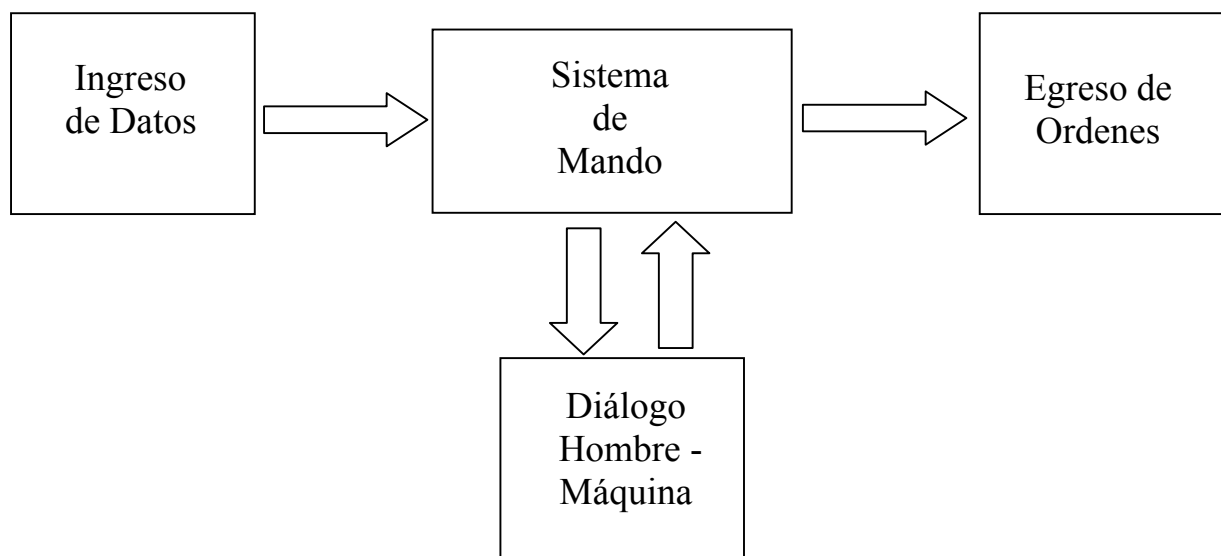


Conceptos Lógicos Básicos

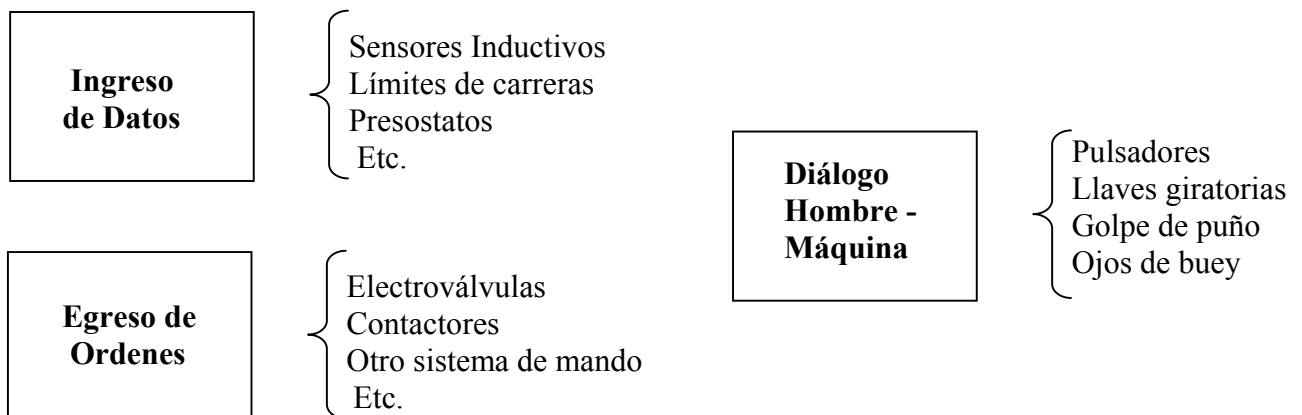
El número de dispositivos industriales que tienen 2 únicos estados de funcionamiento es muy elevado así una lámpara puede estar encendida (estado lógico “1”) o apagada (estado lógico “0”); un contacto eléctrico puede estar accionado (“1”) o no (“0”); un relé puede estar excitado (“1”) o no (“0”), etc. Esta naturaleza de dos estados (todo - nada o nivel alto - bajo) de muchos dispositivos industriales, hace posible tratar su funcionamiento mediante un álgebra binaria. El álgebra de Boole, como veremos satisface estos requisitos.

Características Básicas de un Automatismo

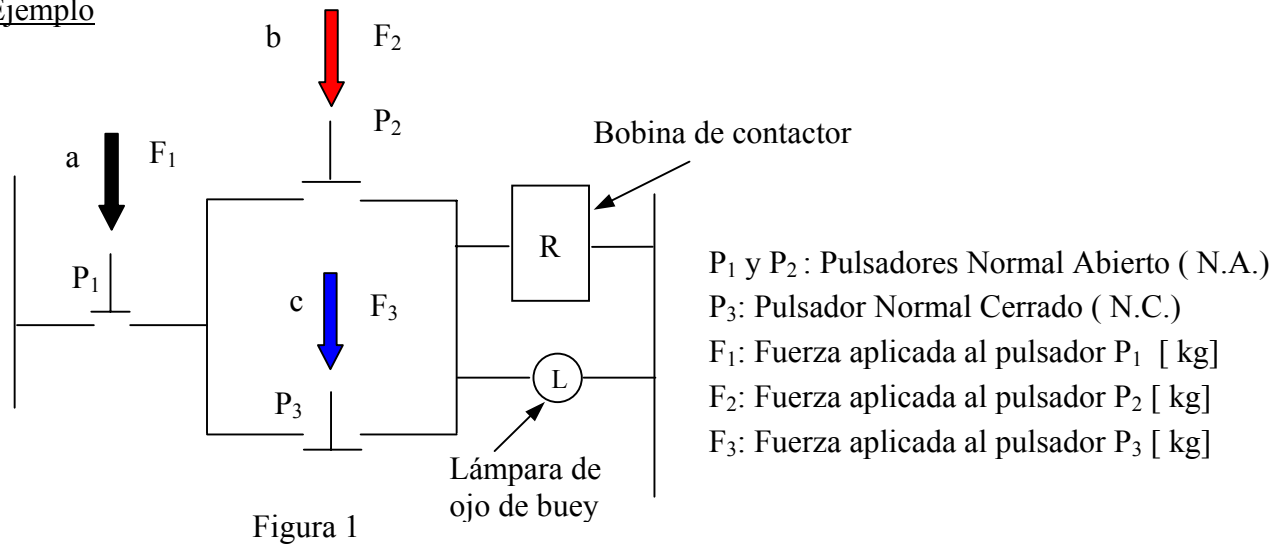
En todo sistema lógico se presentan las siguientes características.



El fin que buscamos es **diseñar el sistema de mando**



Ejemplo



En la Figura 1, la bobina del contactor estará energizada (y la lámpara encendida) **solamente** en las siguientes 2 situaciones:

- Cuando este accionado P_1 y P_2 .
- Cuando este accionado P_1 y no este accionado P_3 .

En la figura anterior, **a** es una variable binaria que valdrá 1 cuando la fuerza F_1 este aplicada; **b** es una variable binaria que valdrá 1 cuando la fuerza F_2 este aplicada y **c** es una variable binaria que valdrá 1 cuando la fuerza F_3 este aplicada. La variable binaria **R** valdrá 1 mientras la bobina del contactor este energizada, en tanto la variable binaria **L** valdrá 1 mientras la lámpara de ojo de buey este encendida.

Las primeras tres variables binarias (que indican si la respectiva fuerza esta aplicada o no) son las variables independientes (binarias) o entradas al sistema de mando; en tanto que las últimas dos variables binarias son las variables dependientes (binarias) o salidas del sistema de mando. Esto ultimo se indica en la Figura 2.



Figura 2

Funciones Lógicas

Definición:

Una función lógica es una variable binaria dependiente, cuyo valor es igual a una expresión algebraica binaria en la que se relacionan entre sí las variables binarias independientes, por medio de 3 operaciones lógicas básicas: inversión, suma lógica y producto lógico.

$$f = f(a, b, c, \dots\dots\dots)$$

Tabla de la verdad de una función lógica

La tabla de la verdad de una función lógica es una forma de representación de la misma, en la que se indica el valor "1" o "0" que toma la función para cada una de las posibles combinaciones de las variables de las cuales depende (variables independientes).

c	b	a	f = f(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Funciones Lógicas Básicas:

1. Función lógica inversión (not o negación)

Consideremos el siguiente sistema de mando (S.M):

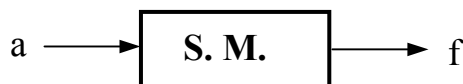


Figura N° 3

Afirmaremos que existe una función lógica inversión si el sistema de la Figura N° 3 está regido por la siguiente tabla de verdad

a	f
0	1
1	0

La forma algebraica de representar esta tabla de verdad está dada por la siguiente expresión

$$f = \overline{a}$$

Una rayita horizontal colocada encima de una variable, representa el valor inverso del estado lógico de dicha variable.

Ejemplo

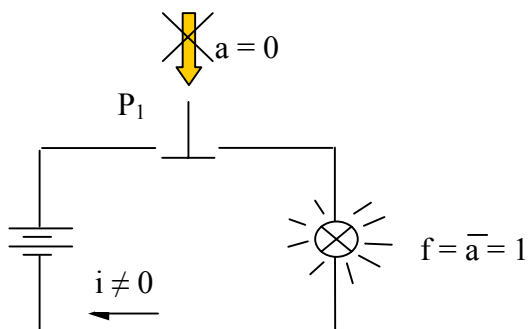


Figura N° 4 (Reposo)

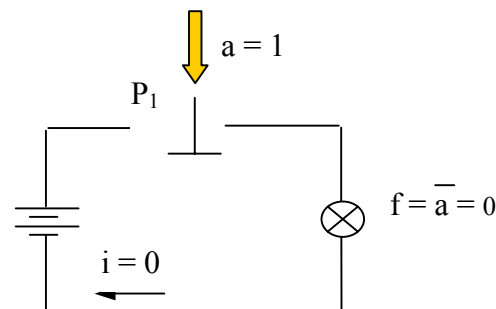


Figura N° 5 (Accionado)

La variable de entrada (independiente) **a**, indica si está accionado ($a=1$) o no ($a=0$) el pulsador normal cerrado P_1 .

Según la Figura N° 4, si la fuerza que acciona el pulsador es nula ($a=0$), éste estará en reposo y por ser un normal cerrado, la lámpara (salida f) estará encendida ($f=1$).

En la Figura N° 5, ocurre lo contrario, cuando no es nula la fuerza que acciona el pulsador (para $a=1$) f valdrá 0.

Podemos concluir que la variable de salida del sistema de mando es la inversa de la variable de entrada, siendo el inversor el pulsador N.C.

Estamos en condiciones de establecer la siguiente nomenclatura general:

Si un pulsador (independientemente si es N.O o N.C.) está abierto, su estado lógico es “0” y no conducirá corriente.

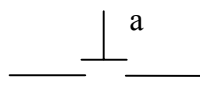
Si un pulsador (independientemente si es N.O o N.C.) está cerrado, su estado lógico es “1” y conducirá corriente.

De esto y lo visto anteriormente, se puede concluir que la relación entre el estado lógico de un pulsador y de la variable binaria **a** que indica si está aplicada o no una fuerza sobre dicho pulsador.

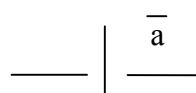
Estado lógico del Pulsador Normal Abierto = **a**

Estado lógico del Pulsador Normal Cerrado = \bar{a}

Por lo tanto cuando representemos a una pulsador si es un N.O lo haremos según la Figura N° 6, mientras que si es un N.C. lo haremos según la Figura N° 7

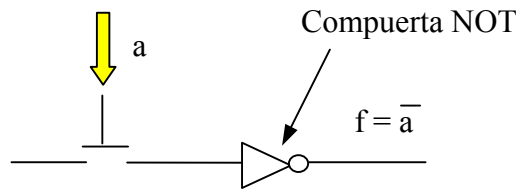


(Figura N° 6)



(Figura N° 7)

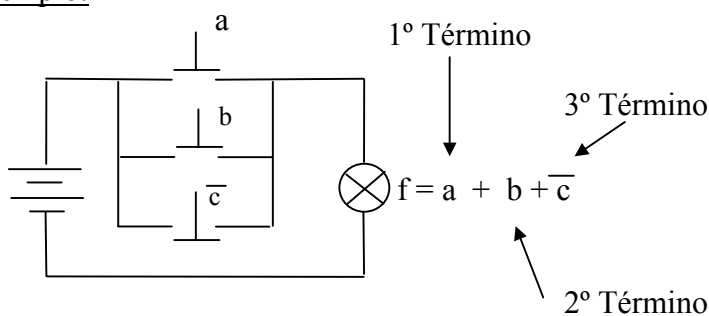
La implementación de la función lógica del ejemplo realizada con una compuerta "NOT" es:



2. Función Suma Lógica (OR u O)

Esta función lógica se caracteriza porque el estado lógico de la salida es "1" siempre que sea "1" el estado lógico de por lo menos uno de los términos de dicha función.

Ejemplo:

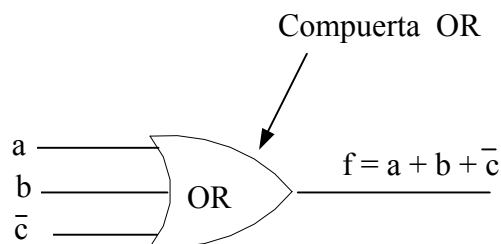


El signo "+" no significa una suma algebraica, sino una SUMA LÓGICA.

La tabla de verdad del ejemplo anterior es:

c	b	a	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

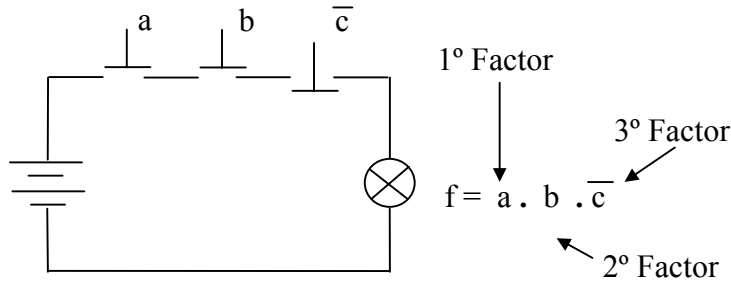
La implementación de la función lógica del ejemplo realizada con una compuerta "OR" es:



3. Función Producto Lógico (Y o AND)

Esta función lógica se caracteriza porque el estado lógico de la salida es “1” solamente si todos los factores de dicha función toman el valor “1”.

Ejemplo:

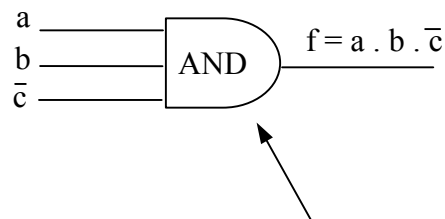


El signo “•” no significa un producto algebraico, sino un PRODUCTO LÓGICO.

La tabla de verdad del ejemplo anterior es:

c	b	a	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

La implementación de la función lógica del ejemplo realizada con una compuerta “AND” es:



Compuerta AND

ALGEBRA DE BOOLE

Introducción:

Esta se aplica a todas las variables que puedan tomar únicamente 2 estados lógicos (a los que llamaremos "0" y "1") estando relacionada mediante las operaciones lógicas de: inversión, suma y producto lógicos.

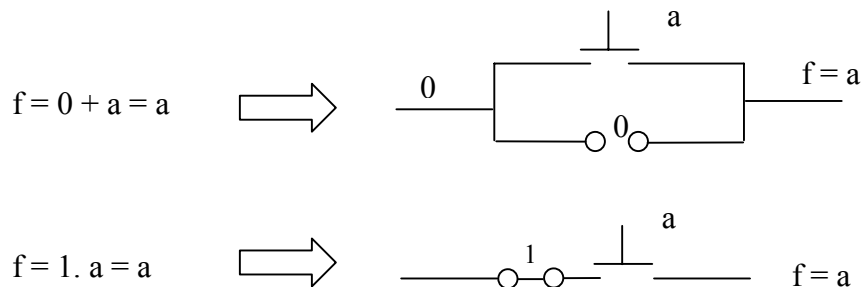
Postulados

El álgebra de Boole se basa en 4 postulados:

a) La suma y producto lógico son conmutativos:

$$a + b = b + a \quad ; \quad a \cdot b = b \cdot a$$

b) Dentro del álgebra, existen 2 elementos (el 0 y el 1) que tienen las siguientes propiedades:



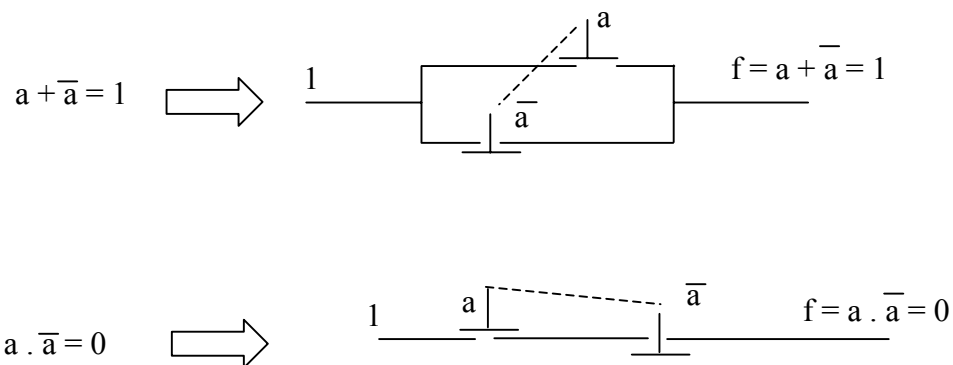
Se entiende que el 0 es un tramo del circuito que está siempre abierto y que el 1 es un tramo del circuito que esta siempre cortocircuitado.

c) Cada operación es distributiva con respecto a la otra:

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

d) Para cada variable a, existe otra variable llamada \bar{a} , tal que:



Estos 4 postulados se pueden verificar mediante el uso de tablas de verdad

Algunos teoremas del Álgebra de Boole

El álgebra de Boole es un conjunto de 7 teoremas, los cuales se basan en los 4 postulados anteriores. Nosotros veremos solo tres de ellos.

- **Teorema 1: Dualidad**

Dada una expresión, se puede encontrar otra de la siguiente manera:

$$\text{Se permuta} \left\{ \begin{array}{l} + \rightarrow \cdot \\ 0 \rightarrow 1 \\ \cdot \rightarrow + \\ 1 \rightarrow 0 \end{array} \right.$$

Este teorema se verifica de la simetría de los 4 postulados anteriores. Dual no es sinónimo de igual.

Ejemplo:

$$\begin{array}{ccccccc} f_1 = a + \bar{b} \cdot c \cdot (a + \bar{c}) + 0 & & & & & & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ f_2 = a \cdot \bar{b} + c + (a \cdot \bar{c}) \cdot 1 & & & & & & \end{array}$$

Esto se interpreta del siguiente modo: partiendo de una igualdad (f_1 y su lado derecho) se llega a otra igualdad (f_2 y su lado derecho) aunque f_1 no tiene que ser porque igual a f_2 . En este ejemplo si confeccionamos la tabla de verdad con a, b, c, f_1 y f_2 se llegara a la conclusión que f_1 no es igual a f_2 .

- **Teorema 7: Leyes de Demorgan**

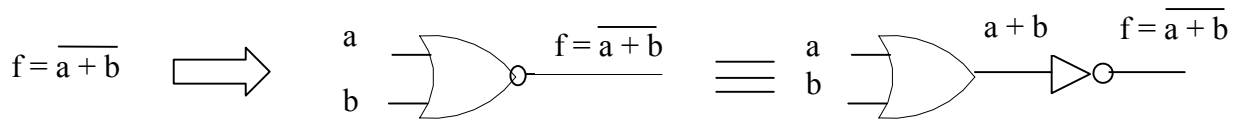
Este teorema se emplea para $\left\{ \begin{array}{l} \text{a) Origen de funciones inversas} \\ \text{b) Implementación con lógica cableada} \end{array} \right.$

$$\overline{a + b + c + \dots} = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \dots$$

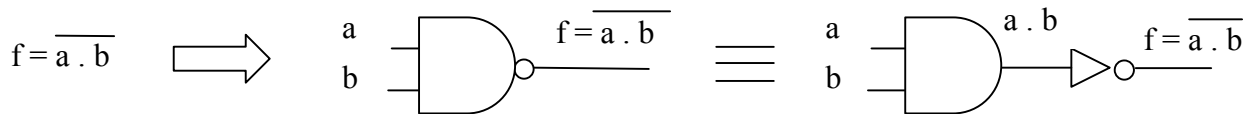
$$\overline{a \cdot b \cdot c \cdot \dots} = \bar{a} + \bar{b} + \bar{c} + \dots$$

a) Origen de las funciones inversas

• Función NOR



• Función NAND



b) Implementación de una función lógica con lógica cableada

Quiero implementar la siguiente función con lógica cableada:

$$f = \overline{a + b}$$

Esta función tal como esta escrita no puede implementarse físicamente con lógica cableada (no existe un paralelo negado). Sin embargo, si a la anterior le aplicamos la ley de DeMorgan, obtendremos:

$$f = \overline{a} \cdot \overline{b}$$

Esta última se puede implementar con dos pulsadores NC conectados en serie

Funciones Lógicas Canónicas

Se define como TÉRMINO CANÓNICO, a aquel término en el cual figuran TODAS las variables binarias de la entrada en el sistema. Si en una función lógica, todos sus términos son canónicos, la función lógica será también canónica.

Ejemplo:

$$f_1(a, b, c, d) = a \cdot b \cdot c \cdot \overline{d} + a \cdot b \cdot \overline{c} \cdot d + a \cdot b \cdot c$$

Vemos que el 1º y 2º término son canónicos, mientras que el 3º no lo es, por esto último f_1 no es canónica.

En cambio, la función lógica f_2 :

$$f_2(a, b, c, d) = a \cdot b \cdot c \cdot \bar{d} + a \cdot b \cdot \bar{c} \cdot d + a \cdot b \cdot c \cdot d$$

Es canónica por serlo cada uno de sus términos

En cambio, la función lógica f_3 :

$$f_3(a, b, c, d) = a \cdot b \cdot c \cdot \bar{d} + a \cdot \bar{b} \cdot c \cdot d + a \cdot b \cdot c \cdot d$$

No es canónica, por no serlo el 2º término (se puede demostrar que $a \cdot \bar{b} \cdot c \cdot d = a \cdot \bar{b} \cdot d + a \cdot c \cdot d$)

Desarrollaremos una forma abreviada de identificar cada uno de los términos canónicos de una función lógica. A cada uno de ellos los identificaremos según un número, de la siguiente forma:

$$\begin{array}{cccc}
 & a & b & \bar{c} & d \\
 & \Downarrow & & & \\
 a \cdot 2^0 & + & b \cdot 2^1 & + & c \cdot 2^2 & + & d \cdot 2^3 \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow \\
 1 \times 2^0 & + & 1 \times 2^1 & + & 0 \times 2^2 & + & 1 \times 2^3 = 1 + 2 + 0 + 8 = 11
 \end{array}$$

Habiendo llamado a la variable

$$\begin{array}{l}
 \text{acertada} \rightarrow 1 \\
 \text{negada} \rightarrow 0
 \end{array}$$

Aplicando el mismo razonamiento con el resto de los términos, la función canónica anterior la puedo representar de la siguiente manera:

$$f(a,b,c,d) = \sum_4(7, 11, 15)$$

El 4 que aparece en la sumatoria se debe a que existe 4 variables binarias en la entrada (a,b,c,d)

Hay que tener cuidado porque $\sum_{(4)}(3,5,7) \neq \sum_{(3)}(3,5,7)$

Deducción de la forma canónica de una función lógica a partir de su tabla de verdad

Construida la tabla de verdad se puede obtener la función lógica canónica expresada como una suma de términos canónicos. Para ello se efectúa la suma lógica de aquellos términos que hagan a la función igual a uno; ASIGNANDO AL ESTADO CERO LA VARIABLE INVERSA Y AL ESTADO UNO LA VARIABLE DIRECTA.

Ejemplo :

c	b	a	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

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

$$f(a,b,c) = \sum_3 (2,4,5)$$

SISTEMAS COMBINACIONALES

Son aquellos cuya salida dependen únicamente de la combinación de los valores alcanzados por las variables de entradas, sin tener en cuenta el orden en el que han evolucionado dichas variables.

Si analizamos esta definición, vemos que un sistema combinacional es realmente una función lógica tal como la hemos visto anteriormente. Por lo tanto los sistemas combinacionales pueden ser representados por una tabla de verdad o una expresión correspondiente a una suma de términos canónicos.

Simplificación de las funciones lógicas

El criterio que usaremos para simplificar (MINIMIZAR) una función lógica, es el de obtener una expresión en forma de sumas de términos,

TAL QUE TENGA UN NUMERO MINIMO DE TERMINOS CON EL MENOR NUMERO DE VARIABLES POSIBLES CADA UNO DE ELLOS.

Para esto nos valdremos del último postulado del álgebra de Boole:

$$a + \bar{a} = 1$$

Vale decir que si tenemos una suma de productos canónicos de la siguiente manera:

$$a b c + \bar{a} b c + \dots = (a + \bar{a}) b c + \dots = 1.b.c + \dots \quad \text{Ec. (1)}$$

Se cumplirá que cuando 2 términos se diferencian solo en el estado de una de la variables, el resultado será un único término en el cual queda suprimida dicha variable.

Estas formas de simplificación de las funciones lógicas, se pueden indicar en forma numérica.

Ejemplo:

$$a b c d + a b c \bar{d} + a \bar{b} c d + a \bar{b} c \bar{d} \quad \text{Ec. (2)}$$

15 7 11 3

De acuerdo con la Ecuación (1):

$$a b c d + a b c \bar{d} = a b c (d + \bar{d}) = a b c \quad \text{Ec. (3)}$$

$$a \bar{b} c d + a \bar{b} c \bar{d} = a \bar{b} c (d + \bar{d}) = a \bar{b} c \quad \text{Ec. (4)}$$

Para la Ecuación (3) se observa que los números 15 y 7 (que solo se diferencian en el estado lógico del bit correspondiente a la variable d) se diferencian en un número (8) que es potencia de 2 y que es igual al que le corresponde a la variable que desaparece (variable d)

Para el caso de la Ecuación (4) se observa que los números 11 y 3 (que solo se diferencian en el estado lógico del bit correspondiente a la variable d) se diferencian en un número (8) que es potencia de 2 y que es igual al que le corresponde a la variable que desaparece (variable d)

Para las Ecuaciones (3) y (4) los términos $a b c$ y $a b \bar{c}$ se pueden a su vez agrupar:

$$a b c + a b \bar{c} = a b (c + \bar{c}) = a b$$

Vemos que los números 7 y 3 (que solo se diferencian en el estado lógico del bit correspondiente a la variable c) se diferencian en el número 4, que es potencia de 2 y que es el que corresponde a la variable que desaparece (variable c).

Según el ejemplo que acabamos de ver podemos generalizar la siguiente conclusión:

“ Si 2 términos, que solo se diferencian en el estado lógico del bit correspondiente a una misma variable, sus números difieren siempre en un número que es potencia de dos, por lo que esa misma variable, no aparecerá en el término final”.

La reciproca no es valida por ejemplo los términos 11 y 13 se diferencian en un numero que es potencia de 2, pero difieren en los bits correspondientes a mas de una variable (b y c)

Métodos de Minimización de las funciones lógicas

Existen dos metodos de minimización óptimos y ello son:

- Diagrama de Karnaugh (limitado a 5 variables de entrada)
- Metodo de Quiney MacClousky

Sin embargo veremos un metodo de cuasi minimizacion, es decir que no es una minimizacion optima, pero que es más sencillo de aplicar que los dos anteriores.

Criterios de agrupamiento

El criterio de agrupamiento dependerá si el número de términos es par o impar.

- 1a) Si el número de términos es PAR, cada uno de ellos se agrupa solo una vez con otro término. La elección de los 2 términos es que entre ellos solo se diferencien en el estado lógico de un bit (cuya variable binaria asociada desaparecerá).
- 2a) En caso que haya 2 nuevos términos que solo se diferencien en el estado lógico de un bit, se vuelven a agrupar y este proceso continúa hasta que no se puedan obtener nuevos grupos.

Ejemplo

$$f(a,b,c,d) = \sum_4 (1,3,6,7,9,11)$$

$$1a) \begin{cases} 1-3 \rightarrow a \bar{b} \bar{c} \bar{d} + a b \bar{c} \bar{d} = a \bar{c} \bar{d} \\ 6-7 \rightarrow \bar{a} b c \bar{d} + a b c \bar{d} = b c \bar{d} \\ 9-11 \rightarrow a \bar{b} \bar{c} d + a b \bar{c} d = a \bar{c} d \end{cases}$$

2a) Se agrupa 1-3 con 9-11

$$a \bar{c} \bar{d} + a \bar{c} d = a \bar{c}$$

El resultado final es:

$$f(a,b,c,d) = a \bar{c} + b c \bar{d}$$

- 1b) Si el número de términos es IMPAR, se realiza el mismo procedimiento que en 1a), excepto que el último término se puede agrupar con otro término que haya participado en algún grupo, cumpliéndose siempre la condición que entre ellos solo se diferencien en el estado lógico de un bit.

2b) Se procede de la misma manera que en 2a).

Ejemplo: Ver Ejercicio 1-1