complejo que cualquiera de los dispositivos lógicos programables discutido hasta ahora.

Una CPLD puede contener circuitería equivalente a la de varios dispositivos PAL vinculadas entre sí por interconexiones programables. En la figura 2.13 se muestra la estructura interna de un típico CPLD. Cada uno de

los cuatro bloques lógicos es equivalente a un PLD tal como un dispositivo PAL. El número de bloques lógicos en un CPLD podría ser más o menos que cuatro. Cada uno de los bloques lógicos programables tiene interconexiones. Una matriz de conmutación se utiliza para el bloque lógico de las interconexiones de bloques de lógica.

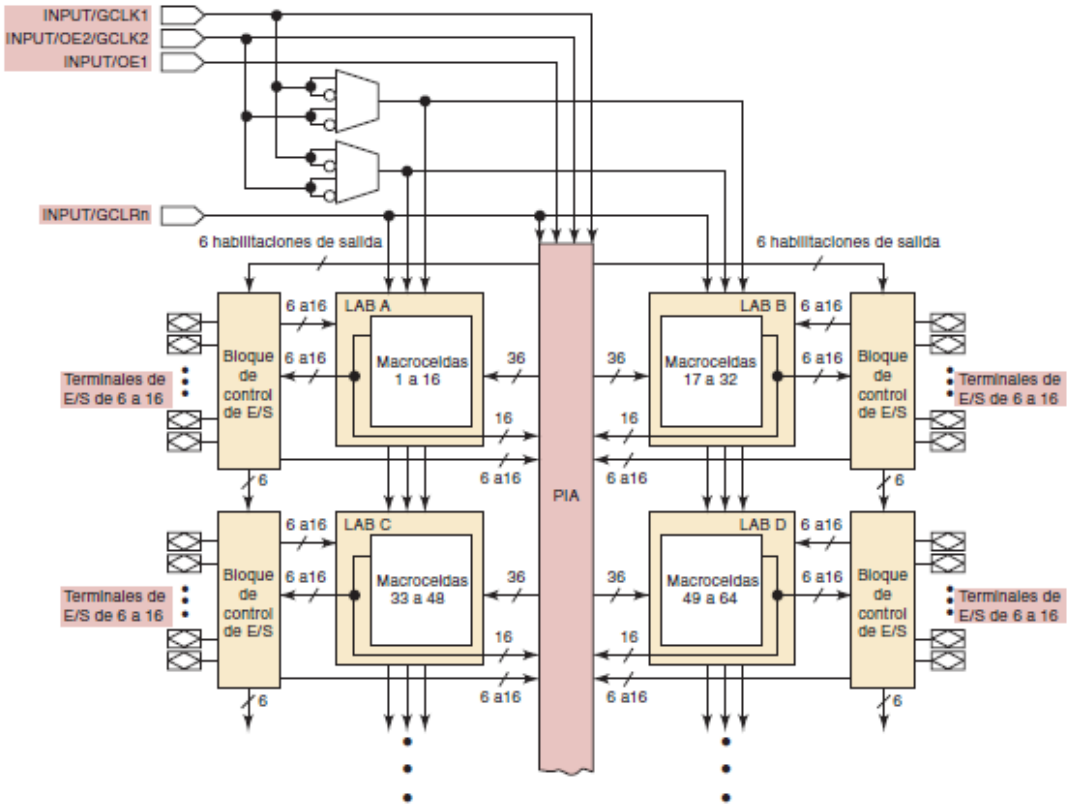


Figura 2. 13: Diagrama de bloques del CPLD de la familia MAX7000S de Altera.
Fuente: (Lattice, 2013; Tocci et al., 2011)

Según Parmar et al. (2015) en su publicación, manifiesta que las PALs y GALs sólo están disponibles en tamaños pequeños, equivalente a unos pocos cientos de compuertas lógicas. También, Parmar et al. (2015) confirma que, para circuitos lógicos más grandes, los PLDs complejos o también conocidos como CPLDs son las más utilizados. Estos contienen el equivalente de varios PALs unidos por interconexiones programables, todo

en un solo circuito integrado. Los CPLDs puede reemplazar a miles o incluso cientos de miles, de compuertas lógicas. En la tabla 2.2 se muestran las características más relevantes de la tarjeta CPLD de la familia MAX7000S.

Tabla 2. 2: Características del CPLD MAX7000S de Altera.

Característica	EPM7032S	EPM7064S	EPM7128S	EPM7160S	EPM7192S	EPM7256S
Compuertas utilizables	600	1250	2500	3200	3750	5000
Macroceldas	32	64	128	160	192	256
LABs	2	4	8	10	12	16
Máximo número de terminales de E/S de usuario	36	68	100	104	124	164

Fuente: (Tocci et al., 2011)

2.3.5. Arreglo de compuertas programables en campo - FPGA.

La tecnología FPGA utiliza sofisticadas herramientas de simulación y verificación del diseño que permiten a los ingenieros para llegar a nuevos niveles de complejidad y robustez, al tiempo que reduce considerablemente el tiempo entre el desarrollo y la utilización (Hernández Z., Camacho N., Huerta R., & Carvallo D., 2015).

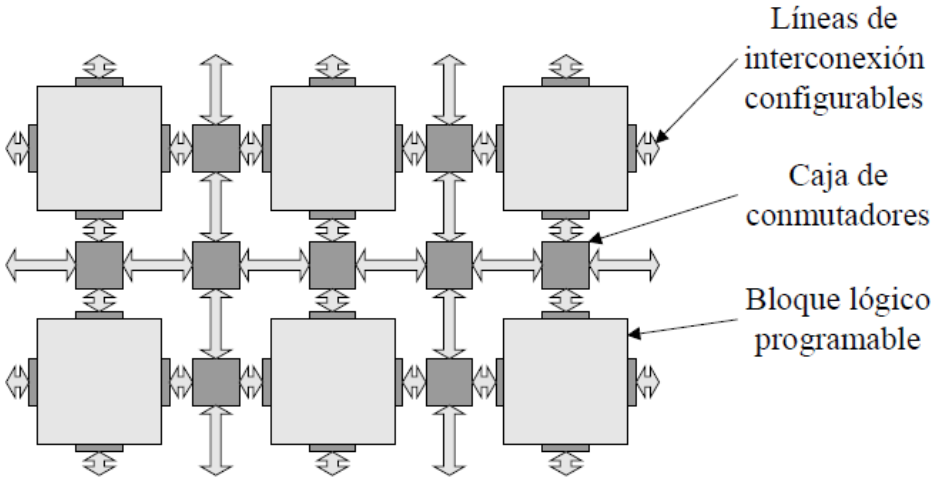


Figura 2. 14: Organización lógica de una FPGA.
Fuente: (Benalcázar P. & Andrade C., 2013)

En la tesis de Benalcázar P. & Andrade C. (2013) indican que una FPGA se definen por su organización lógica, tamaño, recursos de procesamiento, almacenamiento, control, velocidad y consumo de energía. En la misma tesis, indican que la FPGA dispone de CLBs (bloques lógicos configurables) acoplados por líneas de interconexión y cajas de conmutadores, tal como se muestra en la figura 2.14.

Resumiendo, las FPGAs utilizan matrices de bloques lógicos, que el programador las puede configurar. También, el término "campo programable" indica que el dispositivo se puede programar fuera de la fábrica de donde fue construido. La arquitectura interna de un dispositivo FPGA tiene tres partes principales, a saber, el conjunto de bloques lógicos, las interconexiones programables y los bloques de E/S tal como se muestra en la figura 2.15.

Benalcázar P. & Andrade C. (2013) indica que existen dos etapas para diseñar sistemas digitales utilizando matrices lógicas programables, que son:

a. Dividir el circuito en bloques básicos, asignándolos a los bloques configurables del dispositivo.
b. Conectar los bloques de lógica mediante los conmutadores necesarios.

Las FPGAs también permiten la integración de múltiples elementos en un solo chip, con la capacidad de añadir o quitar módulos de acuerdo a las necesidades futuras.

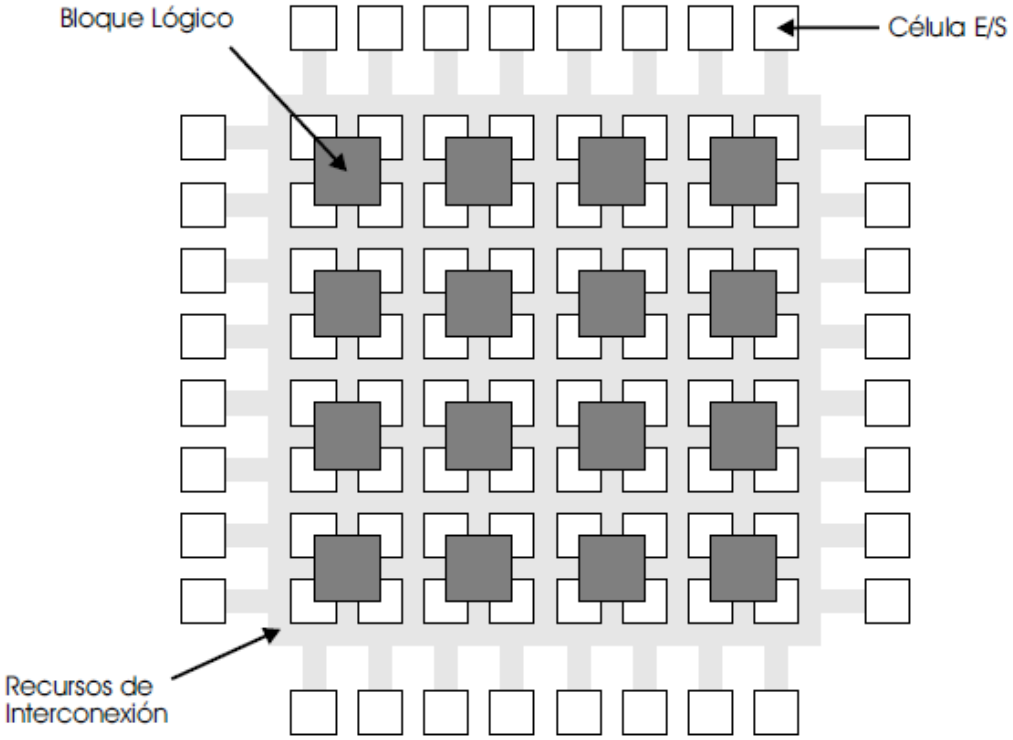


Figura 2. 15: Arquitectura de una FPGA.
Fuente: (Benalcázar P. & Andrade C., 2013)

De la figura 2.15, cada uno de los bloques de E/S proporciona una entrada individual seleccionable, salida o acceso bidireccional para uno de los pines de E/S de propósito general sobre el encapsulado de la FPGA. Mientras que los bloques lógicos de la FPGA, no son más que un par de puertas lógicas o una tabla de búsqueda *(Look-Up Table, LUT)* que alimentan un Flip-Flop, tal como se ilustra en la figura 2.16.

Finalmente, las interconexiones programables conectan bloques lógicos con bloques lógicos y también bloques de E/S a bloques lógicos. Las FPGAs ofrecen una densidad lógica mucho mayor y características de rendimiento mucho más grandes en comparación con las CPLDs. Algunos de los dispositivos FPGAs contemporáneos ofrecen una lógica compleja equivalente a la de 8 millones de compuertas lógicas. Además, estos dispositivos ofrecen características, tales como, gran capacidad de memoria, sistemas de gestión del reloj y el apoyo a muchas de las tecnologías de señalización contemporáneos de dispositivo a dispositivo.

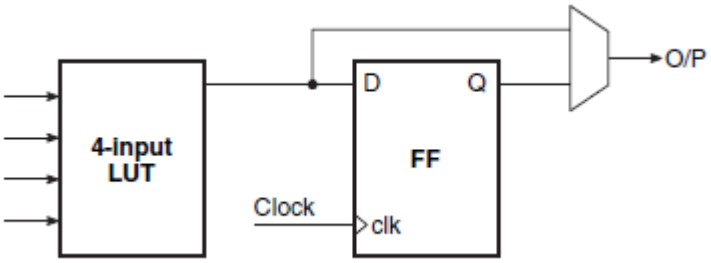


Figura 2. 16: Bloque lógico de un típico FPGA.
Fuente: (Benalcázar P. & Andrade C., 2013)

2.4. Aplicaciones de los dispositivos FPGAs.

En la aparición del dispositivo programable FPGA, comenzó como competidores de los CPLDs debido al aumento de su capacidad lógica y prestaciones, la disponibilidad de una gran cantidad de memoria integrada, de alto nivel de funciones, tales como: sumadores, multiplicadores, aparición de tecnologías híbridas que combinan los bloques lógicos e interconexiones de los FPGA tradicionales con microprocesadores.

El trabajo “Aplicaciones para telecomunicaciones empleando FPGAs: Una aproximación a Radio Software” realizado por Amaya (2006) utiliza el sistema ***SysGen*** para el procesamiento de señales y después ser sintetizadas a una FPGA. ***SysGen*** utiliza Simulink y Xilinx para generar códigos en VHDL, para finalmente ser grabadas en la FPGA. En la figura 2.17 y 2.18 se muestran los diagramas de bloques de la modulación y demodulación BPSK.

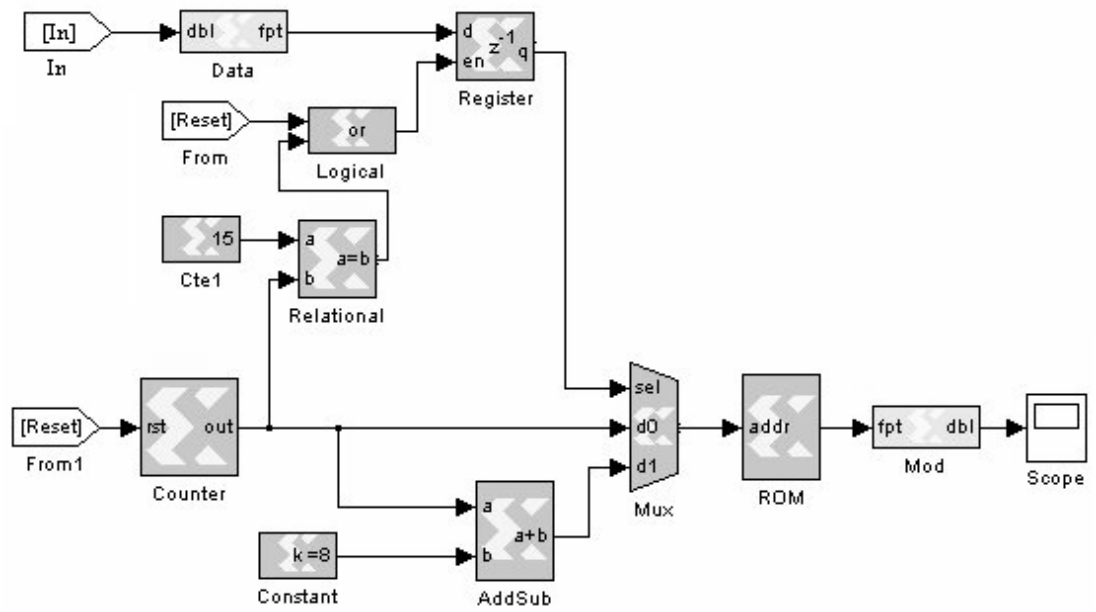


Figura 2. 17: Aplicación FPGA de modulación BPSK usando diagrama de bloque.
Fuente: (Amaya, 2006)

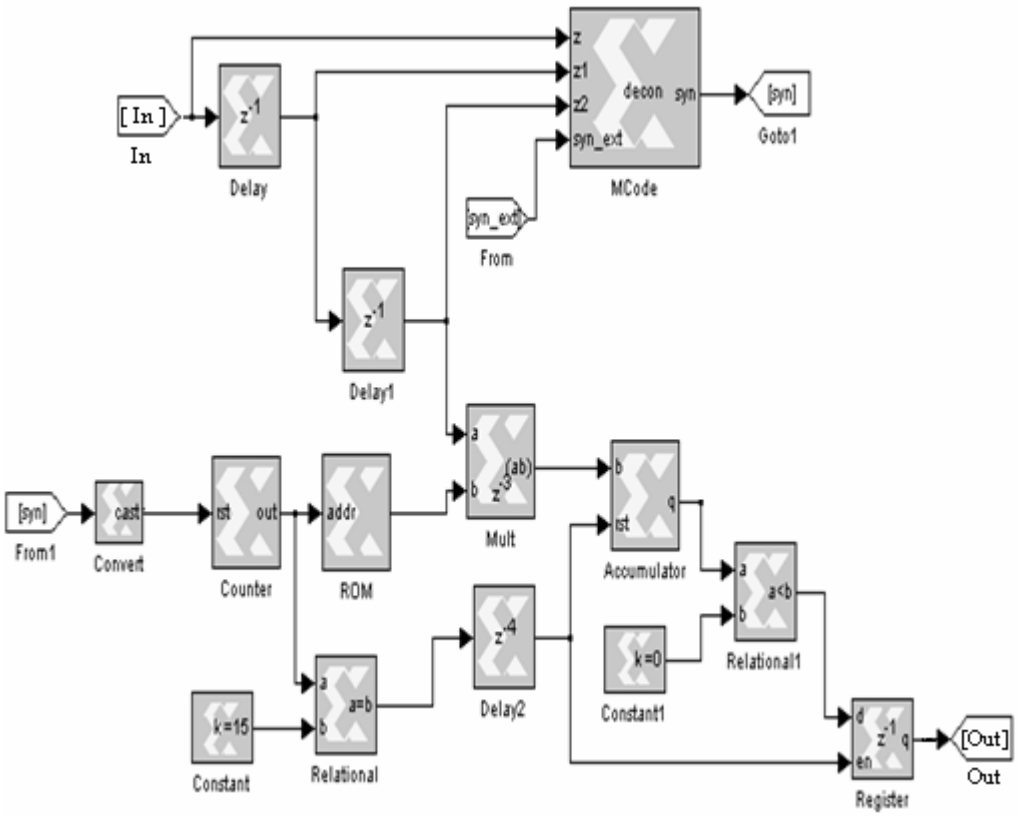


Figura 2. 18: Aplicación FPGA de demodulación BPSK usando diagrama de bloque.
Fuente: (Amaya, 2006)

El trabajo “Aplicación de FPGAs para la conectividad inalámbrica de estación-base” realizado por Newson (2015), donde ha desarrollado diferentes escenarios para que la FPGA se comporte como estación base. En la figura 2.18 se muestra el diagrama de bloques de una macrocelda LTE para estaciones base. La figura 2.16 es solo una muestra de las diferentes configuraciones establecidas para estaciones base, que son expuestas por Newson (2015).

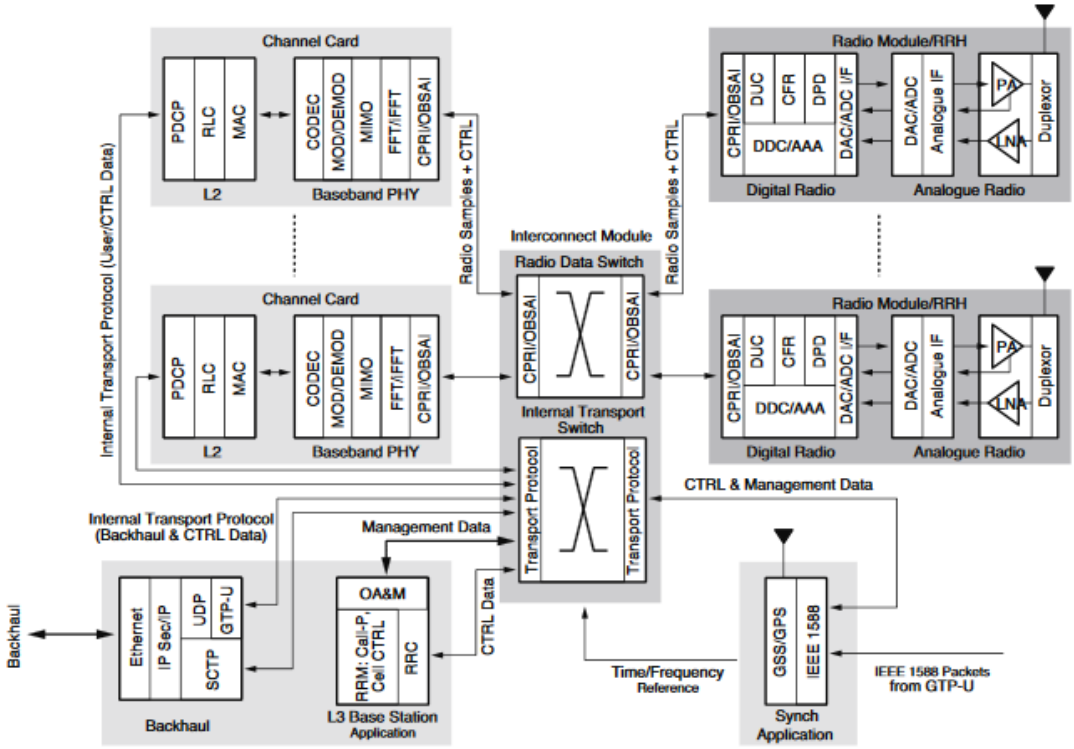


Figura 2. 19: Aplicación FPGA de estación base LTE usando diagramas de bloque.
Fuente: (Newson, 2015)

El artículo denominado “Implementation of the OFDM Physical Layer using FPGA” presentado por los autores Mohamed, Samarah, & Fath (2012). En este trabajo el diseño del sistema OFDM fue simulado en MatLab para estudiar el efecto de varios parámetros de diseño en el rendimiento del sistema, y la implementación del transmisor-receptor OFDM se utilizó la tarjeta FPGA Spartan 3A. Todos los módulos están diseñados utilizando lenguaje de programación VHDL.

Los sistemas FPGAs hoy ofrecen una solución completa en un solo chip, a pesar de sistemas muy complejos que podrían implementarse con más de un dispositivo FPGA. Algunas de las principales áreas de aplicación

de los dispositivos FPGA incluyen: (a) procesamiento de señales digitales, (b) procesamiento y almacenamiento de datos, (c) radio definido por software, (d) prototipos ASIC, (e) reconocimiento de voz, (f) visión por ordenador, (g) criptografía, (h) imágenes médicas, (i) sistemas de defensa, (j) emulación de hardware computacional reconfigurable, (k) telecomunicaciones. La computación reconfigurable, también llamada computación personalizada, implica el uso de piezas programables para ejecutar software en lugar de compilar el software que se ejecuta en una CPU.

2.5. Plataformas de simulación.

En el capítulo 3 se describirá las plataformas de simulación Simulink de MatLab y de Quartus II de Altera. Estas dos herramientas funcionan para diseñar diferentes aplicaciones de circuitos digitales. El presente trabajo de titulación, servirá para desarrollar aplicaciones prácticas de la asignatura sistemas digitales, ya sea a nivel de simulación e implementación en una FPGA. También, permitirá a los estudiantes de las Carreras de Telecomunicaciones, Eléctrico-mecánica y Electrónica en Control y Automatismo, tener como referente a Simulink que permite simular el diseño de sistemas digitales y es compatible con las tarjetas FPGA, Xilinx y Altera.

2.6. Descripción de la programación VHDL.

El Lenguaje de descripción de hardware VHDL *(**V**HSIC **H**ardware **D**escription **L**anguage)*, es un lenguaje de programación potente, usado para

describir el diseño de hardware casi de la misma manera cuando utilizamos diseño esquemático. Fue desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) como estándar VHDL-1076, inicia en 1987 como Std 1076-1987, y actualizado en 1993 como Std 1076-1993. Su popularidad deriva del hecho de que se puede utilizar para la documentación, verificación y síntesis de grandes diseños digitales.

El modelado de circuitos digitales con VHDL, es una forma moderna de diseño digital distinta de los enfoques basados en diseños esquemáticos. El programador escribe una descripción de lo que el circuito lógico final debe hacer y de un compilador de lenguaje, también llamado sintetizador. Los usuarios sin experiencia no siempre son capaces de convencer al sintetizador para poner en práctica algo que parece muy claro en sus mentes.

CAPÍTULO 3: DESARROLLO Y EVALUACIÓN

3.1. Introducción

Comúnmente en clases de Sistemas Digitales I se aprendió a simular circuitos digitales utilizando el programa ISIS Proteus y Multisim. Durante la búsqueda de información, encontré que Simulink (herramienta de MatLab) también permite simular circuitos digitales mediante diseño esquemático, similar a ISIS Proteus. La diferencia entre ISIS Proteus, Multisim y Simulink, es que los dos primeros solo pueden modelar circuitos ya sean eléctricos y electrónicos (analógicos y digitales). Mientras que Simulink permite simular no solamente circuitos eléctricos y electrónicos (circuitos digitales, sean estos, a nivel de compuertas lógicas, sistemas combinatoriales y secuenciales), sino que también modela cualquier sistema en las áreas de Electricidad, Electrónica y Telecomunicaciones.

En la actualidad, se utiliza para simulación ISIS Proteus, mientras que Quartus II se utiliza para desarrollar simulación mediante esquemático, programación VHDL, diagrama de estados, entre otras y posteriormente son cargadas en la tarjeta FPGA DE0 de Altera. La intención del trabajo de titulación es demostrar que existe otra herramienta de simulación como Simulink, para que los estudiantes de las tres carreras de la FETD sean un semillero en desarrollar aplicaciones usando MatLab/Simulink.

3.2. Aplicaciones prácticas de simulación usando SIMULINK.

En esta sección se desarrolla 3 simulaciones de circuitos digitales que corresponden al programa de estudios de la asignatura Sistemas Digitales I, las que se pueden utilizar para el aprendizaje en las tres carreras de la FETD.

3.2.1. Diseño esquemático de expresiones booleanas.

En esta sección se realiza el diseño de un circuito digital mediante expresiones booleanas. En la figura 3.1 se muestra el diseño realizado en Simulink. Se tiene dos bloques: variable A *(constant Block 1)* y variable B *(constant block 2)*, 5 compuertas lógicas NAND, y un visualizador (display) para ver el resultado de la simulación. Esta aplicación es sencilla, pero fundamental para que los estudiantes de V Ciclo de las tres carreras de la FETD puedan utilizar la plataforma Simulink.

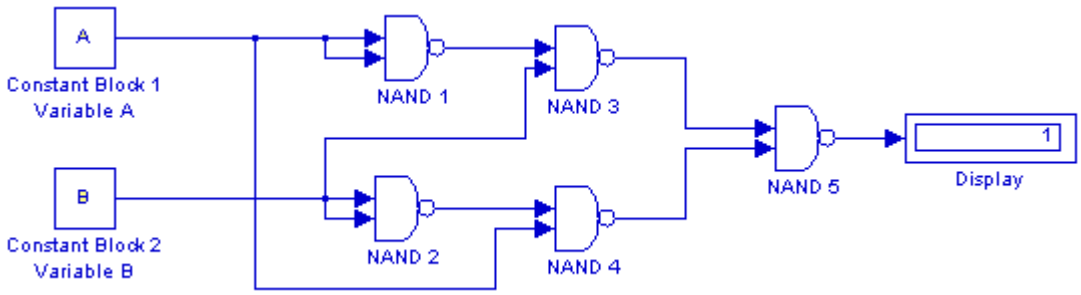


Figura 3. 1: Diseño de la compuerta XOR usando solo NAND.
Elaborado por: Autor.

A partir de la figura 3.1 se muestra la ecuación booleana expresada en suma de productos (SOP).

$$f(A,B) = \overline{\left(\overline{A \cdot \overline{A} \cdot B}\right) \cdot \left(\overline{B \cdot \overline{B} \cdot A}\right)} = \overline{\left(\overline{A\overline{B}}\right) \cdot \left(\overline{\overline{B}A}\right)} = \overline{\overline{A\overline{B}}} + \overline{\overline{\overline{B}A}} = A\overline{B} + \overline{B}A = A \oplus B$$

La tabla de verdad (ver tabla 3.1) indica los posibles valores que asume la simulación. De la figura 3.1 podemos ver que la salida es “1” y de acuerdo a la ecuación booleana corresponde a una compuerta XOR, es decir, que es siempre “1” si se cumple que las variables A=1 y B=0; y viceversa A=0 y B=1.

Tabla 3. 1: Valores de la tabla de verdad de XOR.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Elaborado por: Autor.

La aplicación es muy básica, pero se expone para que los estudiantes que cursan la materia de Digitales I puedan entender el uso de Simulink y a la vez incentivarlos para que profundicen el uso de esta importante herramienta de simulación.

3.2.2. Diagramas de temporización de señales digitales.

En la práctica la mayoría de circuitos electrónicos digitales tanto en la entrada como en la salida del mismo, se utiliza los diagramas de temporización, que normalmente lo vemos en Isis Proteus y Multisim. Simulink también permite visualizar diagramas de temporización, se debe ingresar señales y visualizar al final de acuerdo a la operación lógica.

El diseño realizado en Simulink (véase la figura 3.2) utiliza el bloque Signal Builder, para generar dos señales (señal 1 y señal 2, ver figura 3.3) que son ingresadas en el convertidor de datos a booleanos *(Data Type Conversion)*, a la salida de cada convertidor se envía a las entradas de la compuerta XOR y estas son ingresadas al visualizador de señales *(scope)*.

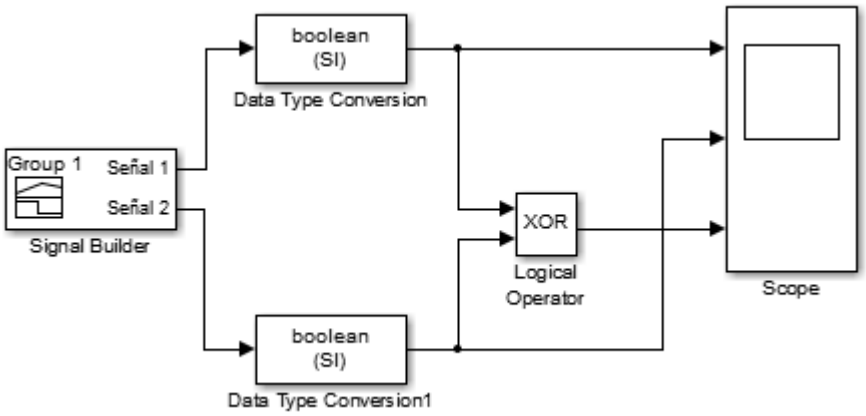


Figura 3. 2: Circuito lógico con diagramas de temporización.
Elaborado por: Autor.

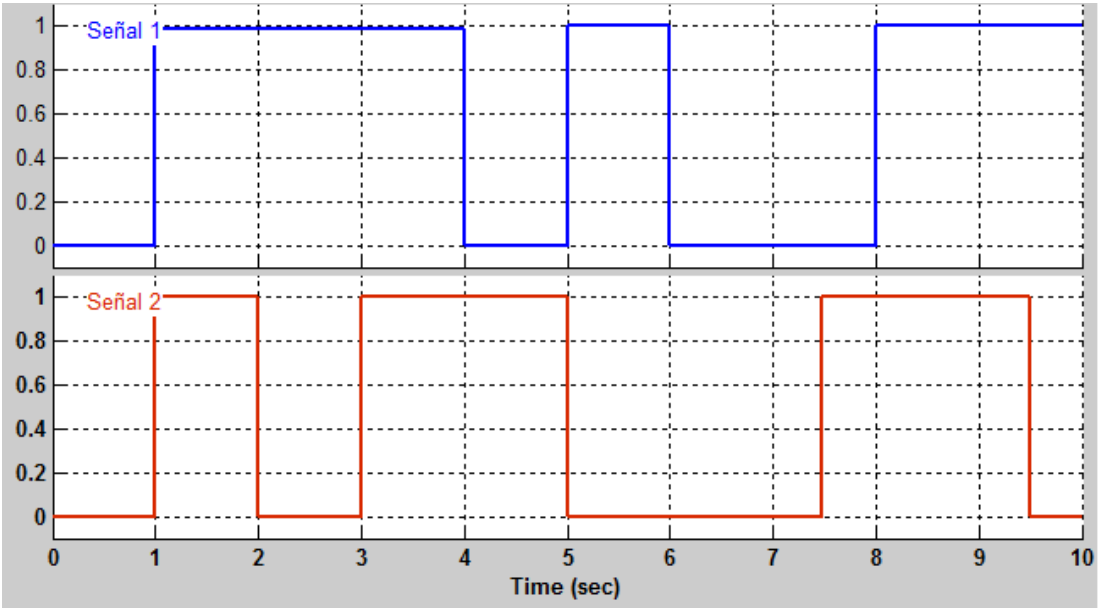


Figura 3. 3: Señales de entrada generadas por Signal Builder.
Elaborado por: Autor.

En la figura 3.4 se muestra el resultado de la operación lógica XOR, se puede observar que las dos primeras señales representan las entradas de XOR y en la señal de salida cumple la función de la tabla de verdad para XOR.

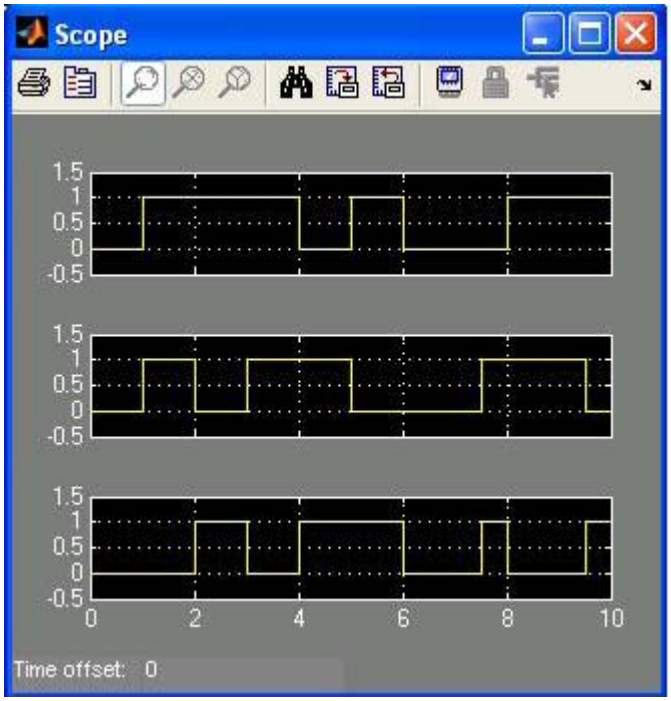


Figura 3. 4: Diagrama de tiempos de la función XOR.
Elaborado por: Autor.

3.3. Aplicaciones prácticas de simulación usando Quartus II.

3.3.1. Manejo de matriz de leds.

La siguiente aplicación práctica consiste en el manejo de una matriz de leds. Para mostrar algún tipo de mensaje en la matriz se lo realizará a través de programación en VHDL sobre la plataforma Quartus II. En la implementación, utilizaremos 4 matrices de led de dimensión 8x8 para

presentar en ellos hasta 4 mensajes diferentes, al pulsar la botonera en la tarjeta DE1 de Altera.

DE1 nos sirve de interface para compilar y cargar el programa realizado en VHDL. Los mensajes pueden ser de tamaño indefinido, pero para efectos de nuestra práctica cada mensaje será de 4 caracteres. Cada mensaje se encuentra previamente grabado en la memoria de programa al principio de este, en la parte correspondiente a la arquitectura, y es allí donde se debe hacer los cambios si se desea aumentar la cantidad de mensajes o variar el contenido. En la figura 3.5 se muestra el diagrama esquemático del manejo de matrices de LED de 8x8.

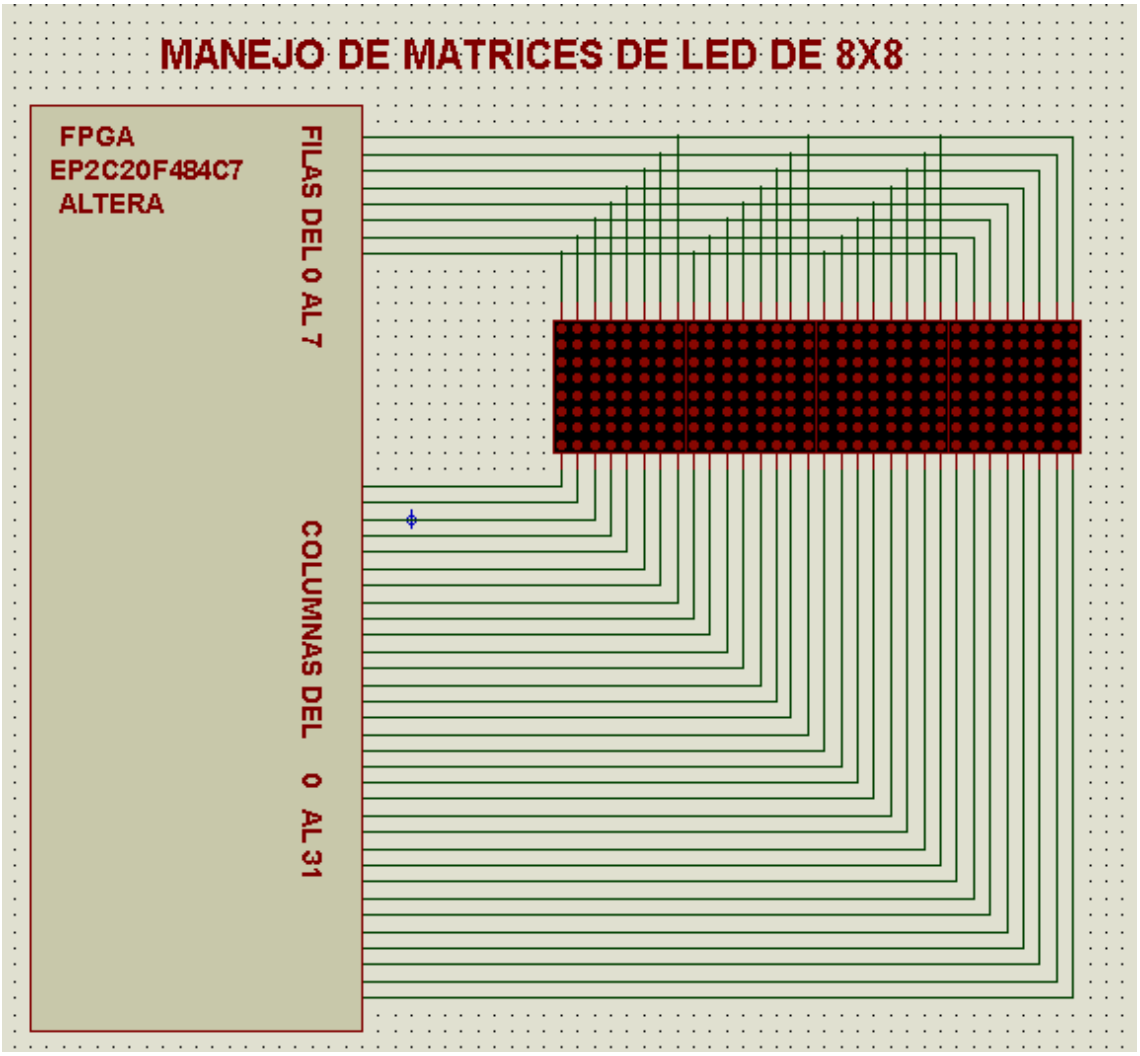


Figura 3. 5: Circuito para el control de matrices led de 8x8.
Elaborado por: Autor.

En el circuito de la figura 3.5, se puede observar la tarjeta Altera DE1 como elemento fundamental en el desarrollo de la práctica, esta controla las 32 columnas de las 4 matrices de led, además de las 8 filas que todas las matrices las tienen en paralelo.

A continuación, se detalla el programa en VHDL para la visualización de los mensajes en la matriz de leds. Después de compilar el programa en el software Quartus II, se procede a configurar el pin planner de la tarjeta DE1.

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

entity MATRIZ is port (
clk : in std_logic;
columnas: inout bit_vector (31 downto 0);
fila: out bit_vector(7 downto 0);
push: in std_logic;
--columnas: inout integer range 0 to 255;
led: out std_logic);
end MATRIZ;

architecture arq_maq of MATRIZ is

type memoria is array(127 downto 0) of bit_vector(7 downto 0);

signal reloj: std_logic;
signal cont: integer range 0 to 32;
signal sel: integer range 0 to 256;
signal internal: integer range 0 to 27000000;
signal fi : memoria;

begin
  --  cont <= 1;
--upsg-- MENSAJE 1
 fi(0) <= x"FF"; fi(1) <= x"B0"; fi(2) <= x"B6"; fi(3) <= x"B6"; fi(4) <= x"BE"; fi(5) <= x"BE"; fi(6)
<= x"80"; fi(7) <= x"FF";
          fi(8) <= x"FF"; fi(9) <= x"B9"; fi(10) <= x"B6"; fi(11) <= x"B6"; fi(12) <= x"B6"; fi(13) <=
x"B6"; fi(14) <= x"CE"; fi(15) <= x"FF";
            fi(16) <= x"FF"; fi(17) <= x"8F"; fi(18) <= x"B7"; fi(19) <= x"B7"; fi(20) <= x"B7"; fi(21)
<= x"B7"; fi(22) <= x"80"; fi(23) <= x"FF";
            fi(24) <= x"FF"; fi(25) <= x"81"; fi(26) <= x"FE"; fi(27) <= x"FE"; fi(28) <= x"FE";
fi(29) <= x"FE"; fi(30) <= x"81"; fi(31) <= x"FF";
          --            MENSAJE 2 Hola --4--3.2--1
          fi(32) <= x"00"; fi(35) <= x"B6"; fi(36) <= x"BE"; fi(37) <= x"BE"; fi(38) <= x"80"; fi(39)
<= x"FF";
          fi(40) <= x"FF"; fi(41) <= x"B9"; fi(42) <= x"B6"; fi(43) <= x"B6"; fi(44) <= x"B6"; fi(45)
<= x"B6"; fi(46) <= x"CE"; fi(47) <= x"FF";
            fi(48) <= x"FF"; fi(49) <= x"8F"; fi(50) <= x"B7"; fi(51) <= x"B7"; fi(52) <= x"B7"; fi(53)
<= x"B7"; fi(54) <= x"80"; fi(55) <= x"FF";
            fi(56) <= x"FF"; fi(57) <= x"81"; fi(58) <= x"FE"; fi(59) <= x"FE"; fi(60) <= x"FE";
fi(61) <= x"FE"; fi(62) <= x"81"; fi(63) <= x"FF";
--          MENSAJE 3 AVANZ--4--3.2--1
          fi(64) <= x"FF"; fi(65) <= x"B0"; fi(66) <= x"B6"; fi(67) <= x"B6"; fi(68) <= x"BE"; fi(69)
<= x"BE"; fi(70) <= x"80"; fi(71) <= x"FF";
          fi(72) <= x"00"; fi(73) <= x"B9"; fi(74) <= x"B6"; fi(75) <= x"B6"; fi(76) <= x"B6"; fi(77)
<= x"B6"; fi(78) <= x"CE"; fi(79) <= x"FF";
            fi(80) <= x"FF"; fi(81) <= x"8F"; fi(82) <= x"B7"; fi(83) <= x"B7"; fi(84) <= x"B7"; fi(85)
<= x"B7"; fi(86) <= x"80"; fi(87) <= x"FF";
```

```
          fi(88) <= x"FF"; fi(89) <= x"81"; fi(90) <= x"FE"; fi(91) <= x"FE"; fi(92) <= x"FE";
fi(93) <= x"FE"; fi(94) <= x"81"; fi(95) <= x"FF";
--          MENSAJE 4 --4--3.2--1
         fi(96) <= x"FF"; fi(97) <= x"B0"; fi(98) <= x"B6"; fi(99) <= x"B6"; fi(100) <= x"BE";
fi(101) <= x"BE"; fi(102) <= x"80"; fi(103) <= x"FF";
         fi(104) <= x"FF"; fi(105) <= x"B9"; fi(106) <= x"B6"; fi(107) <= x"B6"; fi(108) <= x"B6";
fi(109) <= x"B6"; fi(110) <= x"CE"; fi(111) <= x"FF";
          fi(112) <= x"00"; fi(113) <= x"8F"; fi(114) <= x"B7"; fi(115) <= x"B7"; fi(116) <= x"B7";
fi(117) <= x"B7"; fi(118) <= x"80"; fi(119) <= x"FF";
          fi(120) <= x"00"; fi(121) <= x"81"; fi(122) <= x"FE"; fi(123) <= x"FE"; fi(124) <=
x"FE"; fi(125) <= x"FE"; fi(126) <= x"81"; fi(127) <= x"FF";
        --            mi_memoria(8) <= x"AA";

rel: process (reloj)
begin
if (reloj'event and reloj = '1' )then

            fila <= fi(cont+sel);

          case cont is
            when 0 => columnas <=   "00000000000000000000000000000001";

            when 1 => columnas <=  "00000000000000000000000000000010";
            when 2 => columnas <=  "00000000000000000000000000000100";
            when 3 => columnas <=  "00000000000000000000000000001000";
            when 4 => columnas <=  "00000000000000000000000000010000";
            when 5 => columnas <=  "00000000000000000000000000100000";
            when 6 => columnas <=  "00000000000000000000000001000000";
            when 7 => columnas <=   "00000000000000000000000010000000";

            when 8 => columnas <=   "00000000000000000000000100000000";

            when 9 => columnas <=  "00000000000000000000001000000000";
            when 10 => columnas <= "00000000000000000000010000000000";
            when 11 => columnas <= "00000000000000000000100000000000";
            when 12 => columnas <= "00000000000000000001000000000000";
            when 13 => columnas <= "00000000000000000010000000000000";
            when 14 => columnas <= "00000000000000000100000000000000";
            when 15 => columnas <= "00000000000000001000000000000000";
            when 16 => columnas <= "00000000000000010000000000000000";

            when 17 => columnas <= "00000000000000100000000000000000";
            when 18 => columnas <= "00000000000001000000000000000000";
            when 19 => columnas <= "00000000000010000000000000000000";
            when 20 => columnas <= "00000000000100000000000000000000";
            when 21 => columnas <= "00000000001000000000000000000000";
            when 22 => columnas <= "00000000010000000000000000000000";
            when 23 => columnas <= "00000000100000000000000000000000";
```

```
            when 24 => columnas <= "00000001000000000000000000000000";

            when 25 => columnas <= "00000010000000000000000000000000";
            when 26 => columnas <= "00000100000000000000000000000000";
            when 27 => columnas <= "00001000000000000000000000000000";
            when 28 => columnas <= "00010000000000000000000000000000";
            when 29 => columnas <= "00100000000000000000000000000000";
            when 30 => columnas <= "01000000000000000000000000000000";
            when others => columnas <=
"10000000000000000000000000000000";
          end case ;

            cont <= cont + 1;
                  if cont > 31 then
                     cont <= 0;
            end if;

end if;
end process rel;

botonera: process (push)
begin
if(push'event and push='1')then
            sel <= sel + 32;
                  if sel > 127 then
                     sel <= 0;
            end if;
end if;
end process botonera;

conteo: process (clk) is
begin
         if (clk'event and clk='1') then
             internal <= internal +1;
                        if (internal = 7000)then
                           internal <=0;
                                 if (reloj= '0')then
                                          reloj <= '1';
                                          led <= '1';
                                 else
                                          reloj <= '0';
                                          led <= '0';
                                 end if;
                        end if;
         end if;
end process conteo;
end arq_maq;
```

En la tabla 3.2 se muestra la configuración de las botoneras usando el pin planner, sin esta configuración no se podría visualizar ningún mensaje en la matriz.

Tabla 3. 2: Asignaciones de los pines para los pulsadores.

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_R22	Pushbutton[0]
KEY[1]	PIN_R21	Pushbutton[1]
KEY[2]	PIN_T22	Pushbutton[2]
KEY[3]	PIN_T21	Pushbutton[3]

Fuente: (Altera, 2006)

Una vez realizada la configuración del pin planner, se graba el programa sobre el dispositivo Cyclone II, para lo cual se visualizará los cuatro mensajes tal como se muestra en las figuras 3.6 a 3.9.

Primer Mensaje: UCSG

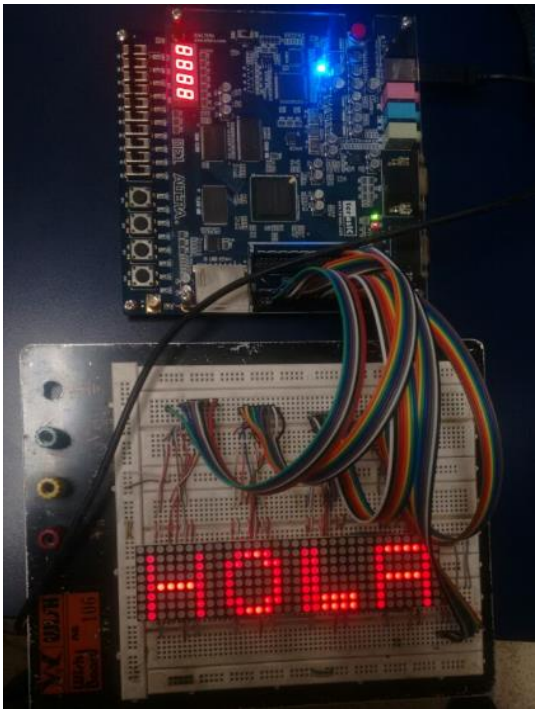


Figura 3. 6: Visualización del primer mensaje ‘HOLA’.
Elaborado por: Autor.

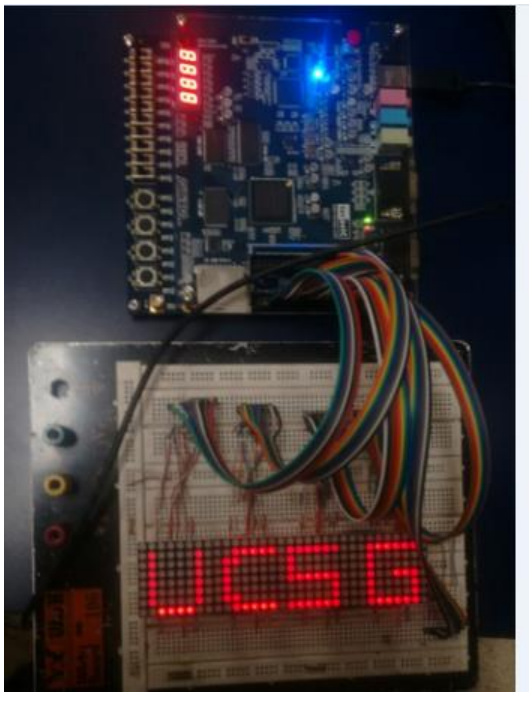


Figura 3. 7: Visualización del segundo mensaje ‘UCSG’.
Elaborado por: Autor.

Figura 3. 8: Visualización del tercer mensaje ‘FETD’.
Elaborado por: Autor.

Figura 3. 9: Visualización del cuarto mensaje ‘2016’.
Elaborado por: Autor.

3.3.2. Control de un parqueadero.

La siguiente aplicación práctica la realizaremos mediante la programación en VHDL en Quartus II y posteriormente ejecutarla sobre la tarjeta DE1 de Altera, la misma que posee el dispositivo Cyclone II EP2C20f484C7.

Esta aplicación, consiste en diseñar un circuito mediante FPGA para el control de un parqueadero con capacidad de 8 vehículos, con indicación mediante LEDs para informar que parqueadero está libre u ocupado. Además, el circuito debe contar con 2 displays de 7 segmentos cada uno, que proporcionen información de cuantos lugares se encuentran desocupados y cuantos ocupados.

El circuito esquemático (véase la figura 3.10) para el control de un parqueadero de vehículos, que, para efectos del trabajo de titulación, se supone un parqueo para 8 vehículos. Se puede observar 8 diodos leds de color rojo, que serán los encargados de indicar al usuario que el sitio donde se parquea el carro está no disponible (ocupado), y 8 leds de color verde para indicar que el espacio de parqueo está disponible (libre).

Además, se dispone de 2 display de 7 segmentos cada uno, los cuales proporcionarán la información de cuantos parqueos están libres y cuantos ocupados. Mediante el uso de 8 switchs, que nos permite emular los sensores de detección de la presencia de un vehículo.

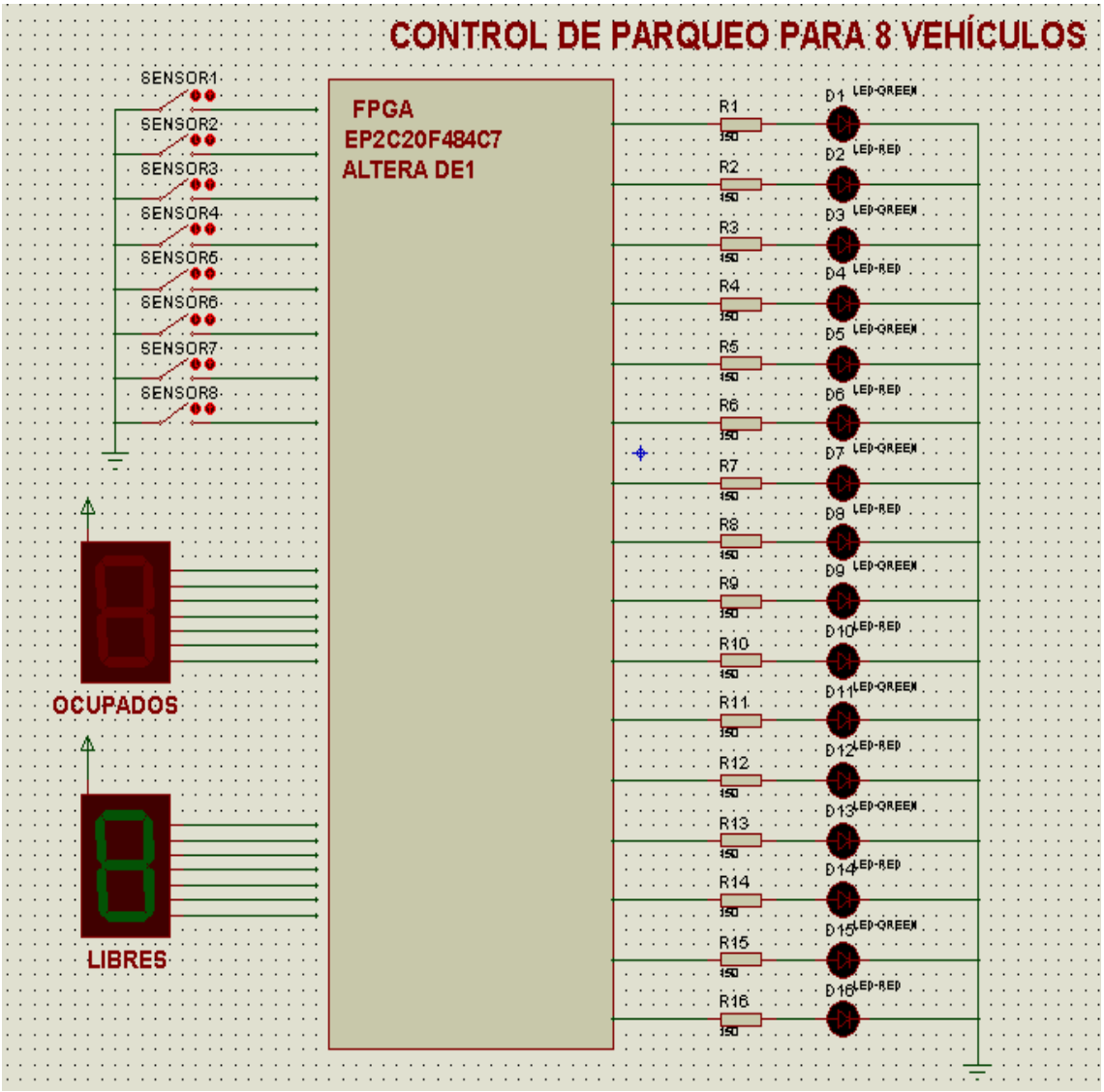


Figura 3. 10: Circuito para el control de parqueo para vehículos.
Elaborado por: Autor.

Del circuito esquemático (véase la figura 3.10), si el switch está en bajo el espacio está disponible y se deberá encender el respectivo led verde, y se considerará para la suma de todos los espacios libres que deberá indicarse en el respectivo display de 7 segmentos verde. Caso contrario, si el switch está en alto se encenderá el respectivo led rojo, y se considerará para la

suma de todos los espacios ocupados en el respectivo display de 7 segmentos rojo.

Para satisfacer el diseño del circuito controlador de parqueos, se realiza el programa en VHDL de Quartus II. A continuación, se presenta el código realizado:

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

entity autos is port (
sensor: in bit_vector (7 downto 0);
total,total2: buffer integer range 0 to 10 ;
d,d2: out bit_vector (6 downto 0);
led_verde, led_rojo: out bit_vector (7 downto 0));
end autos;

architecture arq_auto of autos is
signal b0,b1,b2,b3,b4,b5,b6,b7: integer range 0 to 3;
begin
  led_verde <= not sensor;
  led_rojo <=  sensor;
process(sensor)
begin
  if sensor(0)='0' then  b0<=0;  else  b0<=1;  end if;
  if sensor(1)='0' then  b1<=0;  else  b1<=1;  end if;
  if sensor(2)='0' then  b2<=0;  else  b2<=1;  end if;
  if sensor(3)='0' then  b3<=0;  else  b3<=1;  end if;
  if sensor(4)='0' then  b4<=0;  else  b4<=1;  end if;
  if sensor(5)='0' then  b5<=0;  else  b5<=1;  end if;
  if sensor(6)='0' then  b6<=0;  else  b6<=1;  end if;
  if sensor(7)='0' then  b7<=0;  else  b7<=1;  end if;

  total2 <= b0+b1+b2+b3+b4+b5+b6+b7;
  total <= 8 - total2;
    case total is
                when 0 => d <= "0000001";
                when 1 => d <= "1001111";
                when 2 => d <= "0010010";
                when 3 => d <= "0000110";
                when 4 => d <= "1001100";
                when 5 => d <= "0100100";
                when 6 => d <= "0100000";
```

```
            when 7 => d <= "0001110";
            when 8 => d <= "0000000";
            when 9 => d <= "0000100";
            when others => d <= "1111111";
        end case;

        case total2 is
            when 0 => d2 <= "0000001";
            when 1 => d2 <= "1001111";
            when 2 => d2 <= "0010010";
            when 3 => d2 <= "0000110";
            when 4 => d2 <= "1001100";
            when 5 => d2 <= "0100100";
            when 6 => d2 <= "0100000";
            when 7 => d2 <= "0001110";
            when 8 => d2 <= "0000000";
            when 9 => d2 <= "0000100";
            when others => d <= "1111111";
        end case;
end process;
end arq_auto;
```

Después de realizado el programa en VHDL, se procede a compilar. Posterior a la compilación, debemos configurar los pines usando la asignación de para conmutadores que se muestra en tabla 3.3.

Tabla 3. 3: Asignaciones de los pines para switchs o conmutadores.

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_L22	Toggle Switch[0]
SW[1]	PIN_L21	Toggle Switch[1]
SW[2]	PIN_M22	Toggle Switch[2]
SW[3]	PIN_V12	Toggle Switch[3]
SW[4]	PIN_W12	Toggle Switch[4]
SW[5]	PIN_U12	Toggle Switch[5]
SW[6]	PIN_U11	Toggle Switch[6]
SW[7]	PIN_M2	Toggle Switch[7]
SW[8]	PIN_M1	Toggle Switch[8]
SW[9]	PIN_L2	Toggle Switch[9]

Fuente: (Altera, 2006)

Para la visualización de los indicadores leds de disponible o no disponible, se debe configurar los pines de acuerdo a la tabla 3.4.

Tabla 3. 4: Asignaciones de los pines para leds rojo y verdes.

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_R20	LED Red[0]
LEDR[1]	PIN_R19	LED Red[1]
LEDR[2]	PIN_U19	LED Red[2]
LEDR[3]	PIN_Y19	LED Red[3]
LEDR[4]	PIN_T18	LED Red[4]
LEDR[5]	PIN_V19	LED Red[5]
LEDR[6]	PIN_Y18	LED Red[6]
LEDR[7]	PIN_U18	LED Red[7]
LEDR[8]	PIN_R18	LED Red[8]
LEDR[9]	PIN_R17	LED Red[9]
LEDG[0]	PIN_U22	LED Green[0]
LEDG[1]	PIN_U21	LED Green[1]
LEDG[2]	PIN_V22	LED Green[2]
LEDG[3]	PIN_V21	LED Green[3]
LEDG[4]	PIN_W22	LED Green[4]
LEDG[5]	PIN_W21	LED Green[5]
LEDG[6]	PIN_Y22	LED Green[6]
LEDG[7]	PIN_Y21	LED Green[7]

Fuente: (Altera, 2006)

Posteriormente, se carga y ejecuta el programa en la tarjeta DE1 de Altera. En la figura A continuación, se presenta la tarjeta Altera con el programa cargado

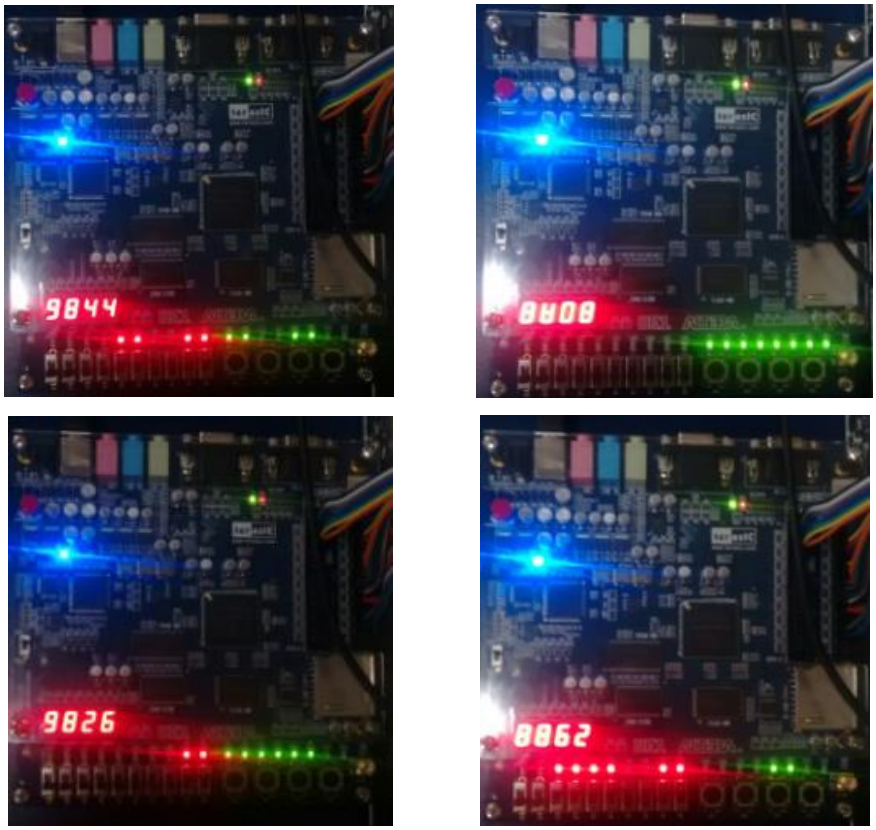


Figura 3. 11: Visualización de los indicadores de parqueo disponible o no disponible.
Elaborado por: Autor.

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES.

4.1. Conclusiones.

- A través de la fundamentación teórica de sistemas combinatoriales y los dispositivos lógicos programables se pudo evidenciar que las teorías con parte simulada son necesarias antes de implementar un circuito sobre un PBC.
- Las plataformas de simulación Simulink y Quartus II permiten realizar aplicaciones en forma conjunta, ya sea para sistemas digitales u otra aplicación en el área de las telecomunicaciones.
- Simulink permite simular sistemas combinatoriales y secuenciales, mientras que Quartus II es más completo al momento de simular cualquier sistema.

4.2. Recomendaciones.

- Incluir en las asignaturas de sistemas digitales el uso de la herramienta de simulación Simulink, como una nueva ayuda para la formación de futuros ingenieros en telecomunicaciones.
- Adquirir la licencia de MatLab/Simulink para que los estudiantes puedan investigar y así generar nuevos proyectos o trabajos de titulación utilizando en forma conjunta Quartus II y la tarjeta FPGA de Altera.

REFERENCIAS BIBLIOGRÁFICAS

Altera. (2006). DE1: Development and Education Board. Recuperado el 9 de agosto de 2016, a partir de ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE1/DE1_User_Manual.pdf

Alvarado B., J. (2016). *Desarrollo de algoritmos en VDHL sobre una FPGA DEO-NANO para prácticas de laboratorio de digitales en la carrera de Ingeniería Electrónica en Control y Automatismo.* Universidad Católica de Santiago de Guayaquil. Recuperado a partir de http://repositorio.ucsg.edu.ec/handle/3317/5340

Amaya, F. (2006). Aplicaciones para telecomunicaciones empleando FPGAs: Una aproximación a Radio Software. *Revista Colombiana de Tecnologías de Avanzada*, *1*(7), 38–43.

Ballesteros L., D., & Piraján A., A. (2004, junio). Diseño VHDL de Sistemas Digitales sobre dispositivos lógicos programables FPGAs. *Umbral Científico*, *4*, 37–49.

Benalcázar P., C., & Andrade C., I. (2013, enero 2). *Diseño e implementación de prácticas con FPGA para la materia de Digital II mediante la herramienta de trabajo Cyclone II de Altera*. Universidad Católica de Santiago de Guayaquil. Recuperado a partir de http://repositorio.ucsg.edu.ec/handle/3317/237

Floyd, T. L. (2015). *Digital fundamentals* (Eleventh edition). Boston: Pearson.

Hernández Z., A., Camacho N., O., Huerta R., J., & Carvallo D., A. (2015). Design of a General Purpose 8-bit RISC Processor for Computer Architecture Learning. *Computación y Sistemas*, *19*(2), 371–385.

Lattice. (2013). GAL16V8 Data Sheet - High Performance E2CMOS PLD Generic Array Logic™. Recuperado el 23 de julio de 2016, a partir de http://ee-classes.usc.edu/ee459/library/datasheets/16v8.pdf

Maini, A. K. (2007). *Digital electronics: principles, devices and applications*. Chichester, England ; Hoboken, NJ: J. Wiley.

Mohamed, M., Samarah, A., & Fath, M. (2012). Implementation of the OFDM Physical Layer Using FPGA. *International Journal of Computer Science Issues, IJCSI*, *9*(2), 612–618.

Newson, P. (2015). The Application of FPGAs for Wireless Base-Station Connectivity. Recuperado el 24 de julio de 2016, a partir de http://www.xilinx.com/support/documentation/white_papers/wp450-base-stn-connect.pdf

Parmar, A., Kumar, D., & Indubala. (2015). Programmable Logic Device. *International Journal Of Innovative Research in Technology (IJIRT)*, *1*(12), 178–180.

Perelroyzen, E. (2006). *Digital Integrated Circuits: Design-for-Test Using Simulink and Stateflow*. CRC Press.

Sosa C., S., & Valdez J., Y. (2015). *Sistema de entrenamiento DE1 de altera : desarrollo de aplicaciones prácticas para el laboratorio de digitales.* Universidad Católica de Santiago de Guayaquil. Recuperado a partir de http://repositorio.ucsg.edu.ec/handle/3317/3731

Tocci, R. J., Widmer, N. S., & Moss, G. L. (2011). *Digital systems: principles and applications* (11th ed). Upper Saddle River, N.J: Prentice Hall.

DECLARACIÓN Y AUTORIZACIÓN

Yo, **REYES ERAZO, CRISTHIAN EDUARDO** con C.C: # 0803044197 autor del Trabajo de Titulación: **EVALUACIÓN DE LAS PLATAFORMAS DE SIMULACIÓN SIMULINK Y QUARTUS II PARA LA ASIGNATURA DE SISTEMAS DIGITALES** previo a la obtención del título de **INGENIERO EN TELECOMUNICACIONES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 12 de Septiembre de 2016

f. _____

Nombre: REYES ERAZO, CRISTHIAN EDUARDO

C.C:

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN			
TÍTULO Y SUBTÍTULO:	EVALUACIÓN DE LAS PLATAFORMAS DE SIMULACIÓN SIMULINK Y QUARTUS II PARA LA ASIGNATURA DE SISTEMAS DIGITALES		
AUTOR(ES)	REYES ERAZO, CRISTHIAN EDUARDO		
REVISOR(ES)/TUTOR(ES)	M. Sc. EDWIN F. PALACIOS MELÉNDEZ		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Ingeniería en Telecomunicaciones		
TITULO OBTENIDO:	Ingeniero en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	12 de Septiembre de 2016	**No. DE PÁGINAS:**	63
ÁREAS TEMÁTICAS:	Instrumentación virtual, Sistemas Electrónicos Digitales y Programación		
PALABRAS CLAVES/ KEYWORDS:	Sistemas Digitales, Simulink, Quartus II, programación VHDL, diseño esquemático, máquinas de estado.		

RESUMEN/ABSTRACT (150-250 palabras):

El presente trabajo de titulación consiste en realizar la evaluación de las plataformas de simulación SIMULINK y QUARTUS II para la asignatura de Sistemas Digitales. La idea del trabajo es fundamental y formativa, porque existe otra herramienta de simulación que no se ha considerado en el programa de estudios de la asignatura Sistemas Digitales I y II. Por lo general, en estas asignaturas los estudiantes utilizan Isis Proteus y Multisim, ambas plataformas son amigables. En la búsqueda de información se pudo constatar que Simulink de MatLab, también permite desarrollar simulaciones de sistemas digitales. Los estudiantes de V Ciclo de Telecomunicaciones, Electrónica en Control y Automatismo y Eléctrico-mecánico pueden hacer uso del presente trabajo como guía y así profundizar más en el tema usando Simulink. Quartus II, es una plataforma que permite realizar programación en VHDL, diseño esquemático y diseño por máquinas de estados y esto a su vez se implementa en la FPGA de Altera disponible en el laboratorio de electrónica. Cabe mencionar que tanto Simulink como Quartus II, realizan múltiples aplicaciones para aplicaciones en telecomunicaciones.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:**	E-mail:
CONTACTO CON LA INSTITUCIÓN: COORDINADOR DEL PROCESO DE UTE	**Nombre:** Palacios Meléndez Edwin Fernando **Teléfono:** +593-9-68366762 **E-mail:** edwin.palacios@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	