PRÁCTICO 2 - Lógica Combinacional

Minitérminos y maxitérminos para tres variables binarias

			Minitérminos		Maxitérminos	
x	y	z	Términos	Designación	Términos	Designación
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Mapas de Karnaugh de 2, 3 y 4 variables:

m_0	m_1
m_2	m_3

a)

$x \backslash y$	0	1 (y)
0	$x'y'$	$x'y$
1 (x)	xy'	xy

b)

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

a)

$x \backslash yz$	00	01 (z)	11 (y, z)	10 (y)
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1 (x)	$xy'z'$	$xy'z$	xyz	xyz'

b)

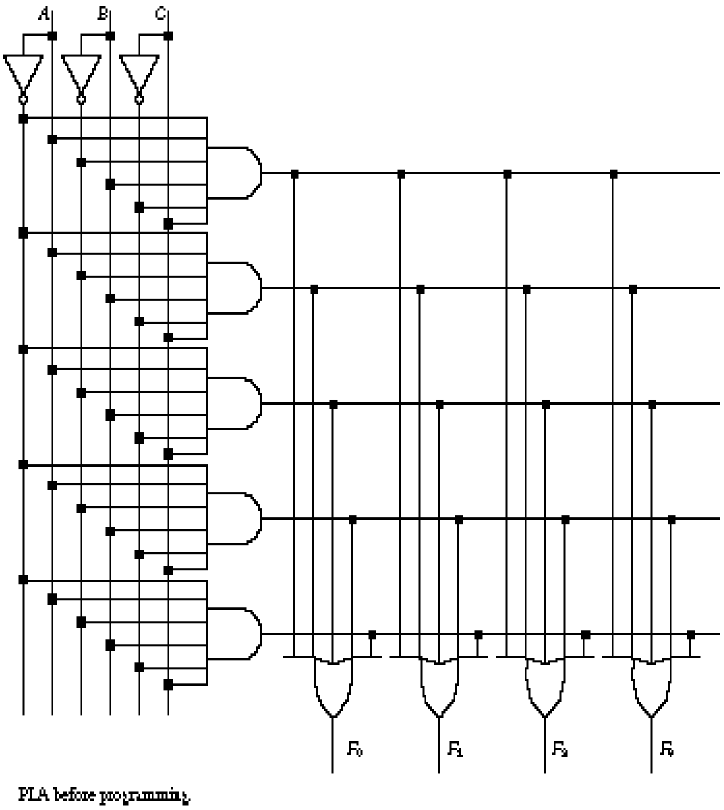
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

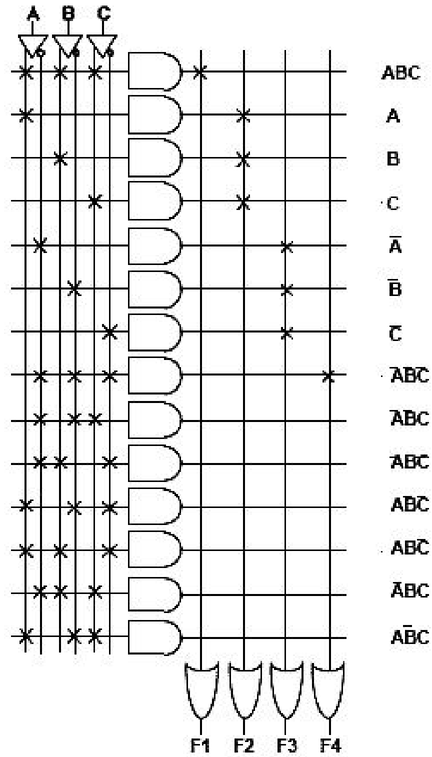
a)

$wx \backslash yz$	00	01 (z)	11 (y, z)	10 (y)
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01 (x)	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11 (w, x)	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10 (w)	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

b)

Programmable Logic Array (PLA):





Ejercicio 1:
Un detector de paridad impar de 4 entradas y una salida funciona de la siguiente manera: si la cantidad de entradas con valor ‘1’ es impar la salida se pone en ‘1’, en el resto de los casos la salida toma valor ‘0’.

a. Construir la tabla de verdad para dicho sistema.
b. Obtener la ecuación lógica como suma de minitérminos y producto de maxitérminos (funciones canónicas).
c. Implementar el sistema con compuertas NAND de la cantidad de entradas requeridas.
d. Implementar el sistema con una PLA.

Ejercicio 2:
Un sistema digital recibe información en forma de palabras de 5 bits (**ABCDE**) en un código protegido contra errores, de tal forma que cualquier dato que se reciba debe contener 3 y sólo 3 bits en ‘1’. Diseñar un circuito con las entradas **ABCDE** y una salida **err** que *se activa por bajo* cuando se recibe un dato incorrecto.

a. Construir la tabla de verdad para dicho sistema.
b. Obtener la ecuación lógica como suma de minitérminos y producto de maxitérminos (funciones canónicas).
c. Implementar el sistema con una PLA.

Ejercicio 3:
Verificar los resultados obtenidos de cada función lógica en la Guía 1 - Ejercicio 1, mediante la utilización de mapas de Karnaugh, el cual garantiza la obtención de la mínima expresión.

a. *x.y + x.y’*
b. *(x + y).(x + y’)*
c. *x.y.z + x’.y + xyz’*
d. *z.x + z.x’.y*
e. *(A + B)’.(A’ + B’)’*
f. *y.(w.z’ + w.z) + x.y*

Ejercicio 4:
Dadas la siguientes tablas de verdad para las funciones Fx:

(F1)

x3	x2	x1	x0	F(x3,x2,x1,x0)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(F2)

x3	x2	x1	x0	F(x3,x2,x1,x0)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(F3)

x2	x1	x0	F(x2,x1,x0)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

a. Encontrar las expresiones canónicas de cada Fx como suma de minitérminos y como producto de maxitérminos.
b. Encontrar la expresión minimizada de cada Fx utilizando mapas de Karnaugh.

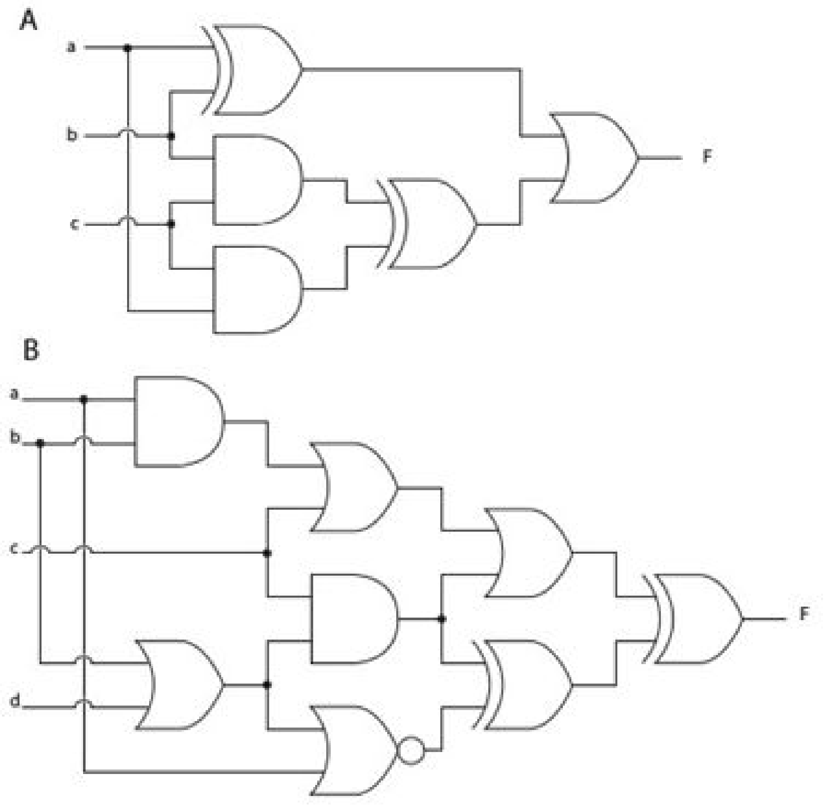
Ejercicio 5:

Un circuito combinacional comparador toma dos números de 2 bits, **A** = (A_1, A_0) y **B** = (B_1, B_0) y retorna tres salidas (“**A>B**”, “**A=B**” y “**A<B**”) de 1 bit cada una.
Ej: si A = (00) y B = (10), entonces “A>B” = ‘0’, “A=B” = ‘0’ y “A<B” = ‘1’.

a. Construir la tabla de verdad para dicho sistema.
b. Obtener la ecuación lógica como suma de minitérminos y producto de maxitérminos.
c. Encontrar la función minimizada de cada salida como suma de productos usando mapas de Karnaugh.
d. Implementar el sistema con compuertas lógicas básicas.

Ejercicio 6:

Analizar los circuitos de lógica combinacional de la figura. Para cada uno:



a. Escribir la función booleana correspondiente.
b. Encontrar la tabla de verdad para la función obtenida.
c. Obtener la función minimizada como suma de productos a partir el mapa de Karnaugh.
d. Dibujar el circuito de lógica combinacional resultante del punto (c).

Ejercicio 7:

Un DECODIFICADOR es un circuito combinacional que convierte información binaria de ‘N’ entradas codificadas (**A**), a ‘2^N’ salidas <u>únicas</u> (**X**). Esto quiere decir que sólo una salida **X** está activa y representa el valor de las señales de entrada **A**.
Considere un Decodificador activo por bajo (salida activa = ‘0’) con N=2 y 2^N=4 (deco 2 x 4).

a. Expresar las tablas de verdad de las cuatro salidas X_0, X_1, X_2 y X_3.
b. Encontrar las expresiones de X_0, X_1, X_2 y X_3 como suma de minitérminos y como producto de maxitérminos.
c. Encontrar expresiones minimizadas de X_0, X_1, X_2 y X_3 utilizando el método de Karnaugh o un método algebraico.
d. Implementar las expresiones anteriores a través del uso de compuertas lógicas.
e. Repetir el punto (d) agregando una entrada de HABILITACIÓN (**E**) activa por bajo, de tal forma que cuando **E**='1' ninguna señal de salida permanezca habilitada.

Ejercicio 8:
Implementar un decodificador de 3 x 8 y otro de 4 x 16 a partir de decodificadores 2 x 4 activos por bajo, con entrada de habilitación (**E**) activa por bajo y compuertas lógicas.

Ejercicio 9:
a. Diseñar un circuito SUMADOR COMPLETO (3 entradas: **X**, **Y**, $\mathbf{C_{IN}}$; 2 salidas: **S**, $\mathbf{C_{OUT}}$) mediante el uso de un Decodificador de salida activa por alto y compuertas OR. Tip: La salida que vale 1 representa el minitérmino equivalente al número binario que está a la entrada.
b. Diseñar un sumador completo usando dos semisumadores y una compuerta.

Ejercicio 10:
Considerando que un CODIFICADOR es un circuito combinacional que realiza la operación inversa de un decodificador:
a. Expresar la tabla de verdad de un codificador de octal a binario (8 entradas x 3 salidas).
b. Mostrar su implementación con compuertas OR a partir de la expresión del Codificador como suma de minitérminos.

Ejercicio 11:
Un MULTIPLEXOR (MUX) es un circuito combinacional que que selecciona información binaria de muchas entradas y la dirige a una única salida (**Y**), conforme al estado de las señales de selección. Si un MUX posee '2^N' entradas de información (**D**) requiere 'N' señales de selección (**S**).
a. Expresar la tabla de verdad de un MUX de 2 entradas (y una salida) y su implementación mediante el uso de compuertas lógicas (AND, OR, NOT, NOR, NAND, etc.)
b. Mostrar cómo se puede usar un MUX para obtener una compuerta NOT.
c. ¿Cómo obtener un MUX de 4 entradas (y una salida) en base a multiplexores de 2 entradas?
d. ¿Cómo obtener un multiplexor de 'N' entradas con multiplexores de 2 entradas?