Unidad IV

Algebra Booleana

4.1 Teoremas y postulados.

Teoremas

Teorema 1: Multiplicación por cero (identidad) Es el factor neutro: Suma: a+1=!--------Producto: a0=0

Teorema 2: Absorción En la suma se identifica primero de forma aislada y luego multiplicando a otra expresión. Suma: A+(AB)=A----------Producto: A(A+B)=A

Teorema 3: Cancelación I Es cuando se encuentra una expresión sumada o multiplicada con su complemento: Suma:A+A'B=A+B-------Producto: A(A'+B)=AB

Teorema 4: Cancelación II Se identifica en 2 términos que comparten un factor común y otro que no es común, uno de ellos es el complemento de la otra: Suma: AB+A'B = B---------Producto:(A+B)(A'+B)=B

Teorema 5: Idempotencia Si se suma o multiplica el termino n número de vecez, dara por resultado el mismo.Suma: A+A+A=A---------Producto:(A)(A)(A)=A

Teorema 6: Consenso Se encuentran 2 términos que contengan una expresión en uno afirmada y en otro negada, anotar los términos con que se multiplica uno y otro, al final se busca otro elemento o termino que sea la multiplicación de estos 2 últimos, este ultimo se multiplica. Suma: AB+A'C+BC=AB+A'C---------- Producto: (A+B)(A'+C)(B+C)=(A+B)(A'+C)

Teorema 7: De Morgan Si hay suma complementada se puede hacer el producto de cada parte con su complemento.Suma: |A+B|=A'B'----------Producto: |AB|=A'+B'

Teorema 8: Involución El complemento de un complemento es el termino sin complementos.-----||A=A

Teorema 9: Complemento de neutros El complemento de la nada es el todo y el del todo es la nada.0'=1----1'=0

Postulados

Postulado 1: Definición En un sistema algebraico definido en un conjunto B, que contiene 2 o más elementos donde pueden darse solo 2 operaciones, la suma u operación "OR" y la multiplicación o multiplicación "AND"

Postulado 2: Identidad (existencia de neutros)En B, el elemento neutro de la suma determinada "0" y en la multiplicación "!" donde X en B: a)n+0=X----------b)X1=X

Postulado 3: Conmutatividad Para cada X,Y,Z en B: a)X+Y=Y+X-----b)XY=YX

Postulado 4: Asociatividad Para cada X,Y,Z en B: a)X+(Y+Z)=(X+Y)+Z---------b)X(YZ)=(XY)Z

Postulado 5: Distributividad Para cada X,Y,Z en B: a)X+(YZ)=(X+Y)(X+Z)----------b)X(Y+Z)=(XY)+(XZ)

Postulado 6: Existencia de complemento Para cada X en B existe un elemento único denotado por X' complemento tal que: a)X+X'= 1-------b)XX'=0

Ejemplos:

x + x = x
x + x = (x + x) . 1
x + x = (x + x) (x + x')
x + x = x + xx'
x + x = x + 0
x + x = x

x = x

x + xy = x
x . 1 + xy = x
x (1 + y) = x
x (y + 1) = x
x (1) = x

4.2 Optimización de expresiones booleanas.

Optimización de expresiones booleanas

Las expresiones booleanas se usan para determinar si un conjunto de una o más condiciones es verdadero o falso, y el resultado de su evaluación es un valor de verdad. Los operandos de una expresión booleana pueden ser cualquiera de los siguientes:

- Expresiones relacionales: que comparan dos valores y determinan si existe o no una cierta relación entre ellos (ver más adelante), tal como mfn<10;
- Funciones booleanas: tal como p (v24), que regresa un valor de verdad (estos se explican bajo "Funciones booleanas"). Las expresiones relacionales permiten determinar si una relación dada se verifica entre dos valores. La forma general de una expresión relacional es:

Expresión-1 operador-de-relación expresión-2

Dónde:

- Expresión-1 es una expresión numérica o de cadena
- Operador-de-relación es uno de los siguientes:

- = Igual
- No igual (diferente de)
- < Menor que
- <= Menor o igual que
- Mayor que
- >= Mayor o igual que
- Contiene (puede ser usado sólo en expresiones de cadena)

Expresión-2 es una expresión del mismo tipo que expresión-1, o sea, expresión- 1 y expresión-2 deben ser ambas expresiones numéricas o ambas expresiones de cadena.

Los operadores de relación = <> < <= > >=tienen su significado convencional cuando se aplican a expresiones numéricas (dentro de los límites de precisión de los valores numéricos definidos bajo "Expresiones numéricas"). Cuando se comparan expresiones de cadena, se aplican las siguientes reglas:

- Excepto por el operador ":" (contiene), las cadenas se comparan exactamente en la forma en que ocurren, o sea, las letras mayúsculas y minúsculas se comparan de acuerdo con el código ASCII que les corresponde (p.ej. A será considerada menor que a);

- Dos expresiones de cadena no son consideradas iguales, a menos que tengan la misma longitud. Si dos expresiones generan cadenas de diferente longitud que son idénticas, carácter por carácter, hasta el total de la longitud de la más corta, entonces, la más corta será considerada menor que la más larga.

El operador: (contiene), busca una cadena de caracteres (definida por expresión-2) en otra cadena (definida por expresión-1). Si el segundo operando existe en cualquier parte del segundo operando, el resultado es Verdadero (TRUE). Este operador es insensible al hecho de que los caracteres se hallen en mayúsculas o minúsculas: por lo que las letras minúsculas se consideran iguales a su letra mayúscula correspondiente. Por ejemplo, el resultado de: v10: 'química'

Será Verdadero (True) si, y sólo si, el campo 10 contiene la cadena química en caso contrario, el resultado será Falso (False). Nótese que el segundo operando puede ser cualquier cadena o carácter, y no necesita ser una palabra como tal. Por lo tanto, en este ejemplo, el resultado será Verdadero no sólo si el campo 10 contiene la palabra química, sino también si contuviera bioquímica, fotoquímicas, químicamente, etc.

Los operandos de una expresión booleana pueden combinarse con los operadores siguientes:

- NOT (NO) Este operador produce el valor Verdadero, si su operando es Falso; y el valor Falso, si su operando es Verdadero. El operador NOT sólo puede usarse como operador signo +, o sea, siempre se aplica a la expresión booleana que le sigue;
- AND (Y) Este operador produce el valor Verdadero si ambos operandos son Verdadero. Si cualquiera de los dos operandos es Falso, entonces el resultado será Falso;
- OR (O) Este operador realiza una operación O-inclusivo. El resultado es Verdadero si cualquiera de los dos operandos, o ambos son Verdadero. En caso contrario, es Falso.

Al evaluar expresiones booleanas, y en ausencia de paréntesis, CDS/ISIS ejecutará las operaciones NOT en primer lugar, después las operaciones AND, y finalmente las OR. Las series de dos o más operadores del mismo nivel, se ejecutan de izquierda a derecha. Se pueden usar paréntesis para alterar el orden de evaluación: las expresiones dentro de paréntesis se evalúan antes, y las expresiones entre paréntesis internos a otros, son evaluadas antes que las expresiones externas a los paréntesis.

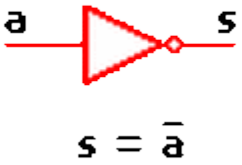
4.3 Aplicación del algebra booleana (Compuertas lógicas)

Las compuertas lógicas son dispositivos que operan con aquellos estados lógicos mencionados en lo anterior y funcionan igual que una calculadora, de un lado ingresas los datos, ésta realiza una operación, y finalmente, te muestra el resultado.

Cada una de las compuertas lógicas se las representa mediante un **Símbolo**, y la operación que realiza (**Operación lógica**) se corresponde con una tabla, llamada **Tabla de Verdad**, veamos la primera.

Compuerta **NOT**

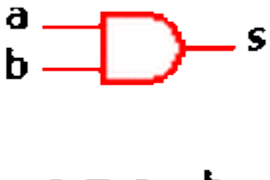
Se trata de un inversor, es decir, invierte el dato de entrada, por ejemplo; si pones su entrada a 1 (nivel alto) obtendrás en su salida un 0 (o nivel bajo), y viceversa. Esta compuerta dispone de una sola entrada. Su operación lógica es **s** igual a **a invertida**



a	s
0	1
1	0

Compuerta **AND**

Una compuerta AND tiene dos entradas como mínimo y su operación lógica es un producto entre ambas, no es un producto aritmético, aunque en este caso coincidan.*Observa que su salida será alta si sus dos entradas están a nivel alto*



a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta **OR**

Al igual que la anterior posee dos entradas como mínimo y la operación lógica, será una suma entre ambas... *Bueno, todo va bien hasta que 1 + 1 = 1*, el tema es que se trata de una compuerta **O Inclusiva** es como **a y/o b***Es decir, basta que una de ellas sea **1** para que su salida sea también 1*

$s = a + b$

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

Compuerta OR-EX o XOR

Es OR EXclusiva en este caso con dos entradas (puede tener más) y lo que hará con ellas será una suma lógica entre **a** por **b** **invertida** y **a** **invertida**por **b**.**Al ser **O Exclusiva** su salida será **1** si una y **sólo una** de sus entradas es 1**

$s = a.\bar{b} + \bar{a}.b$

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

Estas serían básicamente las compuertas más sencillas.

Compuertas Lógicas CombinadasAl agregar una compuerta NOT a cada una de las compuertas anteriores los resultados de sus respectivas tablas de verdad se invierten, y dan origen a tres nuevas compuertas llamadas NAND, NOR y NOR-EX. Veamos ahora como son y cuál es el símbolo que las representa...

Compuerta NAND

Responde a la inversión del producto lógico de sus entradas, en su representación simbólica se reemplaza la compuerta NOT por un círculo a la salida de la compuerta AND.

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

a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

Compuerta NOR

El resultado que se obtiene a la salida de esta compuerta resulta de la inversión de la operación lógica **o inclusiva** es como un **no a y/o b**. Igual que antes, solo agregas un círculo a la compuerta OR y ya tienes una NOR.

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

a	b	s
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta **NOR-EX**

Es simplemente la inversión de la compuerta OR-EX, los resultados se pueden apreciar en la tabla de verdad, que bien podrías compararla con la anterior y notar la diferencia, el símbolo que la representa lo tienes en el siguiente gráfico.

$$s = \overline{a.\bar{b} + \bar{a}.b}$$

a	b	s
0	0	1
0	1	0
1	0	0
1	1	1

Buffer's

En realidad no realiza ninguna operación lógica, su finalidad es amplificar un poco la señal (o refrescarla si se puede decir). Como puedes ver en el siguiente gráfico la señal de salida es la misma que de entrada.

$$s = a$$

a	s
0	0
1	1

4.3.1 Mini y maxi términos.

MINITÉRMINO (mi): Término producto en el que aparecen todas las variables, yasean complementadas o sin complementar.
FÓRMULA CANÓNICA DISYUNTIVA O DE MINITÉRMINOS
: suma de mintérminos (Suma de Productos).Dada la lista completa de mintérminos y asignando 1's y 0's arbitrariamente a las variables, siempre hay un, y sólo un, minitérmino que toma el valor 1. Un mintérmino es un término producto que es 1 exactamente en una línea de la tabla de Verdad.La fórmula compuesta por todos los mintérminos será idénticamente 1. Cada fórmula de conmutación puede expresarse como suma de mintérminos. Y esa fórmula es única.

Algunas veces es conveniente expresar la función booleana en la forma de suma de miniterminos. Si no puede hacerse en esta forma entonces puede realizarse primero por la expansión de la expresión en una suma de los términos AND.

Después cada término se inspecciona para ver si contiene todas las variables, si se han perdido una o más variables, se aplica el operador AND con una expresión x+x’ en donde x es una de las variables perdidas.

NOTACIÓN: Un mintérmino se designa por "mi" siendo i el número decimal correspondiente de la tabla de verdad. Para el producto, el 0 se asocia a la variable complementada y el 1 a la variable sin complementar.

EJEMPLO:C B A F(C,B,A)
0 0 0 1
0 0 1 0
0 1 0 1
0 1 1 1
1 0 0 0
1 0 1 0
1 1 0 0
1 1 1 1
F(C,B,A) = m0 + m2 + m3 +m7 = S m(0,2,3,7)
F(C,B,A) = C'·B'·A' + C'·B·A' + C'·B·A + C·B·A
O bien
F(C,B,A) = C·B·A + C·B·A + C·B·A + C·B·A

Ejemplo: Expresar la función F = A+B'C en una suma de miniterminos.

F= A+B'C
F(A,B,C)
A= A(B+B') = AB+AB'
= AB(C+C') + AB'(C+C')
= ABC + ABC' + AB'C +AB'C'

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

F = ABC+ABC'+AB'C+AB'C'+AB'C+A'B'C
F = A'B'C+AB'C' +AB'C+ABC'+ABC
F = m1+ m4+m5+ m6+ m7
F(A,B,C)=SUM(1,4,5,6,7)

La sumatoria representa al operador OR que opera en los términos y números siguientes son los minitérminos de la función.

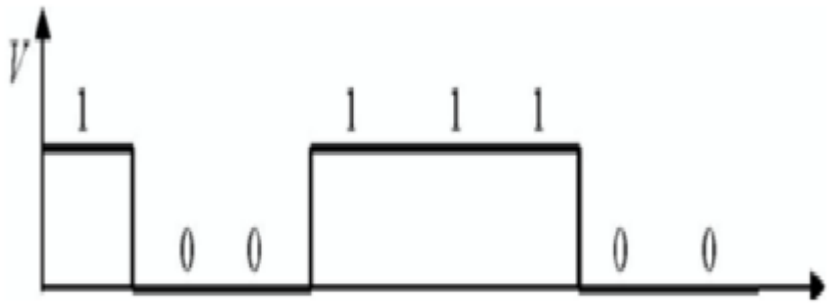
Las letras entre paréntesis que siguen a F forman una lista de las variables en el orden tomado cuando el minitérmino se convierte en un término AND.

MAXTÉRMINO (Mi): término suma en el que aparecen todas las variables, ya sean complementadas o sin complementar.
FÓRMULA CANÓNICA CONJUNTIVA O DE MAXTÉRMINOS: producto de maxtérminos (Producto de sumas).
Dada la lista completa de maxtérminos y asignando 1¶s y 0¶s arbitrariamente a las variables, siempre hay un y sólo un maxtérmino que toma el valor 0. Un maxtérmino es un término suma que es 0 exactamente en una línea de la tabla de verdad. La fórmula compuesta por todos los maxtérminos será idénticamente 0. Cada fórmula puede expresarse como producto de maxtérminos. Y es única. NOTACIÓN: Un maxtérmino se designa por ³Mi´ siendo i el número decimal correspondiente de la tabla de verdad. En la suma, el 1 se asocia a la variable complementada y el 0 a la variable sin complementar.

EJEMPLO:C B A F(C,B,A)
0 0 0 1
0 0 1 0
0 1 0 1
0 1 1 1
1 0 0 0
1 0 1 0
1 1 0 0
1 1 1 1
F(C,B,A) = M1 · M4 · M5 · M6 = P M(1,4,5,6)
F(C,B,A) = (C+B+A') · (C'+B+A) · (C'+B+A') · (C'+B'+A)
O bien:
F(C,B,A) = (C+B+A) · (C+B+A) · (C+B+A) · (C+B+A)

4.3.2 Representación de expresiones booleanas con circuitos lógicos

Un *circuito lógico* es un dispositivo que tienen una o más entradas y exactamente una salida. En cada instante cada entrada tiene un valor, 0 o 1; estos datos son procesados por el circuito para dar un valor en su salida, 0 o 1.
Los valores 0 y 1 pueden representar ciertas situaciones físicas como, por ejemplo, un voltaje nulo y no nulo en un conductor.



Los circuitos lógicos se construyen a partir de ciertos circuitos elementales denominados compuertas lógicas, entre las cuales diferenciaremos:
•Compuertas lógicas básicas: OR, AND, NOT.
•Compuertas lógicas derivadas: NOR, NAND.

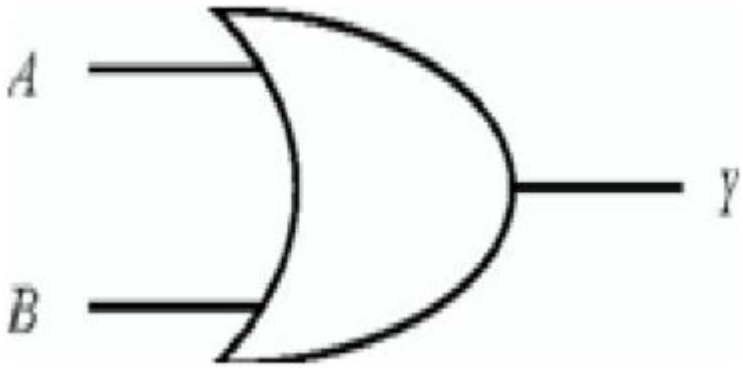
Compuerta OR
En una *compuerta OR* con entradas *A* y *B*, la

salida *Y*resulta:

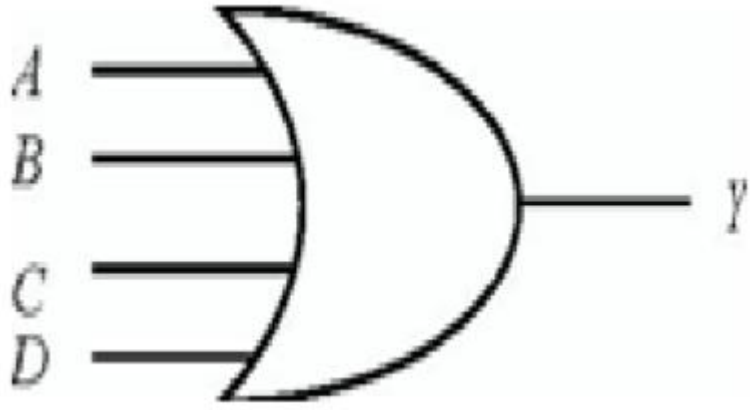
Y= A+ B
donde la suma se define por la siguiente tabla:

A	***B***	***Y=A+B***
0	0	0
0	1	1
1	0	1
1	1	1

La compuerta OR se representa del siguiente modo:



La compuerta OR también puede tener más de dos entradas:

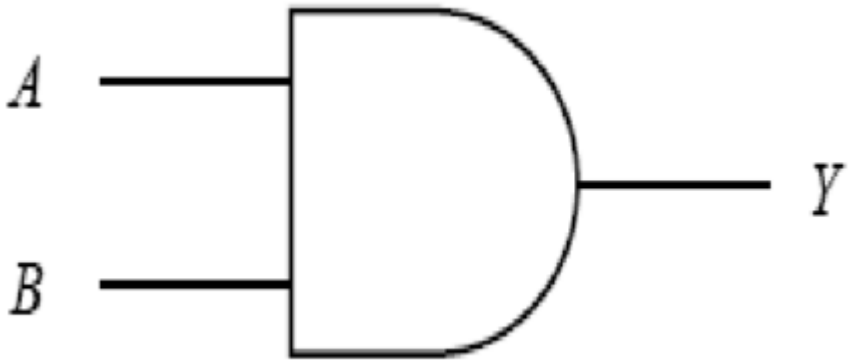


donde la salida *Y=A+B+C+D* puede obtenerse asociando los sumandos:
Y=A+B+C+D = (A+B)+(C+D) = ((A+B)+C)+D

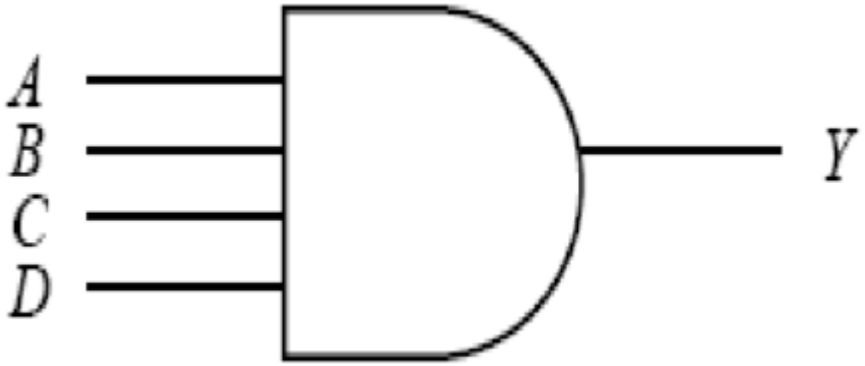
Compuerta AND
En una *compuerta AND* con entradas *A* y *B*, la salida *Y* resulta:
Y =A**B*
donde el producto se define por la siguiente tabla:

A	***B***	***Y=A*B***
0	0	0
0	1	0
1	0	0
1	1	1

La compuerta AND se representa del siguiente modo:



La compuerta AND también puede tener más de dos entradas:

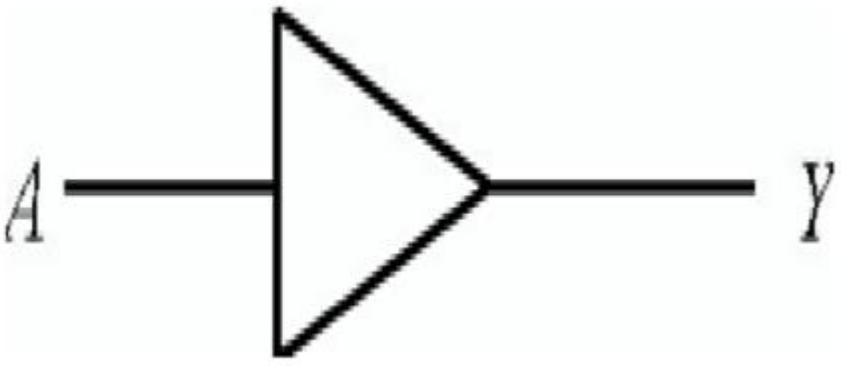


donde la salida *Y=A*B*C*D* puede obtenerse asociando los factores:
*Y =A*B*C*D = (A*B)*(C*D)= ((A*B)*C)*D*

Compuerta NOT
En una *compuerta NOT* con entrada *A*, la salida *Y* resulta:
Y =A
donde el complemento se define por la siguiente tabla:

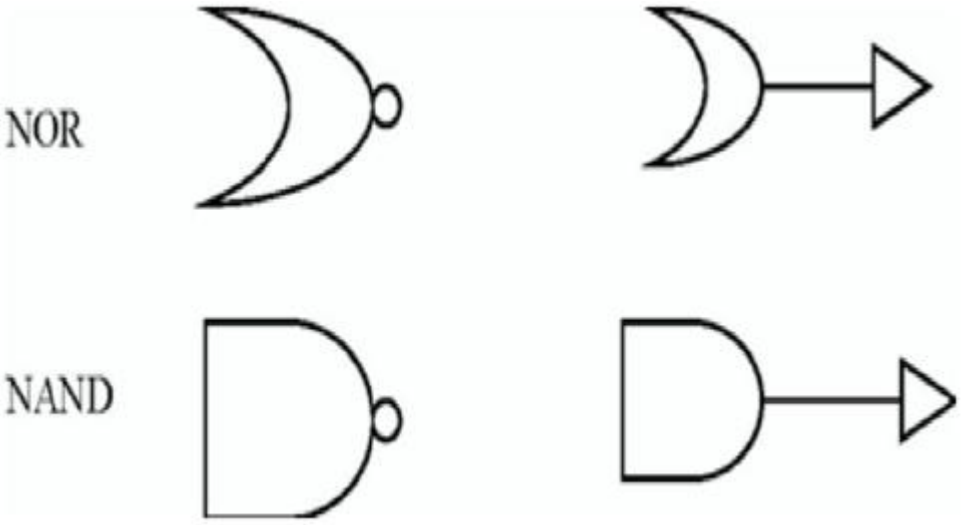
A	*Y*
1	0
0	1

La compuerta NOT se representa del siguiente modo:



Compuertas NOR y NAND

Las compuertas NOR y NAND no son básicas. Una *compuerta NOR* equivale a una compuerta OR seguida de una compuerta NOT. Una *compuerta NAND* equivale a una compuerta AND seguida de una compuerta NOT.



Por lo tanto, cuando las entradas son *A* y *B*, las salidas de estas compuertas resultan:

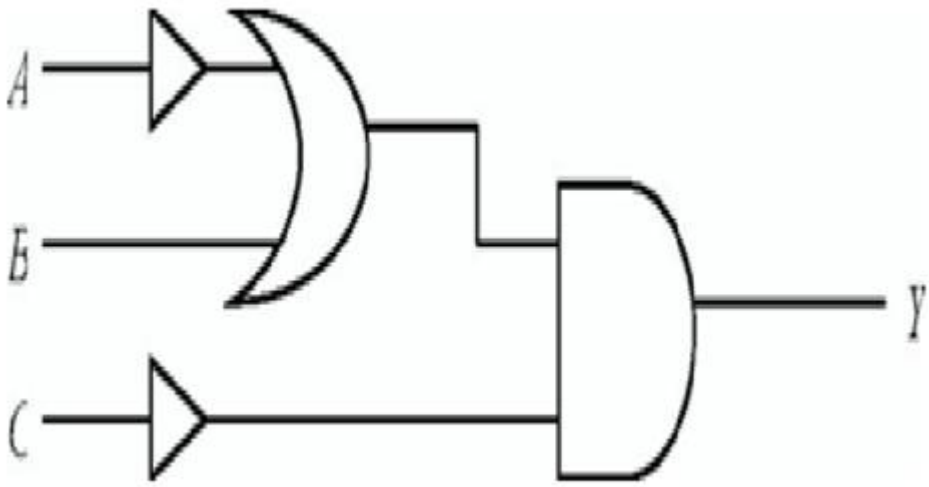
- NOR: Y = A+B
- NAND: Y = A*B

Circuitos lógicos

Los circuitos lógicos se forman combinando compuertas lógicas. La salida de un circuito lógico se obtiene combinando las tablas correspondientes a sus compuertas componentes.

Por ejemplo:

$$Y = (A+B)* C$$



Es fácil notar que las tablas correspondientes a las compuertas OR, AND y NOT son respectivamente idénticas a las tablas de verdad de la disyunción, la conjunción y la negación en la lógica de enunciados, donde sólo se ha cambiado V y F por 0 y 1. Por lo tanto, los circuitos lógicos, de los cuales tales compuertas son elementos, forman un *álgebra de Boole* al igual que los enunciados de la lógica de enunciados.

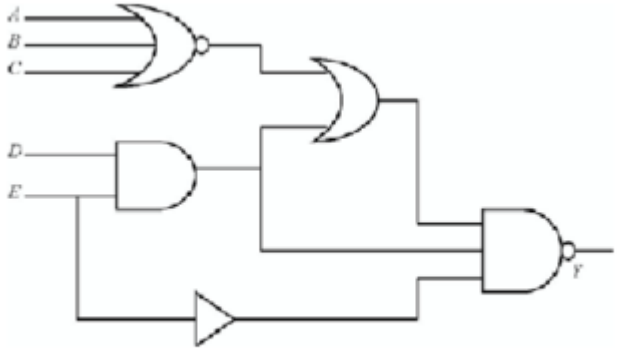
Adoptaremos, entonces, aquí las mismas convenciones adoptadas en el caso del álgebra de Boole:

- Omitimos el símbolo *, usándose en su lugar la yuxtaposición de variables.
- Establecemos que + es más fuerte que * y * es más fuerte que

Puesto que tanto el álgebra de Boole es la estructura algebraica tanto de los circuitos como de la lógica de enunciados, la salida de un circuito lógico también puede expresarse en el lenguaje de la lógica de enunciados. Por ejemplo, la salida del circuito anterior resulta:

(A+B)* C (~p v q) ^ ~r

Ejemplo Y = ((A+B+C)+DE)DEE



= ((A+B+C)+DE)DEE

~((~(p v q v r) v (s ^ t)) ^ s ^ t ^ ~t)