



**Instituto Politécnico Nacional**  
**Escuela Superior de Cómputo**



**Diseño de Sistemas Digitales.**  
**Reporte de práctica: “Repaso”.**

**Alumnos.**

**Hernández López Ángel Zait**

**Morelos Ordóñez Pedro Luis**

**Profesor. Mujica Ascencio César.**

## Introducción.

El álgebra booleana de dos valores se define sobre un conjunto de dos elementos.  $A = \{1,0\}$ , con las reglas para los dos operadores binarios,  $+$  y  $*$ , mostradas en las siguientes tablas de operador.

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1.1

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 1.2

Estas reglas son exactamente las mismas que las operaciones AND y OR respectivamente.

Las tablas de verdad (1.1 y 1.2) se pueden demostrar por los postulados siguientes.

Postulado 1 (a)  $x + 0 = x$ . (b)  $x * 1 = x$

Postulado 2 (a)  $x + 1 = 1$  (b)  $x * 0 = 0$

Ampliando la información de las compuertas mencionadas y las otras 6 compuertas básicas obtenemos:



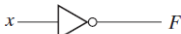
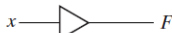
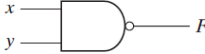



Nombre	Símbolo Gráfico	Función Algebraica	Tabla de verdad															
AND		$F = XY$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	F	0	0	0	0	1	0	1	0	0	1	1	1
X	Y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = X + Y$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	F	0	0	0	0	1	1	1	0	1	1	1	1
X	Y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inversor		$F = X'$	<table><tr><th>X</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	X	F	0	1	1	0									
X	F																	
0	1																	
1	0																	
Búfer	 $F = x$	$F = X$	<table><tr><th>X</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	X	F	0	0	1	1									
X	F																	
0	0																	
1	1																	
NAND		$F = (XY)'$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X	Y	F	0	0	1	0	1	1	1	0	1	1	1	0
X	Y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (X + Y)'$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X	Y	F	0	0	1	0	1	0	1	0	0	1	1	0
X	Y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
OR exclusiva		$F = X \oplus Y$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X	Y	F	0	0	0	0	1	1	1	0	1	1	1	0
X	Y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
NOR exclusiva		$F = X \odot Y$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	F	0	0	1	0	1	0	1	0	0	1	1	1
X	Y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Tabla 1.3

Los circuitos lógicos para sistemas digitales pueden ser combinacionales o secuenciales. Un circuito combinacional consiste en compuertas lógicas cuyas salidas en cualquier momento están determinadas por la combinación actual de las entradas, además de realizar una operación que se puede especificar lógicamente con un conjunto de funciones booleanas.

Un circuito combinacional consiste en variables de entrada, compuertas lógicas y variables de salida, las compuertas lógicas aceptan señales de las entradas y generan señales para las salidas. Este proceso transforma información binaria, de los datos de entrada a los datos de las salidas requeridos.

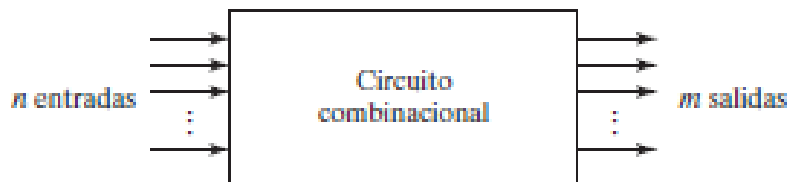


Figura 1.1 Diagrama de bloques de Cto. Combinacional

Un sumador binario es un circuito que realiza la operación aritmética de suma con números binarios (1 y 0). Si se quiere realizar la suma de  $n$  dígitos binarios, existen dos posibles soluciones al problema, las cuales llevan por nombre semisumador y sumador completo. Un circuito combinacional que realiza la suma de  $n$  bits se denomina semisumador, un circuito que realiza la suma de  $n+1$  bits es un sumador completo. Dichos nombres provienen del hecho que es posible usar dos semisumadores para implementar un sumador completo.

Un decodificador es un circuito combinacional que convierte información binaria de  $n$  líneas de entrada a un máximo de  $2^n$  líneas de salidas distintas. Si la información codificada en  $n$  bits tiene combinaciones que no se usan, el decodificador podría tener menos de  $2^n$  salidas.

# Desarrollo

Para el desarrollo de la práctica se tuvo que regresar a estudiar la definición de los conceptos previamente expuestos, ya que no es posible poder realizar dichos circuitos si no conocemos cómo funcionan. Así mismo en esta parte repasamos la sintaxis básica del lenguaje VHDL para recordar cómo era el cuerpo mínimo que debe de llevar el programa para cada uno de los circuitos.

Una vez conociendo el funcionamiento de dichos circuitos, se procedió a realizar el análisis de cada uno de ellos, los cuales se exponen a continuación.

## Compuerta AND

Como se explicó al inicio, la compuerta AND trabaja sobre dos entradas de un bit cada una, la tabla de verdad de la compuerta se presenta nuevamente a continuación.

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1.4

El símbolo gráfico de esta compuerta es el siguiente.



Figura 1.2 Símbolo gráfico  
Compuerta AND

Para describir el funcionamiento de la compuerta, se observa que en el único caso de que las dos entradas se encuentren en 1's la salida de la compuerta será un 1 lógico, en cualquier otro caso la salida producirá un 0 lógico, al darnos cuenta de esto, se propuso utilizar un condicional que describa lo anterior en el lenguaje VHDL.

El diagrama a bloques de este circuito es el siguiente.

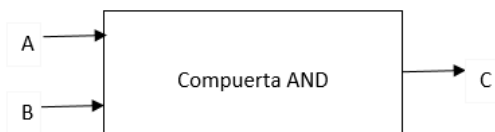


Figura 1.3 Diagrama de bloques de  
compuerta AND

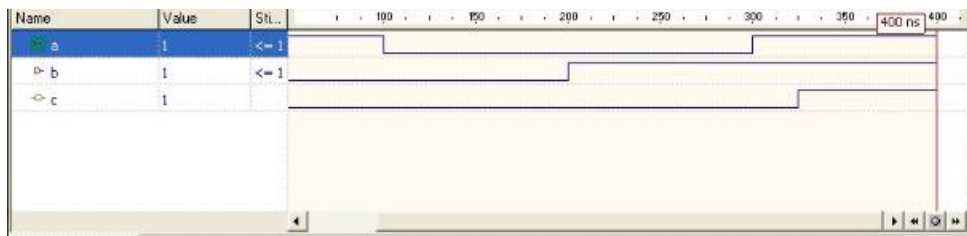


Figura 1.4 Simulación compuerta AND

### Sumador de 2 bits de entrada

La siguiente tabla de verdad, representa un sumador completo, es decir, con un acarreo inicial como ya se explicó al inicio de esta práctica.

A	B	C	Suma	Cf
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 1.5

Aplicando la reducción de mini términos a través del mapa de Karnaugh, se observa lo siguiente.

$$\text{Suma} = \sum (1, 2, 4, 7)$$

$$\text{Cf} = \sum (3, 5, 6, 7)$$

		BA	
		00	01
C	0	1	1
	1	1	

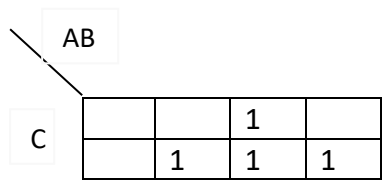
$$\text{Suma} = A'B'C + A'B'C' + ABC + A'BC'$$

$$\text{Suma} = A'(B'C + BC') + A(B'C' + BC)$$

$$\text{Suma} = A'(B \oplus C) + A(B \odot C)$$

$$\text{Suma} = A'(B \oplus C) + A(B \oplus C)'$$

$$Suma = A \oplus B \oplus C$$



$$C_f = AB + AC + BC$$

La elaboración del sumador hasta el momento solo se ha realizado para un solo bit, pero como ya se explicó anteriormente, el sumador se puede hacer con n bits, para este caso el sumador será de 2 bits, por lo tanto el acarreo inicial ( C ) será de 3 bits, pudiendo representar lo anterior en el siguiente diagrama a bloques.

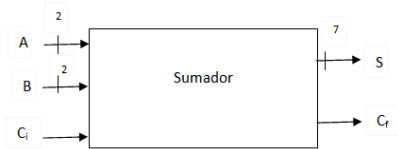


Figura 1.5 Diagrama de bloques de Sumador de 2 bits

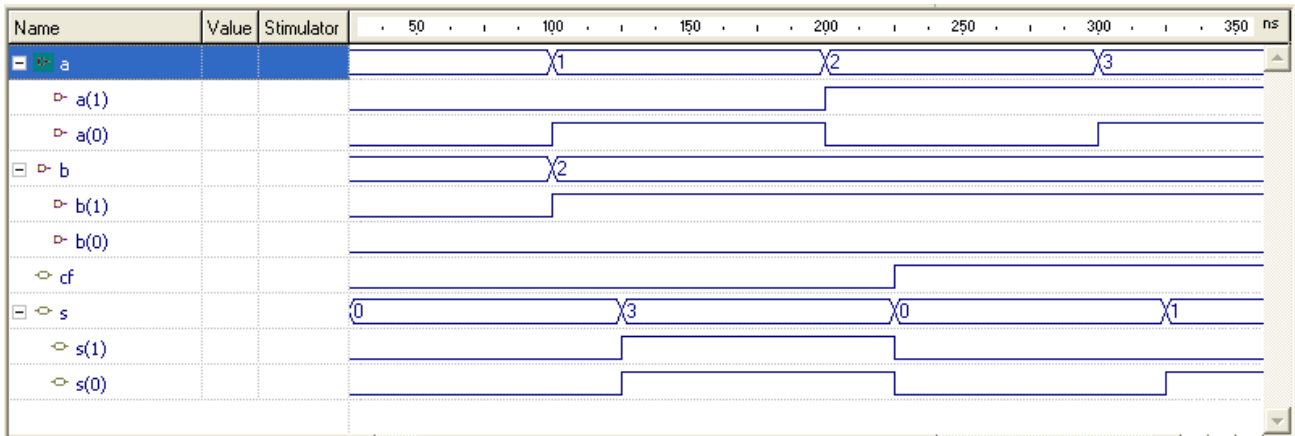


Figura 1.6 a) Inicio de simulación de sumador

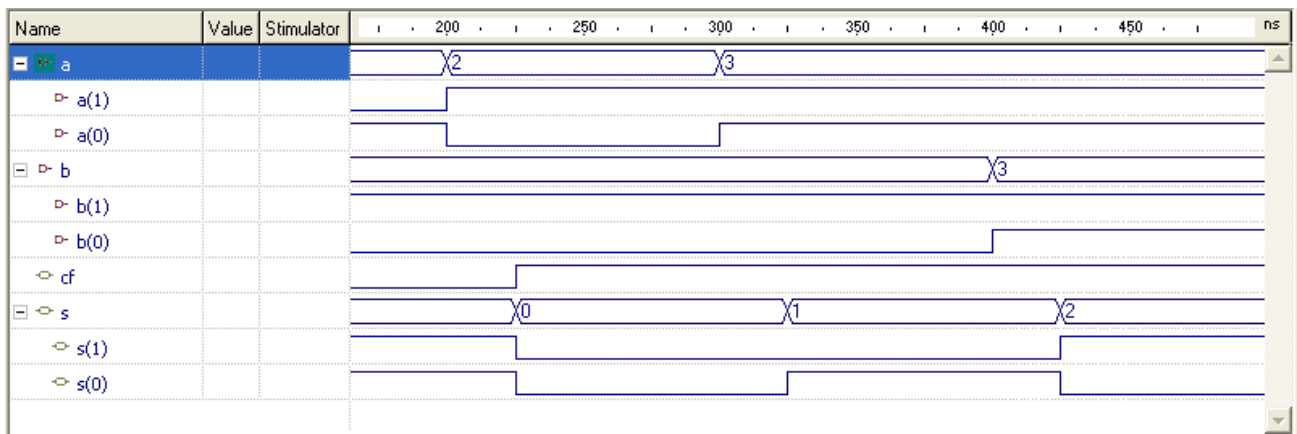


Figura 1.6 b) Continuación de simulación de sumador

### Identificador de números pares e impares de 3 bits

Para el desarrollo de este circuito, se empezó a través del diseño de la tabla de verdad del sistema, para ello se propuso una entrada de 3 bits con una salida de 1 en este caso solo para identificar en qué casos las salidas serán 1's y 0's lógicos.

E <sub>2</sub>	E <sub>1</sub>	E <sub>0</sub>	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Tabla 1.6

Al aplicar la reducción de los mini términos a través del mapa de Karnaugh, se tiene que la salida (S) es:

$$S = \sum (1,3,5,7)$$

	E <sub>2</sub> E <sub>1</sub>			
E <sub>0</sub>				
	1	1	1	1

$$S = E_0.$$

De esta reducción se justifica por qué en el diagrama presentado en el laboratorio solo se tomó en consideración el bit menos significativo para el programa.



Así mismo, a partir de la reducción y de la tabla 1.5, el equipo decidió mostrar la salida del circuito en un display de 7 segmentos ánodo común, recordando que la salida de este se enciende con 0's lógicos, por lo tanto, la tabla que representa las entradas y salidas de nuestro circuito es la siguiente:

E <sub>2</sub>	E <sub>1</sub>	E <sub>0</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	1	1	1	1
0	1	0	0	0	1	1	0	0	0
0	1	1	1	0	0	1	1	1	1
1	0	0	0	0	1	1	0	0	0
1	0	1	1	0	0	1	1	1	1
1	1	0	0	0	1	1	0	0	0
1	1	1	1	0	0	1	1	1	1

Tabla 1.7

Finalmente, la representación del diagrama a bloques para este circuito sería el que se presenta a continuación.



Figura 1.7 Diagrama de bloques de Par/impar de 3 bits

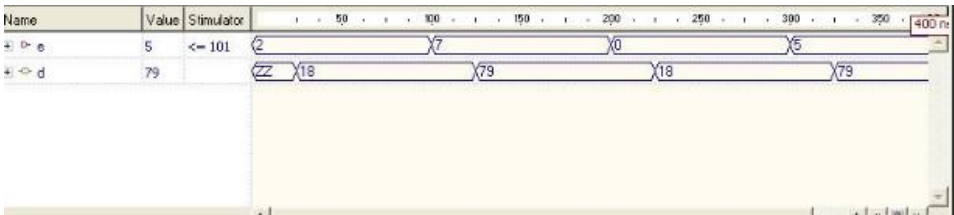


Figura 1.8 Simulación de Detector Par/Impar

## Decodificador de 4 bits.

Previamente se mencionó que un decodificador es un circuito combinacional de  $n$  entradas a un máximo de  $2^n$  posibles salidas. Para este circuito nuestro número de entradas fue de 4, pero con un máximo de salidas de 10, es decir que no se ocuparon todas las posibles salidas que podríamos haber tenido.

El equipo decidió hacer caso omiso a las demás posibles salidas, no se optó por tener un estado de no importa o algún equivalente en el que produjera una determinada salida especificada por nosotros mismos.

Por consiguiente, nuestra tabla de verdad para este circuito es la siguiente, no sin antes aclarar que la salida para este circuito nuevamente será un display de 7 segmento ánodo común.

E <sub>3</sub>	E <sub>2</sub>	E <sub>1</sub>	E <sub>0</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0

Tabla 1.8

El diagrama a bloques para este circuito es:

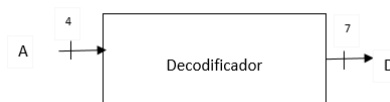


Figura 1.9 Diagrama de bloques de Decodificador de 4 bits

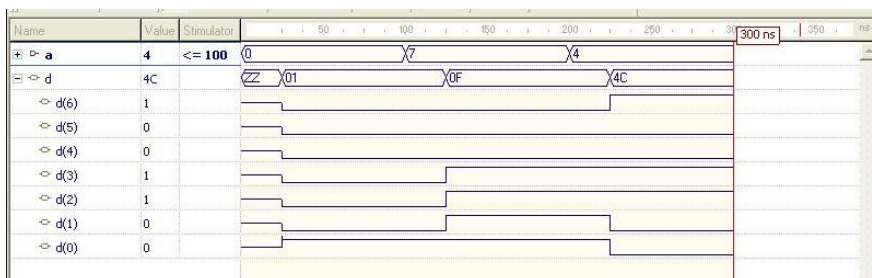


Figura 1.10 Simulación de Decodificador 4 bits

## Conclusiones

Como conclusión de esta práctica, consideramos que más haya de haberla terminado de manera exitosa, se cumplió el objetivo por el cual se nos dejó, el cual era poder volver a recordar conocimientos que previamente hemos adquirido, por ejemplo las compuertas lógicas y su representación algebraica, utilizar los mapas de karnaugh para que a partir de las salidas que requerimos poder encontrar una expresión algebraica que nos lleve a la solución correcta.

Así mismo encontrar su representación por medio de un diagrama de bloques, y a partir del diseño poder pasarlo al lenguaje VHDL, recordando también la sintaxis básica de este lenguaje, para que una vez terminado el programa poder llevarlo a la representación física por medio de la GAL22V10 y los respectivos componentes de cada circuito.

## Bibliografía

- M. Morris Mano. (2003). Diseño Digital. México: Pearson.