Un sistema de numeración está formado por un conjunto de símbolos y reglas de generación que permiten construir todos los números válidos.

SISTEMA DE NUMERACIÓN
- No posicional (No ponderado) → Sistema Romano $\overline{LX} = 60$ $\overline{XL} = 40$
- Posicional (Ponderado) → Sistema Decimal

- Sistema Base 2 (Binario): {0, 1}
- Sistema Base 8 (Octal): {0, 1, 2, 3, 4, 5, 6, 7}
- Sistema Base 10 (Decimal): {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Sistema Base 16 (Hexadecimal): {0, 1, 2, 3, 4, ..., 8, 9, A, B, C, D, E, F}
  A=10, B=11, C=12, D=13, E=14, F=15

* Es necesario expresar el # del cual es la base M Ej: 1011 ; 1011_2 ó 1011_{16}
* En decimal no es necesario

Conversiones entre sistemas de numeración:

a) Conversión de sistema base 10 a base 2. Ej. $87 \rightarrow \underline{\ ?\ }_2$

Divisiones sucesivas:

%2	
87	1
43	1
21	1
10	0
5	1
2	0
1	1

$87_{10} \rightarrow 1010111_2$

b) Conversión de Base 2 a Base 10

Método de suma ponderada: *Base M a Base 10*

Ej. 1: $\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}_2 \Rightarrow (2^0) + (2^2) + (2^3) + (2^5) = 1 + 4 + 8 + 32 = \underline{45}$

($2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$)

Ej. 2: $1327_8 \Rightarrow 1(8^3) + 3(8^2) + 2(8^1) + 7(8^0)$
$= 512 + 192 + 16 + 7 = \underline{727}$

($8^3\ 8^2\ 8^1\ 8^0$)

Ej. 3: $CAFE8_{16} \Rightarrow 12(65536) + 10(4096) + 15(256) + 14(16) + 8$
$786432 + 40960 + 3840 + 224 + 8 = \underline{831,464}$

($16^4\ 16^3\ 16^2\ 16^1\ 16^0$)

- BASE 10 → BASE 2 / BASE N → Divisiones sucesivas
- BASE 2 (BASE N) → Base 10 → Suma ponderada.

Convertir $0.587_{10} \rightarrow \underline{\ ?\ }_2$

$0.587 \times 2 = 1.174$ 0.1001 // Se deja en 4 ó 5 dígitos.
$0.174 \times 2 = 0.348$
$0.348 \times 2 = 0.696$
$0.696 \times 2 = 1.392$

$0.792_{10} \longrightarrow \frac{?}{8}$

$0.792 \times 8 = 6.336$
$0.336 \times 8 = 2.688$
$0.688 \times 8 = 5.504$
$0.504 \times 8 = 4.032$

$\rightarrow 0.6254$

$728.792_{10} \longrightarrow \_\_\_\_\_\__{8}$

Entero Fracc.
Div Suc. Multip.

Números Binarios con Signo.

Para representar un número con signo en el sistema binario, existen 3 sistemas:

Sistema Signo-Magnitud.

En este sistema, el bit mas significativo (MSB) → está más a la izq. se utiliza, para denotar el signo del número, así que un "0" en esta posición indica que el número es POSITIVO, mientras que un "1" indica que el número es NEGATIVO.

El resto de los N-1 bits se usa para representar la cantidad.

N Bits.

MSB (signo)	Magnitud o Cantidad

- +10 ≡ con signo
- 10 ≡ sin signo

Obtener la representación en sistema signo-magnitud de:

a) $+87_{10}$

① Conversión.

$\frac{1}{64}\ \frac{0}{32}\ \frac{1}{16}\ \frac{0}{8}\ \frac{1}{4}\ \frac{1}{2}\ \frac{1}{1}$

② Agregar el MSB

$\underline{0}1010111_2$

b) -103_{10}

$103_{10} \rightarrow 1100111$

$-103 \rightarrow 11100111$

c) -128

$128_{10} \rightarrow 1\ 000\ 0000$

$-128_{10} \rightarrow 1\ 10000000$

*No se puede por sobrepasar el limite de 8 bits.

//Se debe extender a 16 bits de la sig. manera

$-128_{10} \rightarrow \boxed{1}0000000\ 10000000$

MSB — 16 bits.

Sistema Complemento a 1.

- Para representar un número positivo se escribe en formato binario natural de N bits. (signo-magnitud)
- Para representar un número negativo, se obtiene el complemento a 1 de la representación del mismo número positivo.
- El complemento a 1 de un número binario se obtiene al intercambiar "0's" por "1's" y al revés.

Ej. Obtener la representación en sistema complemento a 1 de:

a) $+117_{10}$

① Conversión a binario

$117 = 1110101_2$

② Agregar "0's" para completar los N bits.

$+117 = 01110101$ ✱

b) -98

① Conversión Binario

$98 = 1100010$

② Obtener $+98$:

01100010

③ C1's:

$-98 = 10011101$ ✱

c) $+235_{10}$

$235 = 11101011_2$

$+235 = 0000000011101011$

d) -482_{10}

$482 = 111100010$

$+482 = 0000000111100010$

$-482 = 1111111000011101$

Sistema complemento a 2.

Para representar un número positivo, se escribe en formato binario natural de N bits.

Para representar un número negativo, se obtiene el complemento a 2 del mismo número positivo.

El complemento a 2 de un número binario se obtiene: $\boxed{C2's = C1's + 1}$

Ej: Obtener la representación en sistema complemento a 2 de:

a) $+54_{10}$

$54_{10} \rightarrow 110110$

$+54_{10} \rightarrow 00110110$

b) -35_{10}

$35_{10} \rightarrow 100011$

$+35_{10} \rightarrow 00100011$

$-35_{10} \rightarrow 11011100$

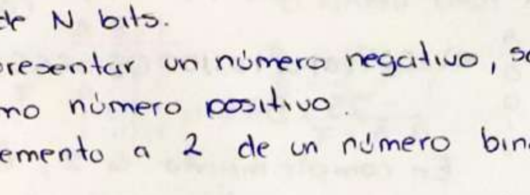
$+ \quad 1$

11011101

$\Rightarrow -35_{10} = 11011101$ ✱

✱ Valor decimal de un número con signo.

a) Sistema signo-magnitud



Realizar la conversión BIN/DEC de los n-1 bits que conforman la magnitud del número. El signo se toma del MSB.

E1.: Obtener el valor decimal de los sig. números con signo:

a) $\boxed{1}$ 0110110$_2$ <<signo - mag>>
↳ negativo → -54

b) Sistema complemento a 1:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Se realiza la conversión BIN/DEC del número con la ponderación descrita y:

1) Si el resultado es POSITIVO, ese es el valor decimal del # con signo
2) Si el resultado se NEGATIVO, a ese resultado se le somara un "1" para obtener el valor decimal del número con signo.

c) Sistema complemento a 2:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Se realiza la conversión BIN/DEC del número con la ponderación descrita y ese será el valor decimal del número con signo.

b) 00100111$_2$ <<C1's>>
+39

c) 11101110$_2$ <<C1's>>
-18 + 1 = -17

d) 01101111$_2$ <<C2's>>
+111

e) 1001101011011000$_2$ <<C2's>>
-25,896
En complemento a 2, el # de la conversión será el resultado.

* Obtener el complemento a 1 y el complemento a 2 de:

a) 01101101$_2$ (C1's) → 10010010

b) 11110110$_2$ (C2's) → 00001001$$0
$$\begin{array}{r} 000010010 \\ 1 \\ \hline 000010011 \end{array}$$

c) 101101000$_2$ (C2's) → 010011000

ALGEBRA DE BOOLE Y COMPUERTAS

Electrónica Digital
- Combinatoria «combinacional» : Ctos s/memoria
- Secuencial : Cto c/memoria.

Niveles lógicos:

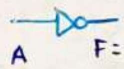
Nivel lógico	Voltaje (IDEAL)	Voltaje (REAL)
0	0v	0-0.8v
1	5v	2-5.5v

Compuertas lógicas:

1. Buffer

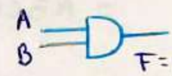
A	F
0	0
1	1

2. Inversor NOT



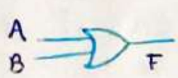
A	F
0	1
1	0

3. AND (Y)



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

4. OR (O)



$F = A + B$

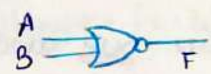
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

5. NAND (Not-And)

$F = \overline{A \cdot B}$

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

6. NOR (NOT-OR)



$F = \overline{A + B}$

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

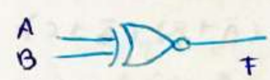
7. XOR (OR-Exclusive)

$F = A\bar{B} + \bar{A}B$

$F = A \oplus B$

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

8. XNOR (NOR-Exclusive)



$F = AB + \bar{A}\bar{B}$

$F = \overline{A \oplus B}$

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

CONVERSIÓN DE FORMA SUP A SUP canónica *

1. Multiplicar cada termino producto no canónico por un término formado por la suma de la variable que falta y su complemento. $(A + \bar{A}) = 1$.

2. Repetir el paso anterior hasta que todos los términos sean canónicos.

CONVERSIÓN DE FORMA POS A POS CANÓNICA *

1. Sumar a cada término suma no canónico un término formado por el producto de la variable que falta y su complemento $(A \cdot \bar{A} = 0)$ * Aplicar la regla $(A+B)(A+C) = A+BC$

2. Repetir el paso anterior hasta que todos los términos sean canónicos.

Ej. Aplicando algebra de Boole, obtenga la función canónica SUP de la siguiente función lógica.

$$* F(A,B,C,D) = \bar{A}B\bar{D} + BCD + A\bar{D} + \underline{\bar{A}B\bar{C}D}$$

$$\bar{A}B\bar{D} = \bar{A}B\bar{D}(C+\bar{C}) = \bar{A}B\bar{D}C + \bar{A}B\bar{D}\bar{C}$$

$$BCD = BCD(A+\bar{A}) = ABCD + \bar{A}BCD$$

$$A\bar{D} = A\bar{D}(B+\bar{B}) = AB\bar{D} + A\bar{B}D = AB\bar{D}(C+\bar{C}) + A\bar{B}D(C+\bar{C})$$

$$= ABC\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D$$

$$F(A,B,C,D) = \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D} + ABCD + \bar{A}BCD$$
$$+ ABC\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D$$

Función SUP canónica $\longleftrightarrow$ miniterminos $F = \sum_m (\ )$

Función POS canónica $\longleftrightarrow$ MAXITERMINOS $F = \Pi_M(\ )$

---

* Obtener maxiterminos:

$$F(A,B,C) = (\bar{A}+B)(\bar{B}+C) \qquad //\text{Aplicando } (x+y)(x+z) = x+yz$$

$$= \underbrace{(\bar{A}+B)}_{x} + \underbrace{C}_{y} \cdot \underbrace{\bar{C}}_{z}$$

$$(\bar{A}+B) + C\cdot\bar{C} = (\bar{A}+B+C)(\bar{A}+B+\bar{C})$$

$$F_2 = (A,B,C) = (\bar{B}+C) + A\bar{A}$$

$$(\bar{B}+C) + A\bar{A} = (A+\bar{B}+C)(\bar{A}+\bar{B}+C)$$

$$F(A,B,C) = \underbrace{(\bar{A}+B+C)}_{4}\underbrace{(\bar{A}+B+\bar{C})}_{5}\underbrace{(A+\bar{B}+C)}_{2}\underbrace{(\bar{A}+\bar{B}+C)}_{6}$$

$$F(A,B,C) = \Pi(2,4,5,6)$$ - pos → pos canónica.

ALGEBRA DE BOOLE

1. $x + 0 = x$ $\qquad$ $x \cdot 1 = x$

$x + \bar{x} = 1$ $\qquad$ $x \cdot \bar{x} = 0$

$x + x = x$ $\qquad$ $x \cdot x = x$

$x + 1 = 1$ $\qquad$ $x \cdot 0 = 0$

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

$x + y = y + x$ $\qquad$ $x \cdot y = y \cdot x$

$x + (y + z) = (x + y) + z$ $\qquad$ $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

$x(y + z) = xy + xz$ $\qquad$ $x + (yz) = (x + y)(x + z)$

$\overline{(x + y)} = \bar{x}\bar{y}$ // Morgan $\qquad$ $\overline{xy} = \bar{x} + \bar{y}$ // Morgan

$x + xy = x$ $\qquad$ $x \cdot (x + y) = x$

$x \cdot (\bar{x} + y) = xy$

$x + \bar{x}y = x + y$

$x \oplus y = \bar{x}y + x\bar{y}$

$\overline{x \oplus y} = xy + \bar{x}\bar{y}$

Ej. 1: Aplicando las reglas de algebra de boole, simplificar las siguientes expresiones lógicas:

1. $F_1 = AB + A(B + C) + B(B + C)$

$= AB + AB + AC + B + BC$

$= AB + AC + B$

$= B + AC$

2. $F_2 = [A\bar{B}(C + BD) + \bar{A}\bar{B}]C$

$= [A\bar{B}C + A\bar{B}B\overset{0}{D} + \bar{A}\bar{B}]C$

$= A\bar{B}C + \bar{A}\bar{B}C$

$= \bar{B}C(A + \bar{A})^{1}$

$= \bar{B}C$

3. $F_3 = \overline{AB + AC} + \bar{A}BC$

$\overline{AB} \cdot \overline{AC} + \bar{A}BC$

$(\bar{A} + \bar{B})(A + \bar{C}) + \bar{A}\bar{B}C$

$\bar{A} + \bar{B}\bar{C} + \bar{A}\bar{B}C$

$\bar{A} + \bar{B}\bar{C}$

4. $F_4 = (x + y)[xyz + (x + z)\bar{y}] + xy\bar{z}(x + \bar{x}y)$

$= (x + y)(xyz + \bar{y}x + \bar{y}z) + xy\bar{z}(x + y)$

$= (x + y)(xyz + \bar{y}x + \bar{y}z) + xy\bar{z} + xy\bar{z}$

$= xyz + x\bar{y} + x\bar{y}z + xyz + xy\bar{z}$

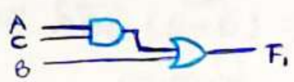
$= xz(y + \bar{y})^{1} + xy(z + \bar{z})^{1} + x\bar{y}$

$= xz + x(\bar{y} + y)^{1}$

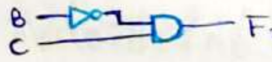
$= xz + x$

$= x$

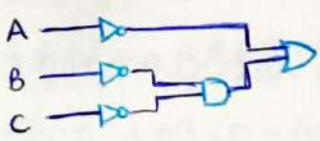
F_1



F_2



F_3



* Funciones Canónicas o Estándar *

- **Término producto:** Conjunto de literales (variables) relacionadas con la conectiva "AND"
- **Término suma:** Conjunto de literales (variables) relacionados con la conectiva "OR".
- **Término canónico o estándar:** Término producto o suma que contiene tantas literales como variables tiene la función. Ninguna literal aparece más de una vez.
- **Forma Suma de productos (SUP):** Función expresada como la suma de términos producto.
- **Forma Producto de sumas (POS):** Función expresada como el producto de términos de suma

* **Forma Canónica de una Función:** Aquella en la que todos sus términos son canónicos y se expresa en forma SUP o POS

> Una función canónica SUP es igual a "1" si y solo si, uno o más de los términos productos que forman la expresión es igual a 1.
Al término canónico producto que hace que la función sea igual a "1" se le llama "minitérmino" <<1>>
Variable no complementada = "1" ; Variable complementada = "0".

> Una función canónica POS es igual a "0", si y solo si, uno o más de los términos suma que forman la expresión es igual a "0".
Al término canónico suma que hace que la función sea igual a "0" se le llama "MAXITÉRMINO" <<0>>
Variable no complementada = "0" ; Variable complementada = "1".

Ej. 1:

$F_1(A,B,C) = \overbrace{\underbrace{AB}_{\to 0}}^{\text{término producto}} + \underbrace{\bar{A}BC}_{\to 1 \text{ minitérmino}} + \overbrace{A\bar{B}\bar{C}}^{\text{término producto canónico}}$ → Forma SUP, no canónica.

$F_2(A,B,C) = \overbrace{(A+\bar{B}+\bar{C})}^{\text{término suma}}(\bar{A}+\bar{B}+C)$ → Forma POS canónica

$F_3(A,B,C) = A\bar{B}C + (A+\bar{B}+C)(\bar{A}+B+\bar{C})$ → No es ninguno ∴ se tiene que pasar a una de esas

E.1 Aplicando Algebra de Boole, obtener los maxiterminos de la sig. Función lógica:

$F(A,B,C) = (\bar{A}+B)(\bar{B}+C)$ //Aplicamos $(x+y)(x+z) = x+yz$

$\underbrace{(\bar{A}+B)}_{x} + \underbrace{C}_{y}\cdot\underbrace{\bar{C}}_{z} = (\bar{A}+B+C)(\bar{A}+B+\bar{C})$

$(\bar{B}+C) + \bar{A}A = (\bar{A}+\bar{B}+C)(A+\bar{B}+C)$

$\Rightarrow F(A,B,C) = \underbrace{(\bar{A}+B+C)}_{\overset{100}{④}}\underbrace{(\bar{A}+B+\bar{C})}_{\overset{101}{⑤}}\underbrace{(\bar{A}+\bar{B}+C)}_{\overset{110}{⑥}}\underbrace{(A+\bar{B}+C)}_{\overset{010}{②}}$

$F(A,B,C) = \Pi(2,4,5,6)$ Pos → pos canónica

$F(A,B,C) = (\bar{A}+B)\cdot(\bar{A}+\overline{B+C})$

$= (\bar{A}+B+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B}\bar{C})$

$= (\bar{A}+B+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B})(\bar{A}+\bar{C})$

$= \underbrace{(\bar{A}+B+\bar{C})}_{\overset{101}{⑤}}\underbrace{(\bar{A}+B+C)}_{\overset{100}{④}}\underbrace{(\bar{A}+B+C)}_{\overset{110}{⑥}}\underbrace{(\bar{A}+\bar{B}+\bar{C})}_{\overset{111}{⑦}}$

$F(A,B,C) = \Pi(4,5,6,7)$ ⇒ Mini: $\Sigma(0-3)$

Maxi

Ej. Aplicando algebra de Boole, obtener los miniterminos de la sig. función lógica:

$F(A,B,C,D) = AB + B\bar{D} + \bar{C}$

$AB = AB(C+\bar{C}) = ABC(D+\bar{D}) + AB\bar{C}(D+\bar{D}) = \overset{1111}{ABCD} + \overset{1110}{ABC\bar{D}} + \overset{1101}{AB\bar{C}D}$ $B\bar{C}$

$+ \underset{1100}{AB\bar{C}\bar{D}} \rightarrow 15, 13, 12, 14$

$B\bar{D} = B\bar{D}(A+\bar{A}) = AB\bar{D}(C+\bar{C}) + \bar{A}B\bar{D}(C+\bar{C}) = ABC\bar{D} + AB\bar{C}\bar{D} + \underset{0110}{\overset{⑥}{\bar{A}BC\bar{D}}} + \underset{0100}{\overset{④}{\bar{A}B\bar{C}\bar{D}}}$

$\bar{C} = \bar{C}(A+\bar{A}) = A\bar{C}(B+\bar{B}) + \bar{A}\bar{C}(B+\bar{B}) = ABC(D+\bar{D}) + A\bar{B}\bar{C}(D+\bar{D})$

$+ \bar{A}\bar{C}B(D+\bar{D}) + \bar{A}\bar{C}\bar{B}(D+\bar{D})$

$= ABCD + ABC\bar{D} + \underset{1001}{\overset{⑨}{A\bar{B}\bar{C}D}} + \underset{1000}{\overset{⑧}{A\bar{B}\bar{C}\bar{D}}} + \underset{0101}{\overset{⑤}{\bar{A}B\bar{C}D}} + \underset{0100}{\bar{A}B\bar{C}\bar{D}} + \underset{0001}{\overset{①}{\bar{A}\bar{C}\bar{B}D}} + \underset{0000}{\overset{⓪}{\bar{A}\bar{C}\bar{B}\bar{D}}}$

$F(A,B,C,D) = \bar{A}\bar{C}\bar{B}\bar{D} + \bar{A}\bar{C}\bar{B}D + \bar{A}B\bar{C}D + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D}$

$+ AB\bar{C}\bar{D} + AB\bar{C}D + ABC\bar{D} + ABCD$

$F(A,B,C,D) = \Sigma(0,1,4,5,6,8,9,12,13,14,15)$.

★ OBTENCIÓN DE MINITÉRMINOS/MAXITERMINOS USANDO ARBOLES BINARIOS.

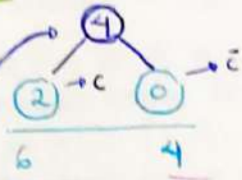
a) Minitérminos:

$F(A,B,C,D) = \bar{A}B\bar{D} + BCD + A\bar{D} + \bar{A}B\bar{C}D$

peso 8 4 2 1

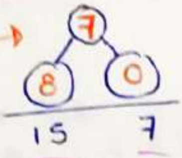
$\bar{A}B\bar{D} = (0)(8) + (4)(1) + (1)(0) = 4$

(falta C con peso 2)



$BCD = (4)(1) + (2)(1) + (1)(1) = 7$

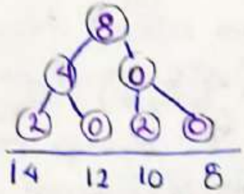
(falta A con peso 8)



$A\bar{D} = (8)(1) + (1)(0) = 8$

Falta B y C

con pesos 4 y 2 ⇒



$\bar{A}B\bar{C}D = (8)(0) + (4)(1) + (2)(0) + (1)(1) = 5$

5

$\therefore F(A,B,C,D) = \Sigma(4,5,6,7,8,10,12,14,15)$

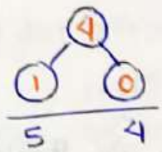
b) Maxitérminos:

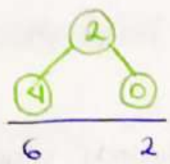
$F(A,B,C) = (\bar{A}+B)(\bar{B}+C)$

peso 4 2 1

$\bar{A}+B \rightarrow (4)(1) + (2)(0) = 4$

$\bar{B}+C \rightarrow (2)(1) + (1)(0) = 2$





$\therefore F(A,B,C) = \Pi(2,4,5,6)$.

- Diseñe un cto. lógico detector de números primos como se muestra en la figura.

① 

② Tabla de verdad.

	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

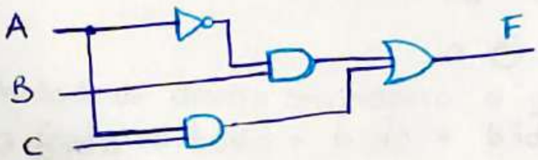
③ Función canónica "sop" (minitermínos) // ó "pos" dependiendo de las salidas de la tabla de vdd.

$\Rightarrow F(A,B,C) = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$

④ Reducción mediante álgebra de Boole.

$$F(A,B,C) = \bar{A}B\cancel{(\bar{C}+C)}^{1} + AC\cancel{(\bar{B}+B)}^{1}$$
$$= \underline{\bar{A}B + AC}$$

⑤ Diagrama con compuertas.



- Diseñar un cto. convertidor de código binario natural a codigo Gray (4 bits)

* Conversion Binario a Gray

Binario 1 + 0 + 1 + 0 + 1 + 1 + 0

Gray 1 1 1 1 1 0 1

Dec	Bin. Nat. a b c d	Gray F_1 F_2 F_3 F_4
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

* Conversion Gray a Binario

// No se consideran acarreos

Gray 1 1 1 1 1 0 1

Binario 1 0 1 0 1 1 0

$$F_1 = a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}c\bar{d} + a\bar{b}cd + ab\bar{c}\bar{d} + ab\bar{c}d + abc\bar{d} + abcd$$
$$= a\bar{b}\bar{c}(\bar{d}+d) + a\bar{b}c(\bar{d}+d) + ab\bar{c}(\bar{d}+d) + abc(\bar{d}+d)$$
$$= a\bar{b}(\bar{c}+c) + ab(\bar{c}+c)$$
$$= a(\bar{b}+b)$$
$$= \underline{a}$$

$$F_2 = \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}c\bar{d} + a\bar{b}cd$$
$$= \bar{a}b\bar{c}(\bar{d}+d) + \bar{a}bc(\bar{d}+d) + a\bar{b}\bar{c}(\bar{d}+d) + a\bar{b}c(\bar{d}+d)$$
$$= \bar{a}b(c+\bar{c}) + a\bar{b}(c+\bar{c})$$
$$= \bar{a}b + a\bar{b} = a \oplus b$$

$$F_3 = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + a\bar{b}c\bar{d} + a\bar{b}cd + ab\bar{c}\bar{d} + ab\bar{c}d$$
$$= \bar{a}\bar{b}c(\bar{d}+d) + \bar{a}b\bar{c}(\bar{d}+d) + a\bar{b}c(\bar{d}+d) + ab\bar{c}(\bar{d}+d)$$
$$= (\bar{a}+a)\bar{b}c + b\bar{c}(\bar{a}+a)$$
$$= \bar{b}c + b\bar{c}$$
$$= b \oplus c$$

$$F_4 = \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + a\bar{b}\bar{c}d + a\bar{b}c\bar{d} + ab\bar{c}d + abc\bar{d}$$
$$= \bar{a}\bar{c}d(\bar{b}+b) + \bar{a}c\bar{d}(\bar{b}+b) + a\bar{c}d(\bar{b}+b) + ac\bar{d}(b+\bar{b})$$
$$= c\bar{d}(\bar{a}+a) + \bar{c}d(\bar{a}+a)$$
$$= c\bar{d} + \bar{c}d$$
$$= c \oplus d$$

Software + Hardware

- VHDL
- PLD: Dispositivo Lógico Programable

SPLD: Simple PLD
1 spld :

CPLD: Complejo PLD

FPGA: Arreglo de compuertas programables en campo.

VHDL.

- Entity [nombre de la identidad] is port
  A, B, C : in bit;
  cout, suma : out bit;
  end [nombre de la unidad];

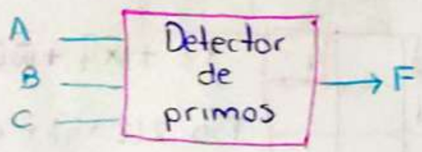
MSB

A : in bit_vector (3 down to 0) ⇒ A [A_3, A_2, A_1, A_0]
A: (0 to 3) ⇒ [A_0, A_1, A_2, A_3]

Use nombre-libreria.nombre-paquete.all; ⟷ llamar una biblio
ej: library ieee;
use ieee.std_logic_1164.all;

Arquitectura (architecture): Unidad de diseño secundario o que describe el comportamiento de una entidad.

$F(A,B,C) = \bar{A}B + AC.$



VHDL:

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

entity PRIMOS is port (
  A,B,C: in std_logic;
  F: out std_logic );
end PRIMOS;

architecture APRIMOS of PRIMOS is
begin
  F <= ((not A) and B) or (A and C);
end APRIMOS;
```

MAPAS DE KARNAUGH

"1's" Suma de productos (SOP)
"0's" Producto de sumas (POS)

// Se agrupan en pot. de 2^n

Mapa de dos variables

x \ y	0	1
0	$x'y'$	$x'y$
1	xy'	xy

1 → 2 var.
2 → 1 var
4 → 1

Ej.

x \ y	0	1
0		1
1	1	1

$xy' + xy = x(\cancel{y'+y})^{1} = x$

$x'y + xy = y(\cancel{x'+x})^{1} = y$

$\Rightarrow F = x + y$

Mapa de tres variables

x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

1 → 3 var
2 → 2 var
4 → 1 var
8 → 1

Ej.

x \ yz	00	01	11	10
0			1	
1	1		1	1

$x'yz + xyz = yz(x'+x) = yz$

$xy'z' + xyz' = xz'(y'+y) = xz'$

$\Rightarrow F = xz' + yz$

Mapa de 4 variables

wx \ yz	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_1	m_{10}

1 → 4 var.
2 → 3 var.
4 → 2 var
8 → 1 var
16 → 1

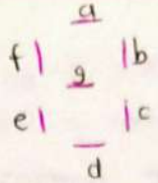
Ej.

wx \ yz	00	01	11	10
00	1	1		1
01		1		
11				
10	1	1		1

$\bar{x}\bar{z} + \bar{x}\bar{y} + \bar{w}\bar{y}z$

$F \Rightarrow \bar{x}(\bar{z} + \bar{y}) + \bar{w}\bar{y}z$

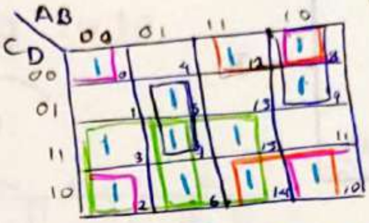
CODIGO HEXADECIMAL A 7 SEG.



	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
A 10	1	0	1	0	1	1	1	0	1	1	1
b 11	1	0	1	1	0	0	1	1	1	1	1
C 12	1	1	0	0	1	0	0	1	1	1	0
d 13	1	1	0	1	0	1	1	1	1	0	1
E 14	1	1	1	0	1	0	0	1	1	1	1
F 15	1	1	1	1	1	0	0	0	1	1	1

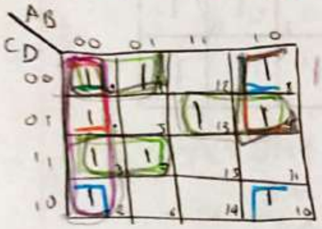
$1 \to 4$
$2 \to 3$
$4 \to 2$
$8 \to 1$

a) Para a)

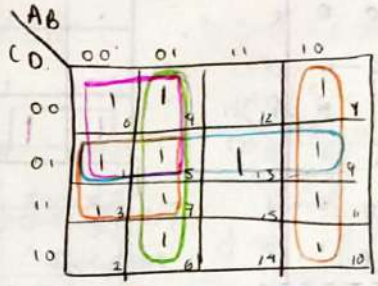


$a = \bar{B}\bar{D} + A\bar{D} + \bar{A}C + BC + \bar{A}BD + A\bar{B}\bar{C}$

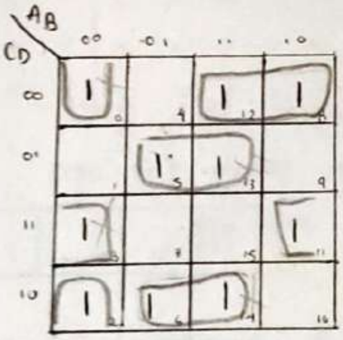
b) $b = \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{B}\bar{D} + \bar{A}\bar{C}\bar{D} + A\bar{C}D$



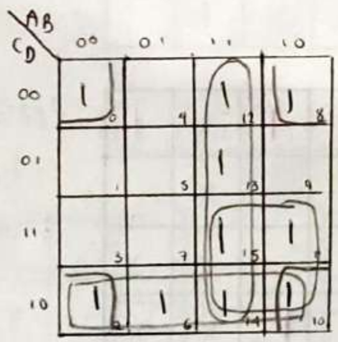
c)



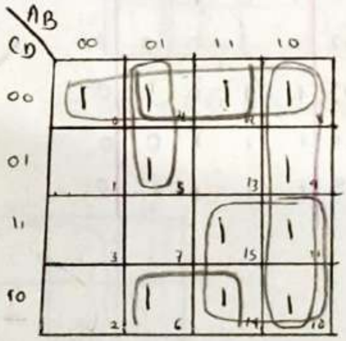
$c = \bar{C}D + A\bar{B} + \bar{A}B + \bar{A}\bar{C} + \bar{A}D$



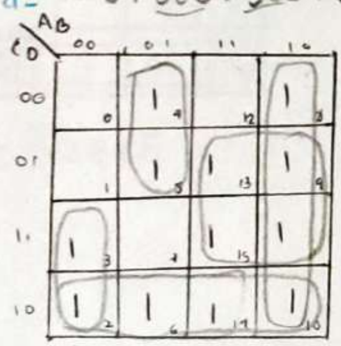
$d = A\bar{C}\bar{D} + B\bar{C}D + \bar{B}CD + BC\bar{D} + \bar{A}\bar{B}\bar{D}$



e) $C\bar{D} + AB + \bar{B}\bar{D} + AC$



f) $\bar{C}\bar{D} + A\bar{B} + AC + B\bar{D} + \bar{A}B\bar{C}$

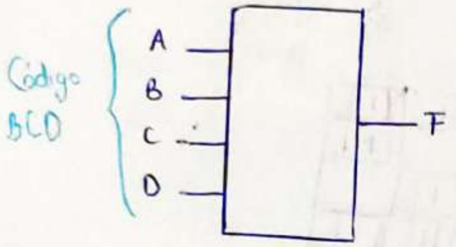


g) $A\bar{B} + C\bar{D} + AD + \bar{A}B\bar{D} + \bar{A}\bar{B}C$

* CONDICIONES IRRELEVANTES ("NO IMPORTA")

Ej. Diseñe un detector de números pares de un código BCD.

Nota: Incluya al "0" como número par.



BCD: Binary Code Decimal

DEC	BIN
0	0000
1	0001
2	0010
⋮	⋮
9	1001

	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

CD \ AB	00	01	11	10
00	1	1		1
01				
11				
10	1	1		

$F = \bar{A}\bar{D} + \bar{B}\bar{C}\bar{D}$

AND → 3

NOT → 4

OR → 1

CD \ AB	00	01	11	10
00	1	1	1	1
01			1	
11			1	1
10	1	1	1	1

$F = \bar{D} + AB + AC$

AND → 2

NOT → 1

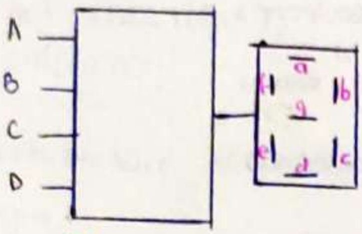
OR → 2

1	1	X	1
		X	
		X	X
1	1	X	X

// Solo ocupar las que beneficien para que sea un grupo mayor

$F = \bar{D}$

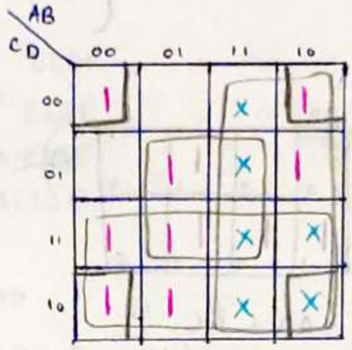
Ej. Diseñar un convertidor de código BCD a código de 7 segmentos



F1

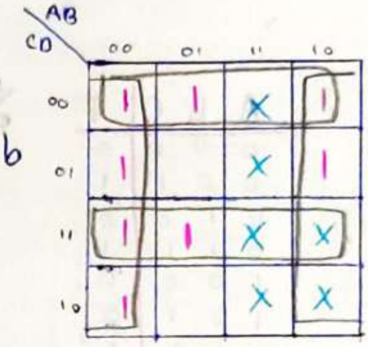
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x

a



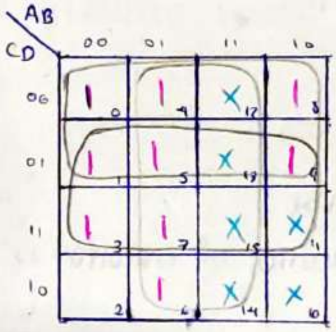
$a = C + A + BD + \bar{D}\bar{B}$

b



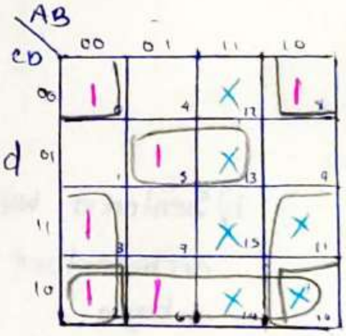
$b = \bar{B} + \bar{C}\bar{D} + CD$

c



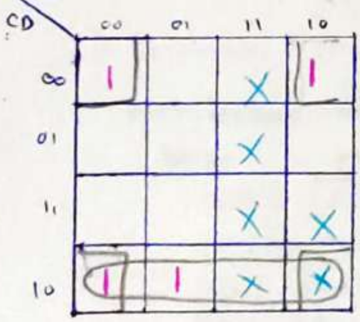
$c = \bar{C} + D + B$

d



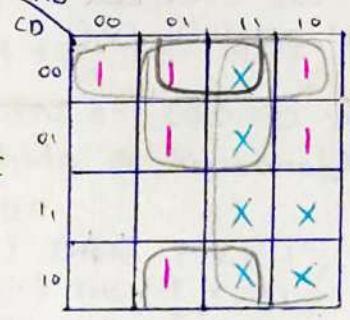
$d = \bar{B}\bar{D} + C\bar{D} + \bar{B}C + B\bar{C}D$

e



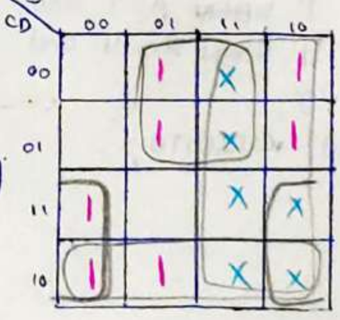
$e = \bar{B}\bar{D} + C\bar{D}$

f



$f = A + B\bar{C} + \bar{C}\bar{D} + B\bar{D}$

g



$g = A + C\bar{D} + \bar{B}C + B\bar{C}$

SENTENCIAS VHDL

Sentencias.
- Concurrentes
  - Ecuaciones booleanas a)
  - WHEN - ELSE b)
  - WITH - SELECT - WHEN
- Secuenciales
  - Ecuaciones booleanas
  - IF - ELSE
  - CASE

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

C \ AB	00	01	11	10
0	(0)	(2)	1 (6)	(4)
1	1 (1)	1 (3)	1 (7)	(5)

$F = AB + \bar{A}C$

VHDL

a)
```
entity CIRCUITO is port (
  A,B,C : in std-logic
     F : out std-logic);
end CIRCUITO;

architecture ACIRCUITO of CIRCUITO is
begin
  F<=(A and B) or ((not A) and C);
end ACIRCUITO;
```

b) Sentencia WHEN-ELSE

```
architecture ACIRCUITO of CIRCUITO is
begin
F<='1' WHEN A='0' and B='0' and C='1' ELSE
    '1' WHEN A='0' and B='1' and C='1' ELSE
    '1' WHEN A='1' and B='1' and C='0' ELSE
    '1' WHEN A='1' and B='1' and C='1' ELSE
    '0';
end CIRCUITO;
```

```
entity CIRCUITO is port (
  A: in std_logic_vector(2 downto Ø),
  F: out std_logic);
end CIRCUITO;

  architecture ACIRCUITO of CIRCUITO is
  begin
    F <= '1' WHEN A="001" ELSE
         '1' WHEN A="011" ELSE
         '1' WHEN A="110" ELSE
         '1' WHEN A="111" ELSE
         'Ø';

  end ACIRCUITO;
```

b) Sentencia WITH-SELECT-WHEN.

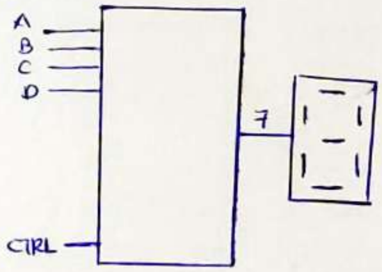
```
architecture ACIRCUITO of CIRCUITO is
begin
   WITH A SELECT
     F <= '1' WHEN "001",
          '1' WHEN "011",
          '1' WHEN "110",
          '1' WHEN "111",
          'Ø' WHEN OTHERS;

end ACIRCUITO;
```

c) Sentencia IF-ELSE (IF-THEN-ELSIF)

```
architecture ACIRCUITO of CIRCUITO is
  begin        //lista de las variables.
   process (A) begin
    IF (A="001") THEN F <= '1';
    ELSIF (A="011") THEN F <= '1';
    ELSIF (A="110") THEN F <= '1';
    ELSIF (A="111") THEN F <= '1';
    ELSE F <= '0';
   end IF
  end process;
end ACIRCUITO;
```

d) Sentencia CASE - WHEN

```
architecture ACIRCUITO of CIRCUITO is
begin
    process (A) begin
        case (A) is
        WHEN "001" => F <= '1';
        WHEN "011" => F <= '1';
        WHEN "110" => F <= '1';
        WHEN "111" => F <= '1';
        WHEN OTHER => F <= 'Φ';
        end case;
    end process;
end ACIRCUITO;
```



Si CTRL = Φ => Convertidor BIN / 7 seg

Si CTRL = 1 => $\underbrace{\text{Convertidor BCD / 7 seg}}_{\text{Ec. booleana}}$ / Códigos no válidos se visualiza

Ejemplo:

En un laboratorio químico se elaboran soluciones con los componentes w, x, y, z.
Estas soluciones pesan 800, 400, 200, y 100 mg respectivamente. Las soluciones son depositadas en frascos que se transportan por medio de una banda hasta una báscula. Si el peso indicado es uno de los siguientes: 200, 500, 700, 800, 1100, 1400 o 1500 mg entonces un dispositivo electromagnetico F sellará los frascos y los apartará de la banda; de otro modo, el frasco permanecerá abierto y seguirá en la banda a otra etapa del proceso.
Por las condiciones previas, no es posible que lleguen a la báscula ni frascos vacios, ni frascos que contengan las siguientes sustancias: WY, YZ, WX ó WZ; todas las demás combinaciones si pueden llegar hasta la báscula.
Use mapas de Karnaugh para determinar la función lógica para el control del dispositivo F.

	W	X	Y	Z	F
0	0	0	0	0	X
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	X
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	X
10	1	0	1	0	X
11	1	0	1	1	1
12	1	1	0	0	X
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

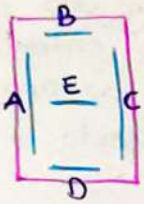
YZ \ WX	00	01	11	10
00	X (0)	(4)	X (12)	1 (8)
01	(1)	1 (5)	(13)	X (9)
11	X (3)	1 (7)	1 (15)	1 (11)
10	1 (2)	(6)	1 (14)	X (10)

$$F = \bar{X}\bar{Z} + WY + \bar{W}XZ$$

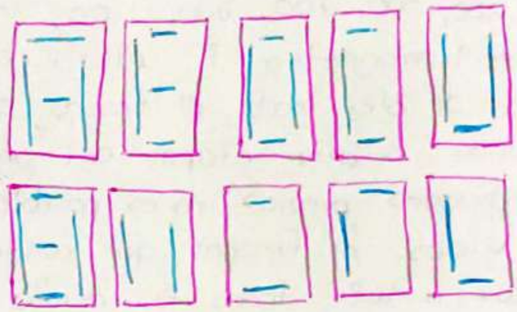
Ejemplo:

En una fábrica un dispositivo con 5 fotoceldas, registra un conjunto de caracteres abriendo pequeñas ranuras en una tarjeta de control.

Tarjeta de control

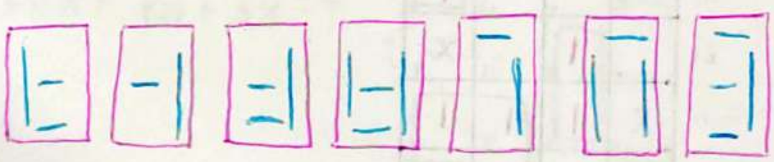


Si en la tarjeta registrada hay uno de los sig. símbolos:



Entonces el dispositivo envía una señal "1" que acciona un taladro.

En el proceso no hay tarjetas con menos de dos ranuras, y tampoco las siguientes:

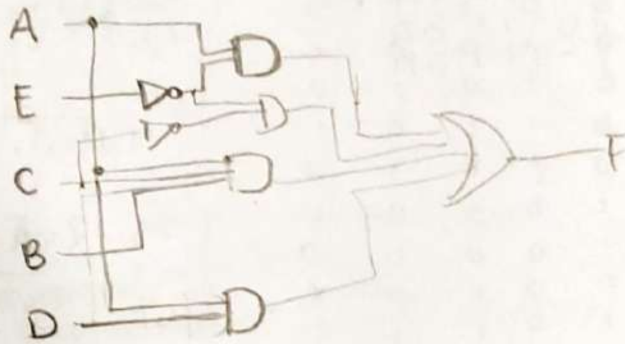


Aplique mapas K para obtener el circuito lógico que acciona el taladro.

	A	B	C	D	E	F
0	0	0	0	0	0	X
1	0	0	0	0	1	X
2	0	0	0	1	0	X
3	0	0	0	1	1	0
4	0	0	1	0	0	X
5	0	0	1	0	1	X
6	0	0	1	1	0	0
7	0	0	1	1	1	X
8	0	1	0	0	0	X
9	0	1	0	0	1	0
10	0	1	0	1	0	1
11	0	1	0	1	1	0
12	0	1	1	0	0	X
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	X
16	1	0	0	0	0	X
17	1	0	0	0	1	0
18	1	0	0	1	0	1
19	1	0	0	1	1	X
20	1	0	1	0	0	1
21	1	0	1	0	1	0
22	1	0	1	1	0	1
23	1	0	1	1	1	X
24	1	1	0	0	0	1
25	1	1	0	0	1	0
26	1	1	0	1	0	1
27	1	1	0	1	1	1
28	1	1	1	0	0	X
29	1	1	1	0	1	1
30	1	1	1	1	0	1
31	1	1	1	1	1	1

DE \ ABC	000	001	011	010	110	111	101	100
00	X	X	X	X	1	X	1	X
01	X	X				1		
11		X	X		1	1	X	X
10	X			1	1	1	1	1

$$F = A\bar{E} + AD + ABC + \bar{C}\bar{E}$$



Ejemplo:

Un circuito lógico tiene cinco entradas: A, B, C, D, E; siendo A la entrada más significativa y entrada de control; mientras que las otras entradas representan un dígito decimal en BCD.

Cuando la entrada de control es igual a '0', la salida Z será igual a '0' si el número decimal representado es impar; en caso contrario es '1'.

Cuando la entrada de control es igual a '1' y cuando el número representado sea múltiplo de 3 la salida Z será igual a '1'; en caso contrario será '0'.

Considere al número cero como un número par.

Use mapas K para obtener la función lógica del sistema.

	A	B	C	D	E	Z	
0	0	0	0	0	0	1	
1	0	0	0	0	1	0	
2	0	0	0	1	0	1	
3	0	0	0	1	1	0	
4	0	0	1	0	0	1	
5	0	0	1	0	1	0	
6	0	0	1	1	0	1	
7	0	0	1	1	1	0	
8	0	1	0	0	0	1	
9	0	1	0	0	1	0	
10	0	1	0	1	0	X	
11	0	1	0	1	1	X	
12	0	1	1	0	0	X	
13	0	1	1	0	1	X	
14	0	1	1		1	X	
15	0	1	1	1	0	X	
16	1	0	0	0	1	X	
17	1	0	0	0	0	1	0
18	1	0	0	0	1	0	1
19	1	0	0	1	0	0	2
20	1	0	1	1	1	1	3
				0	0	0	4
21	1	0	1	0	1	0	5
22	1	0	1	1	0	1	6
23	1	0	1	1	1	0	7
24	1	1	0	0	0	0	8
25	1	1	0	0	1	1	9
26	1	1	0	1	0	X	
27	1	1	0	1	1	X	
28	1	1	1	0	0	X	
29	1	1	1	0	1	X	
30	1	1	1	1	0	X	
31	1	1	1	1	1	X	

A'

DE \ BC	00	01	11	10
00	1	1	X_{12}	1
01	$_1$	$_5$	X_{13}	$_9$
11	$_3$	$_7$	X_{15}	X_{11}
10	1_2	1_6	X_{14}	X

A

DE \ BC	00	01	11	10
00	1	$_4$	X_{12}	$_8$
01	$_1$	$_5$	X_{13}	1
11	1	$_7$	X_{15}	X
10	$_2$	1_6	X_{14}	X_{10}

1 → 5
2 → 4
4 → 3
8 → 2
16 → 1

$$Z=\bar{A}\bar{E}+A\bar{C}DE+CD\bar{E}+\bar{B}\bar{C}\bar{D}\bar{E}+AB\bar{E}$$

1_0	1_4	X_{12}	1_8	$_{24}$	X_{28}	$_{20}$	1_{16}
$_1$	$_5$	X_{13}	$_9$	1_{25}	X_{29}	$_{21}$	$_{17}$
$_3$	$_7$	X_{15}	X_{11}	X_{27}	X_{31}	$_{23}$	1_{19}
1_2	1_6	X_{14}	X_{10}	X_{26}	X_{30}	1_{22}	$_{18}$

LUNES :)

Ej. Aplicando el método de Quine McCluskey, obtener la función simplif. de:

$F(A,B,C,D) = \sum(3,4,7,9,10) + \gamma(0\text{-}2, 13\text{-}15)$

→ Símbolo (puede ser otro) para los irrelevantes

	A	B	C	D	1's
3	0	0	1	1	2
4	0	1	0	0	1
7	0	1	1	1	3
9	1	0	0	1	2
10	1	0	1	0	2
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
13	1	1	0	1	3
14	1	1	1	0	3
15	1	1	1	1	4

⇒

1's	m	A B C D
0	0	0000 ✓
1	4	0100 ✓
	1	0001 ✓
	2	0010 ✓
2	3	0011 ✓
	9	1001 ✓
	10	1010 ✓
3	7	0111 ✓
	13	1101 ✓
	14	1110 ✓
4	15	1111 ✓

⇒

		A B C D	
0-1	0-1	000-	✓
	0-2	00-0	✓
	0-4	0-00	a
1-2	1-3	00-1	✓
	1-9	-001	b
	2-3	001-	✓
	2-10	-010	c
2-3	3-7	0-11	d
		1-01	e
		1-10	f
3-4	7-15	-111	g
	13-15	11-0	h
	14-15	111-	i

		A B C D	
0-1-2	0-1-2-3	00--	j

⇓

Tabla de implicantes. // Ya no consideramos a los irrelevantes

	3	4	7	9	10
a		/			
b				/	
c					/
d	/		/		
e				/	
f					/
g			/		
~~h~~					
~~i~~					
j	/				

// Pasa directo → x ser único en la columna.

// Se quitan x tener irrelevantes.

	3	7	9	10
b			/	
c				/
d	/	/		
e			/	
f				/
g		/		
j	/			

// g y j contenidas en d

// b=e ; c=f

Solución: $F(A,B,C,D) = a + (b \text{ ó } e) + (c \text{ ó } f) + d$

Opción 1 de solución:

$F(A,B,C,D) = a + b + c + d$

$a = 0X00 \rightarrow \bar{A}\bar{C}\bar{D}$

$b = X001 \rightarrow \bar{B}\bar{C}D$

$c = X010 \rightarrow \bar{B}C\bar{D}$

$d = 0X11 \rightarrow \bar{A}CD$

$\Rightarrow F(A,B,C,D) = \bar{A}\bar{C}\bar{D} + \bar{B}\bar{C}D + \bar{B}C\bar{D} + \bar{A}CD$

// para Π hubiera sido en vez de $\bar{B}C\bar{D}$, $B+\bar{C}+D$

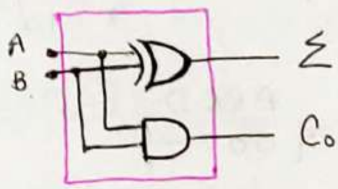
$$F(A,B,C,D) = \bar{A}\underbrace{(\bar{C}\bar{D}+CD)}_{\overline{C\oplus D}} + \bar{B}\underbrace{(\bar{C}D+C\bar{D})}_{C\oplus D}$$

$$F(A,B,C,D) = \bar{A}(\overline{C\oplus D}) + B(C\oplus D)$$

Miércoles

SUMADOR.

a) Medio Sumador

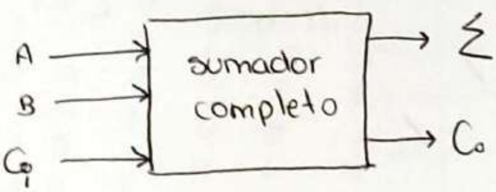


A	B	Σ	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\Rightarrow \Sigma = \bar{A}B + A\bar{B} = A\oplus B$

$C_o = AB$

b) Sumador Completo.



A	B	Ci	Σ	Co
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

$\Sigma = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$

$\Sigma = C_i(\bar{A}\bar{B}+AB) + \bar{C}_i(\bar{A}B+A\bar{B})$

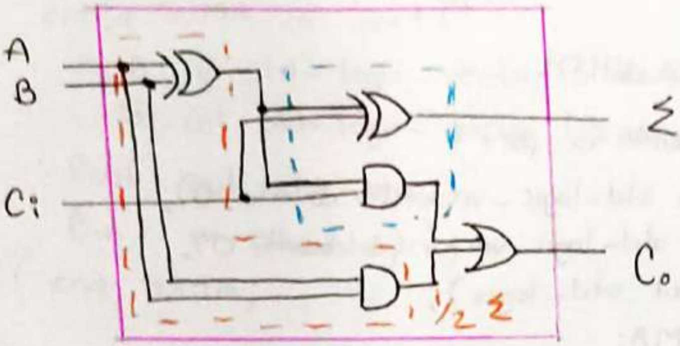
$\Sigma = C_i(\overline{A\oplus B}) + \bar{C}_i(A\oplus B)$

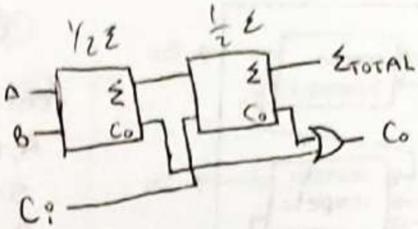
$\boxed{\Sigma = C_i \oplus (A\oplus B)}$

$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$

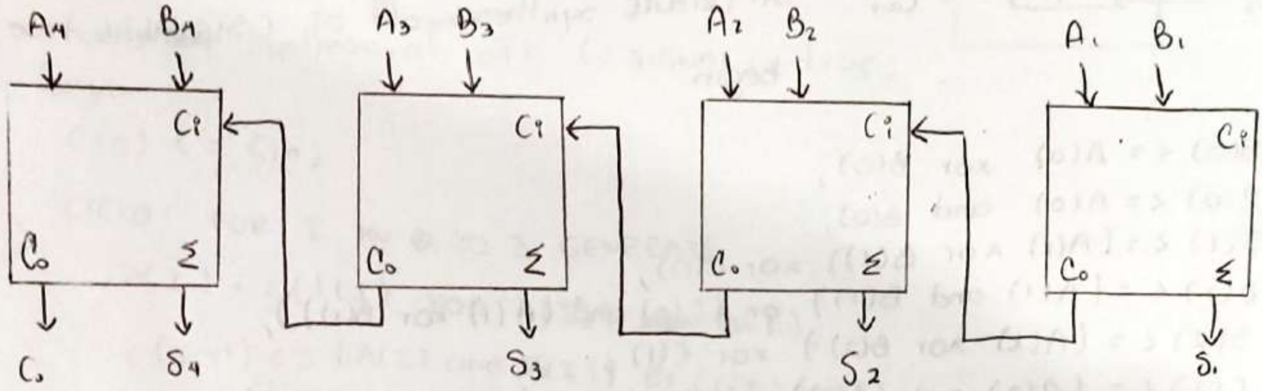
$C_o = C_i(\bar{A}B + A\bar{B}) + AB\cancel{(C_i+\bar{C}_i)}^{=1}$

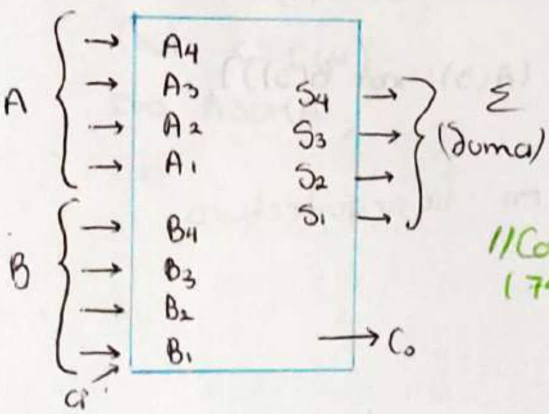
$C_o = C_i(A\oplus B) + AB$





c) Sumador de 4 bits.

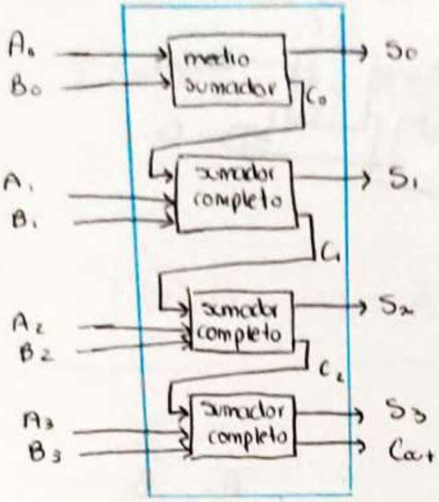




//Comercial
(74LS283)

Para hacer un restador
⇒ Se ponen inversores
en B (o A) y en el
acarreo de entrada
hay que poner un 1.

31/03/17

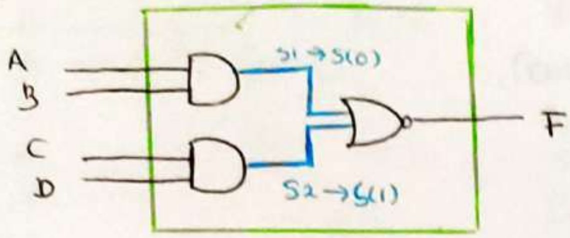


CODIGO VHDL

```
entity SUMA is port (
A, B: in std_logic_vector (3 downto 0);
 S: out std_logic_vector (3 downto 0);
 Cout : out std_logic);
end SUMA;

architecture ASUMA of SUMA is
signal C: std_logic_vector (2 downto 0);
ATTRIBUTE synthesis_off OF C:SIGNAL is true;
begin

S(0) <= A(0) xor B(0);
C(0) <= A(0) and B(0);
S(1) <= (A(1) xor B(1)) xor C(0);
C(1) <= (A(1) and B(1)) or (C(0) and (A(1) xor B(1)));
S(2) <= (A(2) xor B(2)) xor C(1);
C(2) <= (A(2) and B(2)) or (C(1) and (A(2) xor B(2)));
S(3) <= (A(3) xor B(3)) xor C(2);
Cout <= (A(3) and B(3)) or (C(2) and (A(3) xor B(3)));
end ASUMA

//Las conexiones internas se declaran en la arquitectura
// con "signal"
```

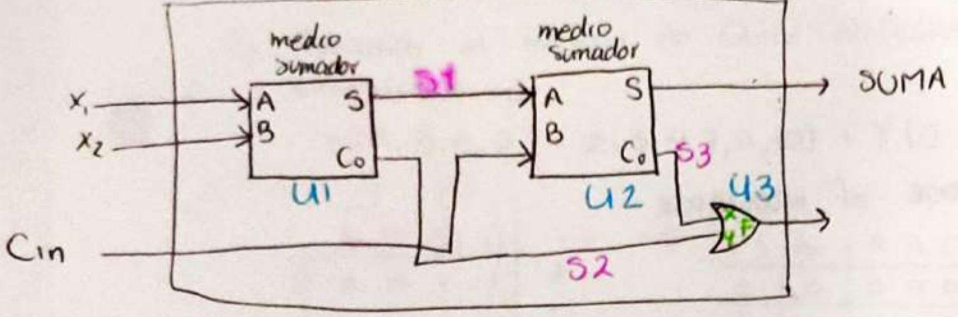
```
entity SUMA is port (
  A, B : in std_logic_vector (3 downto 0);
    S: out std_logic_vector (3 down to 0);
  Cout : out std_logic ;
  Cin : in std_logic);
end SUMA;

architecture ASUMA of SUMA is
  signal C: std_logic_vector (4 downto 0);
  attribute synthesis_off of C: SIGNAL is true;
begin
  C(0) <= Cin;
  CICLO: FOR I IN 0 TO 3 GENERATE
    S(I) <= ((C(I) xor A(I)) xor B(I);
    c(I+1) <= (A(I) and B(I)) or (C(I) and (A(I) xor B(I)));
  end GENERATE;
  Cout <= C(4);
End ASUMA;
```

Estilo Estructura en VHDL ~~A~~ :D




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

entity CIRCUITO is port (
   A, B, C, D:  in std_logic;
          F:  out std_logic;
end entity;

architecture ACIRCUITO of CIRCUITO is
signal    S: std_logic_vector (1 downto 0);
begin
                    # entradas     entradas        → salida
  U1:  and2   PORT MAP (A, B, S(0));
  U2:  and2   PORT MAP (C, D, S(1));                    //ALU
  U3:  nor2   PORT MAP (S(0), S(1), F);                 //Procesador.
end ACIRCUITO;
```

Sumador Completo.



CODIGO VHDL

```
entity SUMADOR is port (
 x1, x2, Cin: in bit;
 SUMA, Cout : out bit);
end SUMADOR;

architecture ASUMADOR of SUMADOR is

signal S1, S2, S3 : bit;
  component MSUMADOR
     port (A, B: in bit;
           S, Co: out bit);
  end component;

  component OR-GATE
     port (x, Y: in bit;
           F: out bit);
  end component;

 begin
   U1:  MSUMADOR   port map( x1, x2, S1, S2);
   U2:  MSUMADOR   port map (S1, Cin, SUMA, S3);
   U3:  OR-GATE    port map (S3, S2, Cout);

end ASUMADOR;
```

```
entity MSUMADOR is port (
  A, B: in bit
  S, Co: out bit;
end MSUMADOR;

  architecture A-MSUMADOR of MSUMADOR
  is begin
     S <= A xor B;
    Co <= A and B;
  end A-MSUMADOR,

entity ORGATE is port (
    x, Y: in bit;
    F : out bit);
  end OR-GATE;

  architecture A-OR-GATE of OR-GATE is
  begin
     F <= x or Y;
   end A-OR-GATE;
```

//(top level) clic derecho al
//archivo principal.

Ej. Aplicando el método de Quine McCluskey, obtener la función simplificada de:

$$F(A,B,C,D) = \Sigma(3,4,7,9,10) + \gamma(0\text{-}2, 13\text{-}15)$$

	A	B	C	D	1's
3	0	0	1	1	2
4	0	1	0	0	1
7	0	1	1	1	3
9	1	0	0	1	2
10	1	0	1	0	2
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
13	1	1	0	1	3
14	1	1	1	0	3
15	1	1	1	1	4

⇒

1's	m	A B C D
0	0	0 0 0 0
1	1	0 0 0 1
	2	0 0 1 0
	4	0 1 0 0
2	3	0 0 1 1
	9	1 0 0 1
	10	1 0 1 0
3	7	0 1 1 1
	13	1 1 0 1
	14	1 1 1 0
4	15	1 1 1 1

⇒

Grupo	Min	A B C D	
0-1	0-1	0 0 0 -	✓
	0-2	0 0 - 0	✓
	0-4	0 - 0 0	a
1-2	1-3	0 0 - 1	✓
	1-9	- 0 0 1	b
	2-3	0 0 1 -	✓
	2-10	- 0 1 0	c
2-3	3-7	0 - 1 1	d
	9-13	1 - 0 1	e
	10-14	1 - 1 0	f
3-4	7-15	- 1 1 1	g
	13-15	1 1 - 1	h
	14-15	1 1 1 -	i

⇒

Grupo	min	A B C D	
0-1-2	0-1-2-3	0 0 - -	j

⇒ Tabla de implicantes

// ya no se consideran los irrelevantes.

	3	4	7	9	10
✓ a		/			
b				/	
c					/
d	/		/		
e				/	
f					/
g			/		
~~h~~					
~~i~~					
j	/				

// Hacer otra tabla.

	3	7	9	10
b			/	
c				/
d	/	/		
e			/	
f				/
g		/		
j	/			

// g y j contenidos en d

// b = e c = f.

para Π hubiera sido en vez de $\bar{B}C\bar{D}$: $B + \bar{C} + D$?

⇒ Sol : $F(A,B,C,D) = a + (b \text{ ó } e) + (c \text{ ó } f) + d$.

opción 1. de sol:

$F(A,B,C,D) = a + b + c + d$

$a = 0x00 = \bar{A}\bar{C}\bar{D}$

$b = x001 = \bar{B}\bar{C}D$

$c = x010 = \bar{B}C\bar{D}$

$d = 0x11 = \bar{A}CD$

$$\Rightarrow F(A,B,C,D) = \bar{A}\bar{C}\bar{D} + \bar{B}\bar{C}D + \bar{B}C\bar{D} + \bar{A}CD$$
$$= \bar{A}(\bar{C}\bar{D} + CD) + \bar{B}(\bar{C}D + C\bar{D})$$
$$= A(\overline{C \oplus D}) + \bar{B}(C \oplus D)$$

* SUMADOR

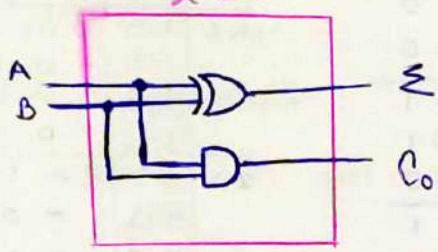
a) Medio Sumador.

A	B	Σ	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\Rightarrow \Sigma = \bar{A}B + A\bar{B} = A \oplus B$

$C_o = AB$

$\frac{1}{2}\Sigma$



b) Sumador Completo.

Kilo = 2^{10} ~~Mbem~~ M(mega) 2^{20} G(giga) (2^{30}) Tera (2^{40})

10 - A
11
12
13
14
15

En un sistema digital, los elementos discretos de información se representan mediante cantidades físicas llamadas señales. Las más comunes son la eléctricas, como voltajes y corrientes.

16 17 18 19 20 21

1.3 $\underbrace{1111}\underbrace{1111}\underbrace{1111} = 2048+1024+1536+1792+128+64$
$+32+16+8+4+2+1$
$= 6655$
$= FFF$

Codigos binarios.

Los sistemas digitales emplean señales que tienen dos valores distintos, y elementos de circuito que tienen dos estados estables.
Un numero binario de n digitos, por ejemplo, podría representarse con n elementos binarios de ctos., cada uno de los cuales tiene una señal de salida equivalente a 0 o 1.

Un codigo binario de n bits es un grupo de n bits que puede tener 2^n combinaciones distintas de unos y ceros; cada combinación representa un elemento del conjunto que se está codificando

Las combinaciones de bits de un codigo de n bits se determinan contando de 0 a $2^n - 1$.

Código BCD. "Decimal codificado en binario"

Sim. Decimal: Digito BCD

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Los códigos BCD y 2421 son ejemplos de códigos ponderados.

(Se asigna a cada posición de bit un factor de ponderación (o peso) de modo que cada dígito puede evaluarse sumando los pesos de todos los unos de la combinación codificada)

Los códigos 2421 y exceso-3 son ejemplos de códigos autocompletadores (el complemento a 9 de un # decimal se obtiene directamente cambiando 0's x 1's y 1's x 0's en el código.

Código Gray

Los datos de salida de muchos sistemas físicos producen cantidades que son continuas. Es necesario convertir esos datos a una forma digital antes de aplicarse a un ~~forma~~ sistema digital.

La información continua o analógica se convierte a una forma digital con un convertidor analógico a digital.

La ventaja del codigo Gray sobre la sucesión continua de números binarios es que la dif. entre dos números consecutivos cualesquiera en código Gray es de un solo bit.

Codigo Gray			Equivalente decimal
0000	0	m	0
0001	1	M	1
0011	3	M	2
0010	2	m	3
0110	6	m	4
0111	7	M	5
0101	5	M	6
0100	4	m	7
1100	12	m	8
1101	13	M	9

Codigos para detectar errores.

Cuando se desea detectar errores en la comunicación y el procesamiento de datos, a veces se añade un 8vo bit al caracter ASCII para indicar su paridad. El bit de paridad es un bit adicional que se incluye en un msj. de modo q' el # total de unos sea par o impar

El bit de paridad ayuda a detectar errores durante la transmisión de información de un lugar a otro.

Almacenamiento binario y registros

La información binaria de una computadora digital debe existir físicamente en algún medio de almacenamiento capaz de guardar bits individuales. Una celda binaria es un dispositivo que tiene dos estados estables y puede almacenar un bit de información.

Registros

Un registro es un grupo de celdas binarias. Un registro con n celdas es capaz de almacenar cualquier cantidad discreta de información que contenga n bits. El estado de un registro es una n-tupla de unos y ceros, en la que cada bit designa el estado de una celda de registro. El contenido de un registro es función de la interpretación que se da a la información ahí almacenada.

Un registro puede almacenar elementos discretos de info a la misma configuración de bits puede interpretarse de dif. maneras, dependiendo del tipo de los datos.

Transferencia de un registro

Un sistema digital se caracteriza por sus registros y los componentes que efectúan el procesamiento de datos. La operación transferencia de registro es básica en los sistemas digitales. Consiste en una transferencia de información binaria de un conjunto de registros a otros.

Los registros del sistema son los elementos básicos para almacenar y retener la info binaria. Los cts. de lógica digital procesan la info binaria almacenada en los registros.

$x + 0 = x$
$x + x' = 1$
$x + x = x$
$x + 1 = 1$ $\quad (x')' = x$
$x + y = y + x$
$x + (y + z) = (x + y) + z$
$\cancel{(x')'} = x$
$x(z + y) = xz + xy$
$x + xy = x$

Lógica binaria.

Se ocupa de variables que adoptan dos valores discretos y de operaciones que asumen un significado lógico.

Def.

Consiste en variables binarias y operaciones lógicas. Las variables se designan con letras del alfabeto, y cada variable tiene dos valores: 1 y 0. Hay 3 operaciones lógicas básicas: AND, OR, NOT.

1. AND $\quad x \cdot y = z$ ó $\quad xy = z \quad$ "x AND y es igual a z"
2. OR $\quad x + y = z \quad$ se lee "x OR y es = a z"
3. NOT $\quad x' = z \quad (o\ \bar{x} = z)$

Compuertas lógicas.

Son circuitos electrónicos que operan con una o más señales de entrada para producir una señal de salida

Tablas de vdd de las op. lógicas

AND

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

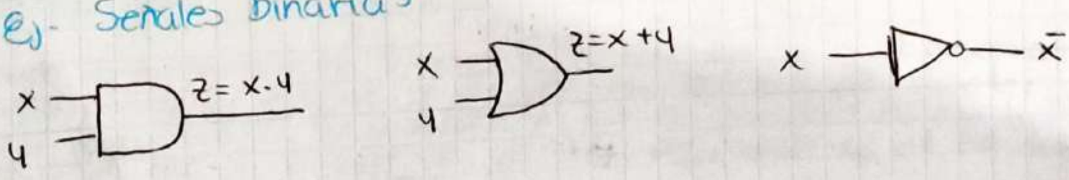
OR

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	$\bar{x}$
0	1
1	0

Ej. Señales binarias



Sistemas binarios.

x   0 1 1 0 0

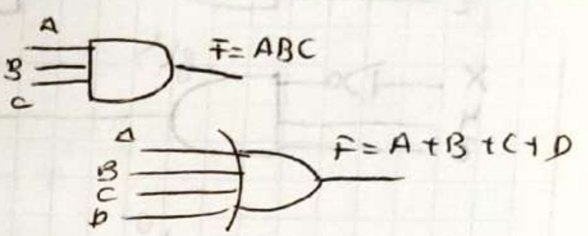
y   0 0 1 1 0

AND $x \cdot y$ 0 0 1 0 0

OR $x+y$ 0 1 1 1 0

NOT $\bar{x}$ 1 0 0 1 1

Señales de entrada-salida



$x(x'+y) = xx' + xy = 0 + xy = xy$

$x + x'y = (x+x')(x+y) = 1\cdot(x+y) = x+y$

$(x+y)(x+y') = x + (yy') = x + 0 = x$

$\cancel{xx'}^{\,0} + xy' + xy + \cancel{yy'}^{\,0} = x$

$x(y'+y) = x1 = x$

$xy + x'z + yz = xy + x'z + yz(x+x')$

$= xy + x'z + yzx + x'zy$

$= xy(1+z) + x'z(1+y)$

$= xy(1) + x'z(1)$

$= xy + x'z$

$(x+y)(x'+z)(y+z) =$ función 4 ?

$(x+y)(x'y + x'z + zy + z)$

$(x+y)(x'y + x'z + z)$

$(x+y)\cancel{(x'(y+z)+z)}$

$(x+y)(x'y + z) = (x+y)(x'y(z+z') + z)$

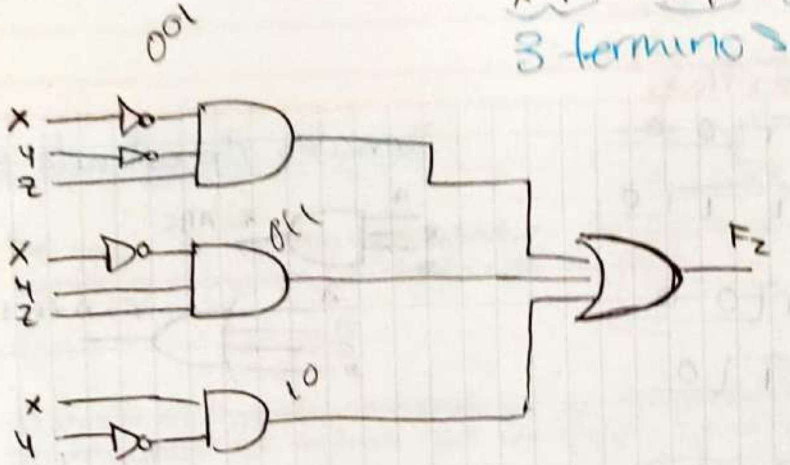
$= (x+y)(x'yz + x'z'y + z$?

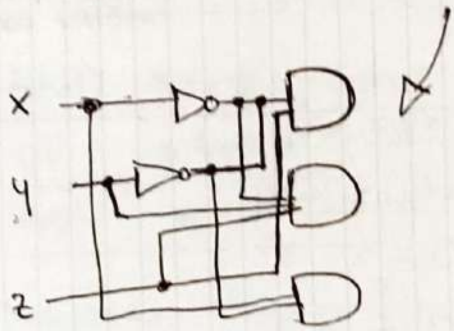
4

8 literales

$\underbrace{x'y'z}_{} + \underbrace{x'yz}_{} + \underbrace{xy'}_{}$

3 terminos 4





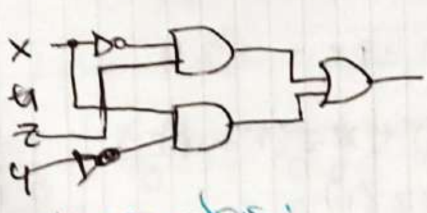
$F_2 = x'y'z + x'yz + xy'$

$= x'z(y'+y) + xy'$

$= x'z(1) + xy'$

$= x'z + xy$

$= \cancel{x(z+y)}$



O:

Complementos:

$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x+y'+z)(x+y+z')$

$F_2' = [x(y'z' + yz)]' = x' + [(y+z)(y'+z')]$

~~Algebra de~~ Boole

Tabla 2.4

x	y	z	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

Para f_1

$x'y'z,\ xy'z',\ xyz$

$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$ Mintérminos

$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$

Max...

$f_1 = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

Ahora $(f_1)' = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$
$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

$f_2 = x'y'z' + x'y'z + x'yz' + \cancel{xy'z' + xy'z + xyz' + xyz}$

$(f_2)' =$

$f_2 = x'y'z' + x'y'z + x'yz' + xy'z'$

$f_2' = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$
$= (M_0)(M_1)(M_2)(M_4)$

Suma de min?

Expresar la función booleana $F = A + B'C$ como suma de minitérminos

E: $A = A(B+B') = AB + AB'$

$AB(C+C') + AB'(C+C')$

$ABC + ABC' + AB'C + AB'C'$

$B'C(A+A') = AB'C + A'B'C$

$F = A + B'C = ABC + ABC' + \underline{AB'C} + \underline{AB'C'} + \underline{AB'C} + \underline{A'B'C}$

$\cancel{ABC + ABC'}$

$A'B'C + AB'C' + AB'C + ABC' + ABC$

$= m_1 + m_4 + m_5 + m_6 + m_7$

$F(A,B,C) = \Sigma(1,4,5,6,7)$

Otro procedimiento:

$F = A + B'C$

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

$F = xy + x'z = (xy + x')(xy + z)$

$= (x' + x)(x' + y)(z + x)(z + y)$

$= (x' + y)(z + x)(z + y)$

$(x' + y) + zz' = (x' + y + z)(x' + y + z')$

$(z + x) + yy' = \underline{(z + x + y)}(z + x + y')$ ✓

$(z + y) + xx' = \cancel{(z + y + x)}\cancel{(x' + y + z)}$

$F = (x + y + z)(z + x + y')(x' + y + z)(x' + y + z')$

$= M_0 M_2 M_4 M_5$

$F(x, y, z) = \Pi(0, 2, 4, 5)$

$(C)'$ C: complemento $m_j' = M$

$F = xy + x'z$

x	y	z	xy	x'z	F
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Suma de minterm.

$F(x, y, z) = \Sigma(1, 3, 6, 7)$

Producto de Max...

$F(x, y, z) = \Pi(0, 2, 4, 5)$

Resumen Cap. 2. Algebra booleana y compuertas lógicas

Algebra booleana: Conjunto de operadores y varios axiomas o postulados.

Operador binario: Definido sobre un conjunto S de elementos es una regla que asigna a cada par de elementos de S. un elemento único de S.

Los postulados más comunes que se utilizan para formular diversas estructuras algebraicas son:

1. Cerradura 2. Ley Asociativa 3. Ley Conmutativa 4. Elemento de identidad 5. Inverso 6. Ley distributiva.

=> Algebra booleana de dos valores <=

Se define sobre un conjunto de dos elementos, $B = \{1, 0\}$, con las reglas para los dos operadores binarios, + y •

x	y	x • y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

Principio de dualidad: Establece que toda expresión algebraica que pueda deducirse de los postulados de algebra booleana seguirá siendo válida si se intercambian los operadores y los elementos de identidad. En un algebra booleana de dos valores, los elementos de identidad y los elementos de conjunto, B, son los mismos; 1 y 0.

Si queremos el dual de una expresión algebraica solo cambiamos OR y AND y 1's x 0's y 0's x 1's.

Teorema: $x + x = x$

Postulado	$x + 0 = x$	$x \cdot 1 = x$
"	$x + x' = 1$	$x \cdot x' = 0$
Teorema	$x + x = x$	$x \cdot x = x$
Teorema	$x + 1 = 1$	$x \cdot 0 = 0$
Teorema	$(x')' = x$	
Postulado	$x + y = y + x$	$xy = yx$
Teorema	$x + (y + z) = (x + y) + z$	$x \cdot yz = xy \cdot z$
Postulado	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Teorema	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Teorema	$x + xy = x$	$x \cdot (x + y) = x$

Teorema $x + x = x$

$$
\begin{aligned}
x + x &= (x + x) \cdot 1 \\
&= (x + x)(x + x') \\
&= (x + xx') \\
&= x + 0 \\
&= x
\end{aligned}
$$

Teorema $x + 1 = 1$

$$
\begin{aligned}
x + 1 &= (x + 1) \cdot 1 \\
&= (x + 1) \cdot (x + x') \\
&= x + x' \cdot 1 \\
&= x + x' \\
&= 1
\end{aligned}
$$

Teorema ~~x+x~~ $x + xy = x$

$$
\begin{aligned}
x + xy &= x \cdot 1 + xy \\
&= x(1 + y) \\
&= x1 \\
&= x
\end{aligned}
$$

Es posible demostrar los teoremas de algebra booleana utilizando tablas de vdd.

Demostración de DeMorgan x medio de tablas...

x	y	$x + y$	$(x + y)'$	x'	y'	$x'y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

$\Rightarrow (x + y)' = x'y'$

Precedencia de operadores: 1) parént. 2) NOT 3) AND 4) OR

FUNCIONES BOOLEANAS.

Una función booleana descrita por una expresión algebraica consta de variables binarias, las constantes 0 y 1 y los símbolos lógicos de operación. Para un valor dado de las variables binarias, la función puede ser igual a 1 o bien a 0.

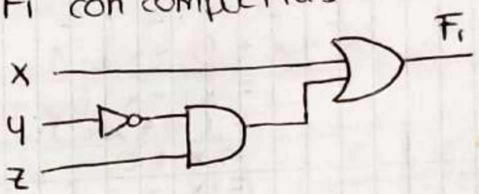
Una función booleana puede representarse x medio de una tabla de vdd.

Las combinaciones binarias para la tabla de vdd se obtienen de los #'s binarios, contando de 0 hasta $2^n - 1$.

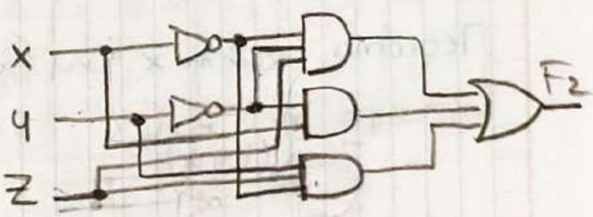
Ej. $F_1 = x + y'z$ $\quad F_2 = x'y'z + x'yz + xy'$

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

F_1 con compuertas



F_2

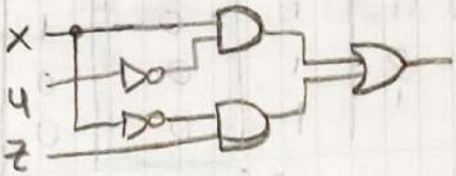


Pero F_2 puede simplificarse

$x'y'z + x'yz + xy'$
$= x'z(y' + y) + xy'$
$= x'z \cdot 1 + xy' = x'z + xy'$

★ La que tiene menos compuertas y menos entradas a las comp. es preferible x tener − componentes.

$\Rightarrow$



Manipulación algebraica.

literal: variable dentro de un término

Si reducimos el núm. de términos, el núm. de literales, o ambos, en una expre. booleana, podría obtenerse un cto. más sencillo.

Simplificar:

$x(x'+y) = xx' + xy = 0 + xy = xy$

$x + x'y = (x+x')(x+y) = 1\cdot(x+y) = x+y$

$(x+y)(x+y') = x + yy' = x + 0 = x$

4) $\boxed{xy + x'z + yz} = xy + x'z + yz(x+x')$

$= xy + x'z + xyz + x'yz$

$= xy(1+z) + x'z(1+y)$

$= xy(1) + x'z(1)$

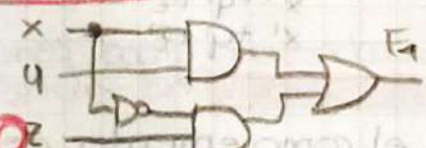
$= xy + x'z$

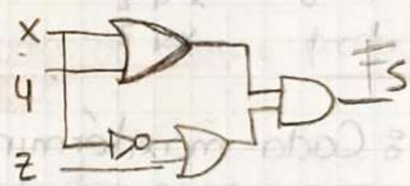
Dual

$\boxed{5.\ (x+y)(x'+z)(y+z)} = (x+y)(x'+z)$ Por dualidad Complem.

4.

Teorema del concenso





Complemento de una función

Compl. de una función F es F' (cambias 1's x 0's y 0's x 1's) (y AND's por OR's y viceversa)

$F_1 = x'yz' + x'y'z$ $F_2 = x(y'z' + yz)$

$(F_1)' = (x+y'+z)(x+y+z')$ $(F_2)' = x' + (y+z)(y'+z')$

Minitérminos y Maxitérminos.

Una variable binaria podría aparecer en su forma normal (x) o complementada (x'). Ahora considere 2. Cada variable podría aparecer en sus dos formas, hay cuatro combinaciones posibles ($x'y'$, $x'y$, xy' y xy). Cada uno de estos 4 términos es un minitérmino (considerar, q' se combinan con una operación AND)
↳ o producto estándar.

o sumas estandar? Complemento?

x	y	z	Minitérminos: Términos	Minitérminos: Designación	MAXITÉRMINOS: Términos	MAXITÉRMINOS: 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

Sí: Cada maxitérmino es el complemento de su minitérmino.

Ejemplo: $f_1 = x'yz + xy'z' + xyz = m_1 + m_4 + m_7$

$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$

x	y	z	f_1	f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

Se puede expresar algebraicamente una función booleana a partir de una tabla de vdd dada (formando un minitérmino para cada combinación de las variables q' produce **1** en la función y formando el OR

↳ ⇒ $f_1 = x'y'z + xy'z' + xyz$

$F_2 = x'yz + xy'z + xyz' + xyz$

Cualquier función booleana se puede expresar como una suma de minitérminos (suma refiere al OR)

Dual → Sob AND x OR y 0's x 1's ⇔

Maxi → Complem (literales)

Cualquier función booleana se puede expresar como un producto de maxitérminos (producto refiere al AND).

Para maxitérminos

formar miniter. para cada combinación q' produce 0 (1)

$\Rightarrow \quad f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

ahora sacar complemento: de la función. (2)

$f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$
$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

$f_2' = x'y'z' + x'y'z + x'yz' + xy'z'$
$f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$

Las funciones bool. expresadas como suma de mini. o producto de maxi están en forma canónica

Suma de minitérminos

Para expresar la función bool. en suma de minit. primero expandiendo la expresión a una suma de términos AND.

(2). Se examina cada término para ver si tiene todas las variab.

(3) Si hacen falta variables se les hace AND con una expresión como $(x + x')$ Donde x es una de las variab. faltantes.

Ejemplos

Expresar la fun. bool. $F = A + B'C$ como suma de min.

$A = A(B+B') = AB + AB'$
$AB(C+C') + AB'(C+C')$
$ABC + ABC' + AB'C + AB'C' \rightarrow F = ABC + ABC' + AB'C + AB'C' + A'B'C$
$m_7 + m_6 + m_5 + m_4 + m_1$
~~m + m4 + m5 + m~~

Ahora $B'C(A+A')$
$\cancel{B'CA} + A'B'C$

7

soma de min. abreviada $F(A,B,C) = \Sigma(1,4,5,6,7)$

PRODUCTO DE MAXITÉRMINOS

Para expresar la función booleana como producto de MAXIT., primero ponemos en formato de términos OR

① Aplicar ley distributiva $x + yz = (x+y)(x+z)$

② Hacer OR de cualquier variable faltante faltante x en cada término OR con xx' (Sea x la variable faltante).

Ejemplo: $F = xy + x'z$

$$\begin{aligned} F = xy + x'z &= (xy+x')(xy+z) \quad ① \\ &= (x'+x)(x'+y)(z+x)(z+y) \\ &= (x'+y)(z+x)(z+y) \end{aligned}$$

a $(x'+y)$ le falta $z \Rightarrow$

$$x'+y+zz' = (x'+y+z)(x'+y+z')$$

luego $(x+z)+yy' = (x+y+z)(x+y'+z)$

y $(y+z)+xx' = (\cancel{x+y+z})(\cancel{x'+y+z})$

$$\Rightarrow F = (x+y+z)(x'+y+z)(x'+y+z')(x+y'+z)$$

$$\therefore F(x,y,z) = \Pi(0,2,4,5)$$

Conversión entre formas canónicas.

Considere $F(A,B,C) = \Sigma(1,4,5,6,7)$ $F'(A,B,C) = \Sigma(0,2,3)$

Ahora Compl. de $F' = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3$
$= \Pi(0,2,3)$

~~x~~	~~y~~	~~xy~~	x'z
~~0~~	~~0~~	~~0~~	
~~0~~	~~1~~	~~0~~	
~~1~~	~~0~~	~~0~~	
~~1~~	~~1~~	~~1~~	

x	y	z	xy	x'z	xy + x'z
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Es posible convertir una función booleana de una expresión algebraica a un producto de maxitérminos utilizando una tabla de vdd. y el procedimiento de conversión canónica.

Ej. expresión booleana $F = xy + x'z$

Expresada como suma de minitérminos:

$$F(x,y,z) = \Sigma(1,3,6,7)$$

Expresada como producto de maxitérminos:

$$F(x,y,z) = \Pi(0,2,4,5)$$

FORMAS ESTANDAR

Las dos formas canónicas del algebra booleana son formas básicas obtenidas al leer una función de su tabla de verdad. Cada minitérmino o maxitérmino debe contener todas las variables, sea complementadas o sin complementar.

Otra forma de expresar funciones booleanas es en forma **estándar**
Hay dos tipos:

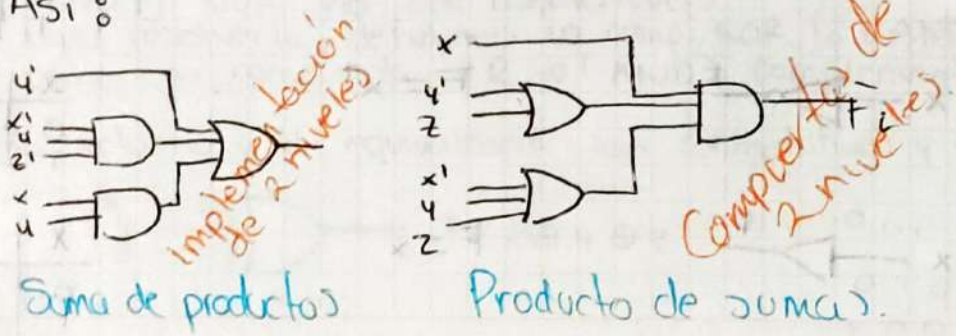
Suma de productos: Contiene términos AND (términos de producto, con una o más literales cada uno.

Ejemplo: $F_1 = \underbrace{y'}_{\text{término}} + \underbrace{xy}_{\text{término}} + \underbrace{x'yz'}_{\text{término con 3 literales}}$

Producto de sumas: Contiene términos OR (términos de suma)
Cada término puede tener cualquier cantidad de literales.
El producto denota el AND de éstos terminos.

Ejemplo: $F_2 = \underbrace{x}_{\text{término}}\underbrace{(x'+z)}_{\text{término}}\underbrace{(x'+y+z)}_{\text{término}}$

Así:



Suma de productos   Producto de sumas.

Ejemplo: $F_3 = AB + C(D+E)$ No es ni suma de prod. ni prod. de sumas

Puede transformarse a: $F_3 = AB + CD + CE$

⚠ Es preferible una implementación de dos niveles porque produce el mínimo de retardo en compuertas cuando la señal se propaga de las entradas a la salida.

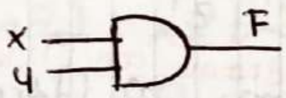
Las sig. 16 funciones se subdividen en:

1. Dos funciones que producen una constante 0 o 1
2. Cuatro funciones con operaciones unarias: complemento y transferencia.
3. Diez funciones con operadores binarios que definen 8 operaciones distintas: AND, OR, NAND, NOR, OR exclusivo, equivalencia, inhibición e implicación.

☆ Compuertas lógicas Digitales:

Nombre | Símbolo gráfico | Función Algebraica | Tabla de verdad

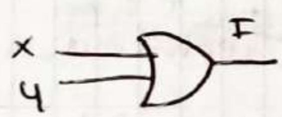
AND (y)



$F = xy$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR (o)



$F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Inversor (NOT)

$F = x'$

x	F
0	1
1	0

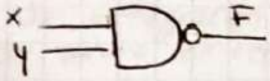
Búfer

$F = x$

x	F
0	0
1	1

Nombre — Símbolo gráfico — Función — Tabla de vdd

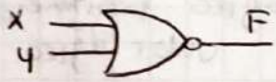
NAND



$F=(xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

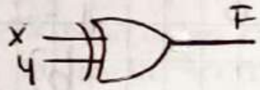
NOR



$F=(x+y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

OR Exclusivo
XOR

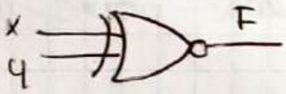


$F = xy' + x'y$
$= x \oplus y$

Da 1 cuando las entra. a 1 son impar

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

NOR Exclusivo o equivalencia



$F=(x\oplus y)'$
$xy + x'y'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Estas compuertas (a excepción del inversor y el búfer) se pueden extender de modo que tengan más de dos entradas.

NAND y NOR no son asociativos.
Para resolverlo, definimos la comp. NOR (o NAND) múltiple como una compuerta OR o (AND) complementada.

OR exclusivo y de equivalencia son conmutativas y asociativas.

$F = x \oplus y \oplus z$

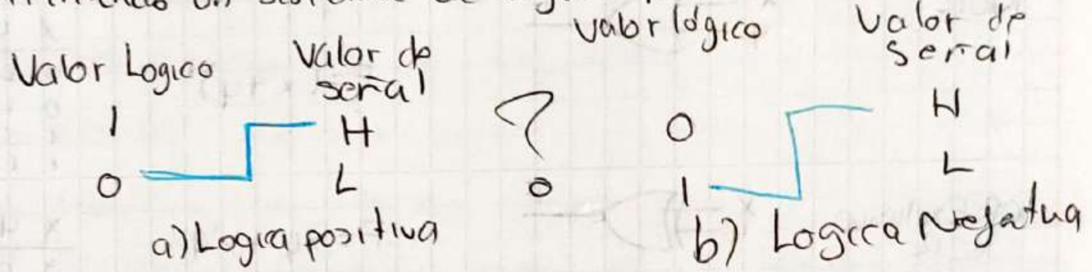
0 0 1	1
0 1 0	1
1 0 0	1
1 1 1	1

impar

Lógica positiva y negativa.

El nivel de señal más alto: H
" " " " " bajo: L

Si escogemos el nivel alto H para representar el 1 lógico, estamos definiendo un sistema de lógica positiva



La conversión de lógica positiva a negativa y viceversa, es cambiar 1's a 0's y 0's a 1's tanto en entradas como en salidas.

CIRCUITOS INTEGRADOS

Un circuito integrado **(CI)** es un cristal semiconductor de silicio, llamado chip, que contiene los componentes electrónicos para construir compuertas digitales. Las diversas compuertas se interconectan dentro dentro del chip para formar el circuito requerido.

Niveles de integración.

Los CI digitales suelen clasificarse según la complejidad de sus circuitos, la cual se mide por el # de compuertas lógicas incluidas
Hay:

- Dispositivos de integración a pequeña escala (SSI) contienen varias compuertas independientes en un solo paquete

- Dispositivos de integración a mediana escala (MSI) tienen una complejidad de entre 10 y 1000 compuertas x paq

- Dispositivos de integración a gran escala (LSI) contienen miles de compuertas en un solo paquete.

- Dispositivos de integración a muy grande escala (VLSI) contiene cientos de miles de compuertas en un solo paquete. (matrices de memoria y los microprocesadores complejos)

Familias de lógica digital.

Los circuitos lógicos integrados se clasifican también por la tecnología específica de circuitos utilizada en su construcción.
Llamamos a esa tecnología familia de lógica digital.
Cada familia tiene su propio circuito electrónico básico sobre el cual se desarrollan circuitos digitales y componentes más complejos.
El cto. básico en cada tecnología es una compuerta NAND, NOR o inversora. Por lo regular se usan los componentes electrónicos empleados en la construcción del cto básico para dar nombre a la tecnología.

TTL lógica transistor - transistor
ECL lógica acoplado por emisor
MOS metal - óxido - semiconductor
CMOS metal - óxido - semiconductor complementario.

TTL familia lógica estándar
ECL resulta ventajoso en sistemas que deben operar a alta velocidad.
MOS es apropiado para cts. que requieren una densidad elevada de comp.
CMOS preferible en sistemas q' requieren bajo consumo de energía.

Características:

- Abanico de salida (Fan-out): # de carga estándar que la salida de una compuerta representativa es capaz de alimentar sin merma de su funcionamiento normal.

- Abanico de entrada (fan-in): # de entradas con que cuenta la compuerta

- Disipación de potencia: Energía consumida por la compuerta y que la fte. debe suministrar.

- Retardo de propagación: Tiempo medio de transición que la señal tarda al propagarse de la entrada a la salida. La vel. de op. es inversamente proporcional al retardo de propagación.
- Margen de ruido: Voltaje externo máximo de ruido que puede añadirse a una señal de entrada sin causar un cambio indeseable en la salida del cto.

DISEÑO ASISTIDO POR COMPUTADORA : CAD

Las herramientas CAD consisten en programas de software que apoyan la representación computarizada y ayudan a desarrollar hardware digital automatizando el proceso de diseño. La automatización del diseño electrónico cubre todas las fases del diseño de CI.

El diseñador puede escoger entre un circuito integrado para una aplicación específica (ASIC), un arreglo de compuertas programable en el campo (FPGA), un dispositivo de lógica programable (PLD) o un CI hecho totalmente a la medida.

Algunos sistemas CAD incluyen un programa editor para crear y modificar diagramas esquemáticos en la pantalla de una compu. Este proceso se llama captura de esquemas o introducción de esquemas.

>> Un adelanto importante en el diseño de sistemas digitales es el uso de un lenguaje de descripción de hardware (HDL)

El HDL se parece a los lenguajes de programación, pero está orientado específicamente a la descripción de hardware digital. Representa diagramas de lógica y otra información digital en forma textual. Sirve para simular el sistema antes de construirlo, a fin de verificar la funcionalidad y la operación. Una aplicación importante es su software de síntesis lógica, que automatiza el diseño de sistemas digitales. (3.9)

$(x+y+z)' = x'y'z'$ $\qquad$ $x'+y'+z' = (xyz)'$

Capítulo 2. Ejercicios

2.1

a)

x	y	z	$x+y+z$ (→A)	A'	$x'y'z'$	$(xyz)'$	$x'+y'+z'$	yz	$x+yz$	$x+y$	$x+z$	C
0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	0	1	0
0	1	0	1	0	0	1	1	0	0	1	0	0
0	1	1	1	0	0	1	1	1	1	1	1	1
1	0	0	1	0	0	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0	1	1	1	1
1	1	0	1	0	0	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1

$C = (x+y)(x+z)$

2.2

a) $xy + xy'$
$x(y+y')$
$x \cdot 1 = x$

b) $(x+y)(x+y')$
$x \cdot x + xy' + xy + yy'$
$x + x(y'+y) + 0$
$x + x \cdot 1$
$x + x = x$

c) $xyz + x'y + xyz'$
$xy(z+z') + x'y$
$xy + x'y$
$y(x+x') = y$

d) $(A+B)'(A'+B')'$
$(A+B)'(A')'(B')'$?
$(A'B')AB = 0$

2-3

a) $ABC + A'B + ABC'$
$AB(C+C') + A'B$
$AB + A'B = B(A+A')$
$= B$

b) $x'yz + xz$
$z(x'y + x) =$
$z(x+y) = \cancel{xz}+yz$

c) $(x+y)'(x'+y')$
$(x'y')(x'+y')$
$x'x'y' + x'y'y'$
$x'y' + x'y'$
$x'y'$

d) $xy + x(wz + wz')$?
$xy + x[w(z+z')]$ O
$xy + xw = x(y+w)$
$x(w+z)$

e) $(BC' + A'D)(AB' + CD')$
$BC'AB' + CD'BC' + A'DAB' + A'DCD'$
$= 0$:

2.4

a) $A'C' + ABC + AC'$
$C'(A'+A) + ABC$
$C' + ABC = C' + AB$

b) $(x'y'+z)' + z + xy + wz$
$(x'y')'z' + z + xy + wz$
$(x+y)z' + z + xy + wz$
$z'x + z'y + z + xy + wz$
$z'x + z + y + xy + wz$
$z + x + y + xy + wz$
$z + y + x + wz$
$z + y + x$

c) $A'B(D' + C'D) + B(A + A'CD)$
$A'B(D' + C') + B(A + CD)$
$A'BD' + A'BC' + BA + BCD$
$B(A'D' + A) + B(A'C' + CD)$ x
$B(A + D') + A'BC' + BCD$
$BA + BD' + A'BC' + BCD$
$B(A + A'C') + B(D' + CD)$
$B(A + C') + B(D' + C)$
$BA + BC' + BD' + BC$
$B(C' + C) + BA + BD'$
$B + BA + BD'$

→ $B + BD' = B$ O:

d) $(A'+C)(A'+C')(A+B+C'D)$

$(A'A' + A'C' + A'C + CC')(A+B+C'D)$

$(A'C' + A'C)(A+B+C'D)$

$A'(C'+C)(A+B+C'D)$

$A'A + A'B + A'C'D$

$A'(B+C'D)$ ○

2-5 $\boxed{F = x + yz \quad F' = x'(y'+z')}$

$FF' = (x+yz)\,x'(y'+z')$

$(x+yz)(x'y' + x'z')$

$xx'y' + xx'z' + yzx'y' + yzx'z'$

$F + F' = x + yz + x'(y'+z')$

$x + y' + z' + yz$

$x + y' + z' + y$

$x + z' + 1$

$x + 1 = 1$ ○

$AB'DE + AB'DE'$

2.6 Complementos: b) $AB'D + CD' + E = F \quad F' = (A'+B+D')(C'D)E'$

a) $(x'+y)(x+y')$ b) $\{[(A'+B)C'] + D\}E$ c) $(x'yz') + xz + xy$

$x'x + xy' + yx + yy'$

2.7 ○ 2.8

$F = xy + xy' + y'z$

x	y	z	xy	xy'	y'z	F
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	1	1	1
1	1	0	1	0	0	1
1	1	1	1	0	0	1

Check

$F(x,y,z) = \Sigma(1,4,5,6,7)$

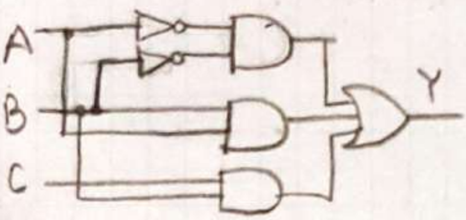
2.8

2.9 Bit x Bit

A	B	AND	OR	XOR	NOT A	NOT B
1	1	1	1	0	0	0
0	0	0	0	0	1	1
1	0	0	1	1	0	1
0	0	0	0	0	1	1
1	1	1	1	0	0	0
1	1	1	1	0	0	0
0	1	0	1	1	1	0
1	0	0	1	1	0	1

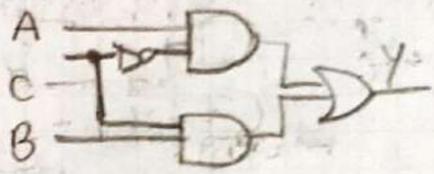
2.10 a) $A'B' + BA + BC$

a)

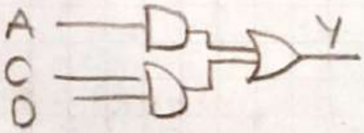


$Y = BC + AC'$

b)

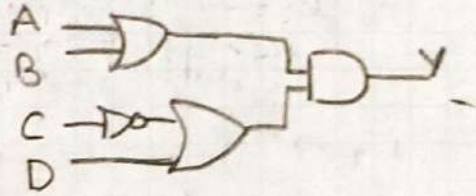


c) $Y = A + CD$



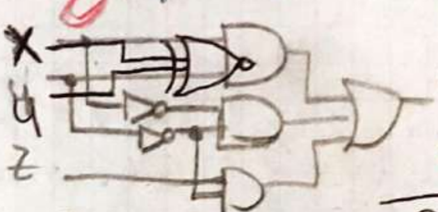
$Y = (A+B)(C'+D)$

d)

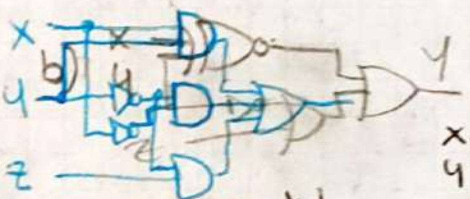


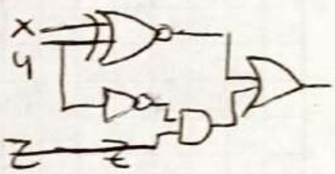
2.11 ? $F = (xy + x'y') + y'z$

a)



b)





$\overline{x \oplus y} = xy + x'y'$

d) ?

$x'y'(z'+z) + x'yz'$

$x'y' + x'yz' = x'(y' + yz') = x(y' + z')$

2.12 $T_1 = x'y'z' + x'y'z + xyz'$ $\quad T_2 = x'yz + xyz' + xy'z + xyz' + xyz$

$x'y'(z'+z) + x'yz'$

$x'y' + x'yz'$

$x'(y' + yz')$

$x'(y' + z')$ ✓

$= z'(x+y)$ Checar

$xz'(y'+y) + yz(x'+x) + xy'z$

$xz' + yz + xy'z$

$xz' + z(y + xy') = xz' + z(y+x)$

$xz' + zy + zx = zy + x(z'+z)$

$= zy + x$ ✓

2.13 ?

2.14. $(xy+z)(y+xz)$

x	y	z	xy+z	y+xz	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

$F(x,y,z) = \Sigma(3,5,6,7)$ ✓

$F(x,y,z) = \Pi(0,1,2,4)$

10:38

A	B	C	A'+B	B'+C	F
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1

$F(A,B,C) = \Sigma(0,1,3,7)$

$F(A,B,C) = \Pi(2,4,5,6)$

2.15 $F\ xy'z + x'y'z + w'xy + wx'y + wxy)$

a)

x	y	w	z	xy'z	x'y'z	w'xy	wx'y	wxy	F
0	0	0	0	0	0	0	0		0
0	0	0	1	0	1	0	0		1
0	0	1	0	0	0	0	0		0
0	0	1	1	0	1	0	0		1
0	1	0	0	0	0	0	0		0
0	1	0	1	0	0	0	0		0
0	1	1	0	0	0	0	1		1
0	1	1	1	0	0	0	1		1
1	0	0	0	0	0	0	0		0
1	0	0	1	1	0	0	0		1
1	0	1	0	0	0	0	0		0
1	0	1	1	1	0	0	0		1
1	1	0	0	0	0	1	0		1
1	1	0	1	0	0	1	0		1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	0	0	0	1	1	1

$y'z(x+x') + xy(w'+w) + wx'y$

$y'z + xy + wx'y$

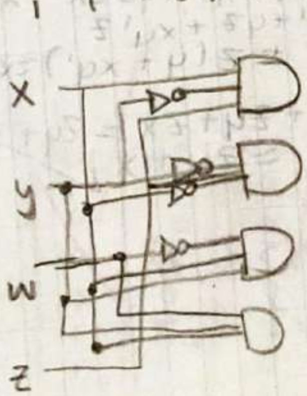
$y'z + y(x + x'w)$

$y'z + y(x+w)$

$y'z + yx + yw$ ✓

Simplif.

c)



b)