Introducción al Diseño Lógico (E0301)

Ingeniería en Computación

Gerardo E. Sager

Clase 5 curso 2023

Introducción al Diseño Lógico

- *Temas que se tratan*
  - Variables booleanas o lógicas
  - Tablas de verdad. (TdV)
  - Ejemplos de TdV para compuertas AND, NAND, OR, y NOR, y el circuito inversor NOT.
  - Expresiones booleanas para compuertas lógicas.
  - Teoremas de Boole y DeMorgan para simplificar expresiones lógicas.

Variables booleanas o lógicas

- **Constantes y Variables Booleanas:**
  - **Solamente puede adoptar dos valores**
  - **Muchas veces se utilizan distintas palabras como sinónimos de estos valores. En la tabla se muestran los más usuales**

0 lógico	1 lógico
Falso	Verdadero
Apagado	Encendido
Bajo	Alto
No	Si
Interruptor abierto	Interruptor cerrado

Operaciones lógicas

- Operaciones básicas
  - Conjunción: AND, Y, intersección. *(A • B)*
  - Disyunción inclusiva : OR, Ó, unión. *(A+B)*
  - Negación: Not, No, complemento. (\A) o también $\overline{A}$
- Minterm o minitérmino
  - Expresion con una o más variables no repetidas, directas o negadas, relacionadas entre sí mediante conjunción
  - Ejemplos: $\overline{A}.B.C$ $\quad x.\overline{y}.z$
- Maxterm o maxitérmino
  - Expresion con una o más variables no repetidas, directas o negadas, relacionadas entre sí mediante disyunción inclusiva
  - Ejemplos: $(\overline{A}+B+C)$ $\quad (x+\overline{y}+z)$

Tablas de Verdad (TdV)

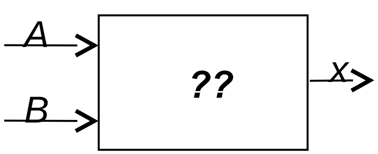
- La Tabla de Verdad describe la relación entre entradas y salidas.
- Los valores posibles de las entradas y salidas son valores binarios (0 o 1).
- El número de entradas define la cantidad de combinaciones posibles a considerar
- N entradas → 2^N combinaciones
- Deben listarse todas las combinaciones posibles de entradas y los valores que toma la salida para cada combinación.
- Esto puede interpretarse como que la TdV, define una *función lógica* cuyo dominio son las combinaciones posibles de entradas y su imagen son los valores que adopta la salida en cada caso

Tablas de Verdad 1, 2, 3 y 4 variables

A	x
0	0
1	1

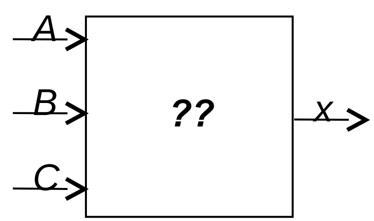
1 entrada → 2 combinaciones

A	B	x
0	0	0
0	1	0
1	0	1
1	1	0



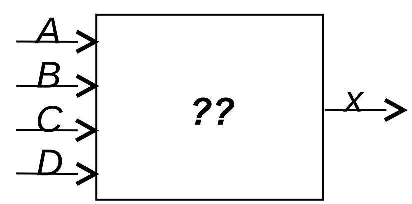
2 entradas → 4 combinaciones

A	B	C	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



3 entradas → 8 combinaciones

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



4 **entradas → 16 combinaciones**

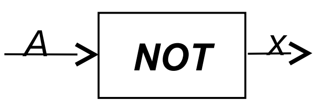
Tablas de verdad de funciones lógicas

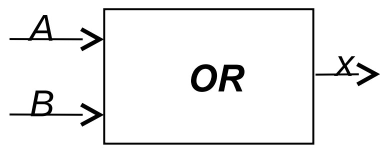
A	x
0	1
1	0

A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0





- De una variable: NOT
  - Invierte el valor lógico de la entrada
- De dos variables OR, AND, XOR
  - OR da una salida Verdadera, cuando cualquiera de sus entradas es verdadera
  - AND da una salida Verdadera cuando todas sus entradas son Verdaderas
  - XOR da una salida Verdadera si sus entradas son distintas.

Funciones y compuertas

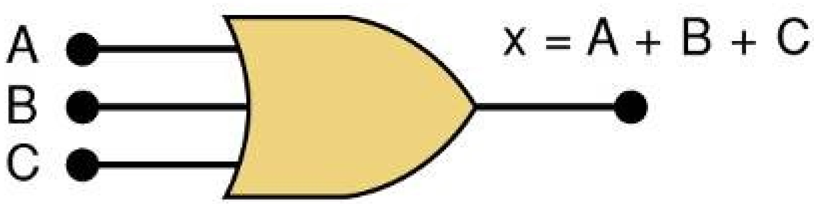
- Las primeras implementaciones de funciones lógicas digitales recibieron el nombre de compuertas lógicas (*logical gates*).
- La Compuerta NOT (También llamada INVERSOR) tiene una sola entrada.
  - NOT: la salida es opuesta a la entrada.
- Los tipos más comunes que poseen más de una entrada son OR, NOR, AND, NAND, XOR y XNOR.
  - OR: la salida es verdadera si cualquier entrada es verdadera.
  - NOR: la salida es la opuesta a la de la OR.
  - AND: la salida es verdadera si todas las entradas son verdaderas.
  - NAND: la salida es la opuesta a la AND.
  - XOR: (Or Exclusiva o disyunción exclusiva) la salida es verdadera si un número impar de entradas es verdadera.
  - XNOR: XOR Negada, la salida es verdadera si un número par de entradas es verdadera.

Funciones y compuertas

- Las funciones lógicas pueden denotarse de distintas maneras, hemos visto la TdV y los nombres de algunas de ellas. Tambíen pueden representarse de manera gráfica con los símbolos de compuerta que veremos más adelante o con una notación de tipo algebraico que permite operar matemáticamente (algebra de boole)
- Función NOT si la entrada es x la salida es $\overline{x}$. Como es difícil escribir la barra sobre la variable, se suele usar otra notación $\overline{x}=\backslash\mathrm{x}$.
- Función AND: si las entradas son A y B, y la salida es x, se escribe como si fuera un producto $x=A.B=AB$
- Función OR: si las entradas son A y B, y la salida es x, se escribe como si fuera una suma: $x=A+B$
- XOR: si las entradas son A y B, y la salida es x, se escribe como se muestra a continuación: $x=A\oplus B$
- NAND: si las entradas son A y B, y la salida es x, se escribe como se muestra a continuación: $x=\overline{AB}=\backslash\,(\mathrm{AB})$
- NOR: si las entradas son A y B, y la salida es x, se escribe como se muestra a continuación: $x=\overline{A+B}=\backslash\,(\mathrm{A+B})$
- XNOR: si las entradas son A y B, y la salida es x, se escribe como se muestra a continuación: $x=\overline{A\oplus B}=\backslash\left(A\oplus B\right)$

Compuerta OR

Compuerta OR de tres entradas, Función y Tabla de Verdad

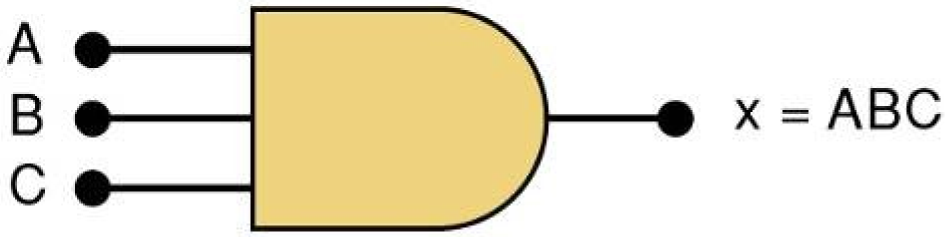


A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Compuerta AND

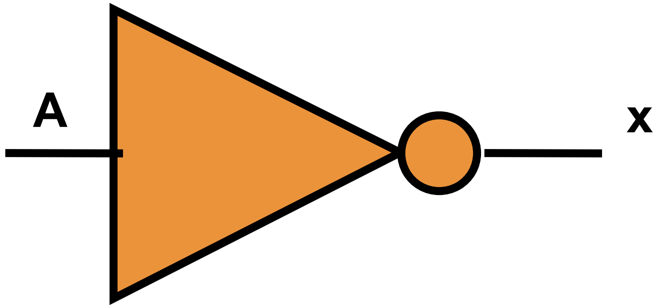
Compuerta AND de tres entradas, Función y Tabla de Verdad

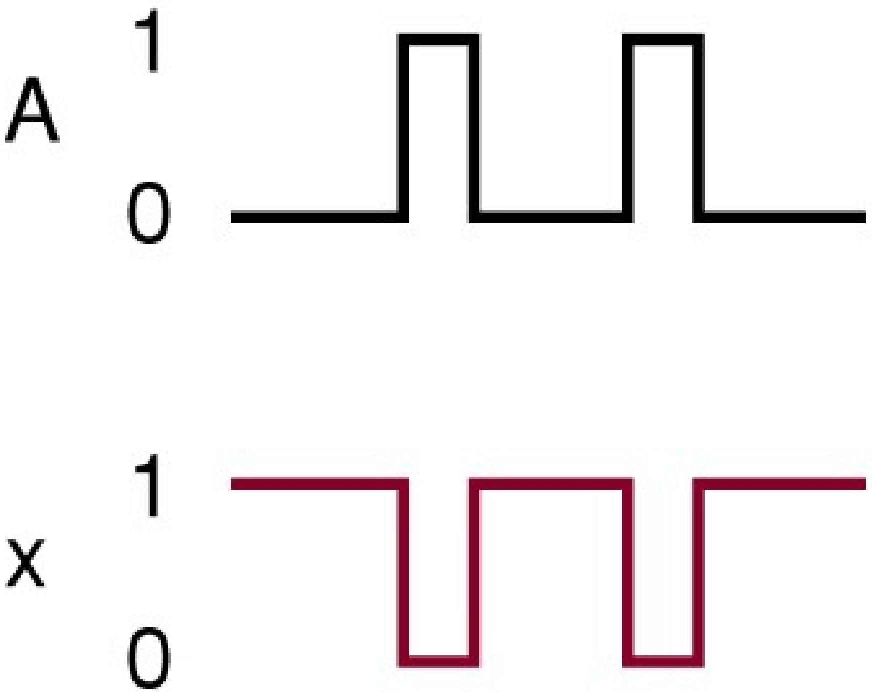
A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Compuerta Not o Inversor

El inversor genera una salida que es la opuesta de la entrada para todos los puntos de la señal de entrada. La operación se llama también NOT o complementación.





Siempre que la entrada sea = 0, salida = 1,

y viceversa.

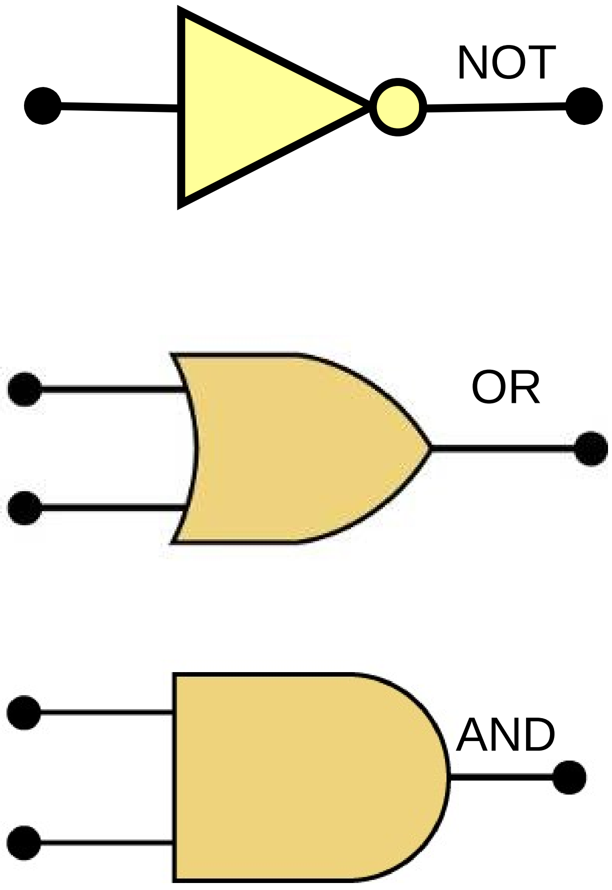
Equivalencia entre compuertas y operaciones booleanas

Sumario de reglas para OR, AND y NOT

OR	*AND*	*NOT*
$0+0=0$	$0 \cdot 0=0$	$\overline{0}=1$
$0+1=1$	$0 \cdot 1=0$	$\overline{1}=0$
$1+0=1$	$1 \cdot 0=0$	
$1+1=1$	$1 \cdot 1=1$	

Cualquier operación booleana puede escribirse mediante estas tres operaciones elementales

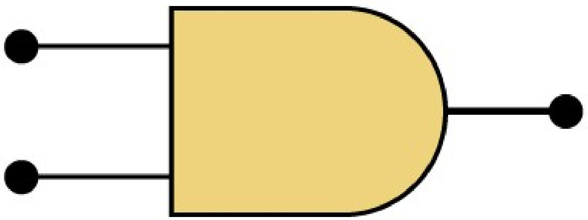
Equivalencia entre compuertas y operaciones booleanas

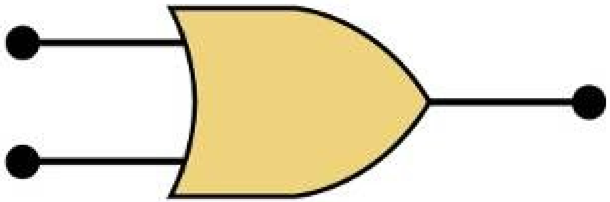


Como estas compuertas implementan las operaciones elementales, con ellas se puede implementar cualquier operación booleana.

Equivalencia entre compuertas y operaciones booleanas

El símbolo AND en un circuito lógico dice que la salida va a ir a HIGH *sólo* si *todas* las entradas son HIGH





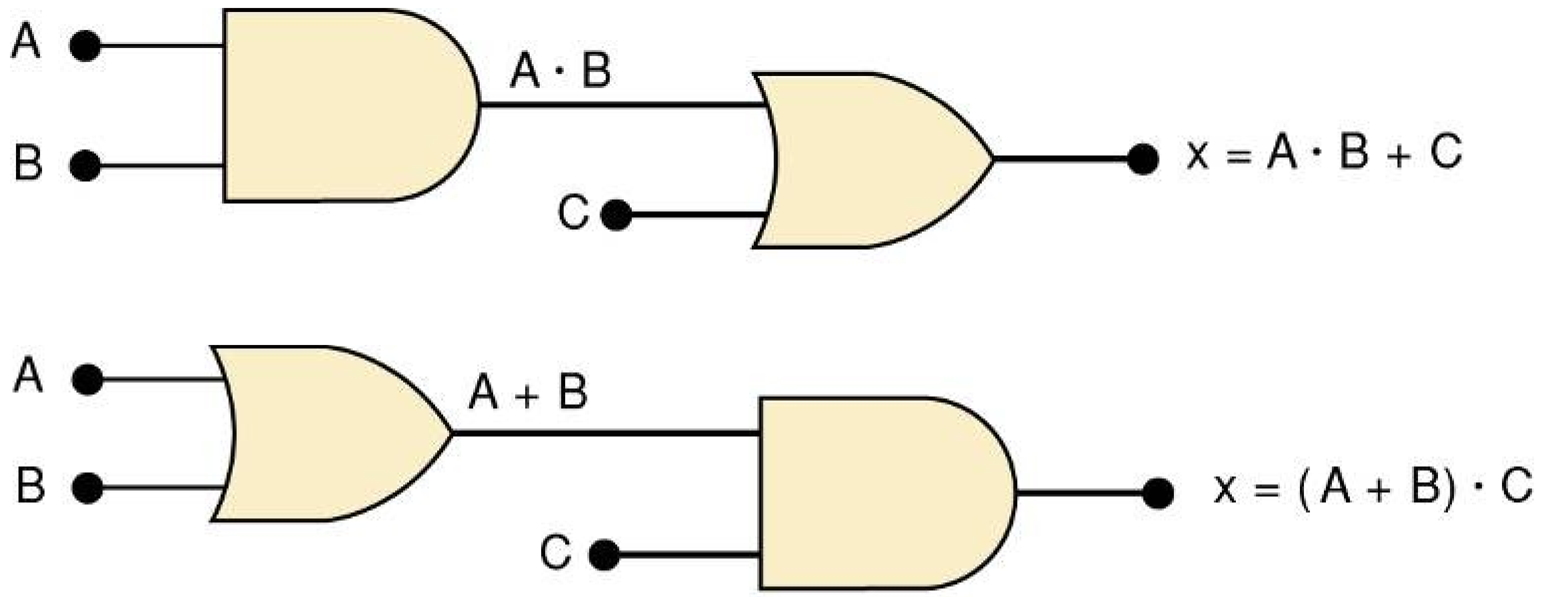
El símbolo OR dice que la salida va a ir a HIGH cuando *cualquier* entrada sea HIGH

Operaciones y Precedencia

- En una expresión booleana, se analiza de izquierda a derecha.
- La operación AND (•) tiene precedencia sobre la operación OR (+).
- Como en la multiplicación, la operación AND puede tomarse como implícita ABC=A•B•C
- Si se quiere explicitar o modificar la precedencia puede utilizarse paréntesis:
  - A+BC=A+(BC)
  - x=(A+B)C es distinto que x=A+BC
- La operación de negación tiene la mayor precedencia \AB = (\A)•B

Ejemplos: Análisis

- Determinar cual es la función lógica que implementa un circuito dado



  - También se procede de izquierda a derecha y se va operando con los resultados parciales como se muestra en las figuras.

Ejemplos: Análisis

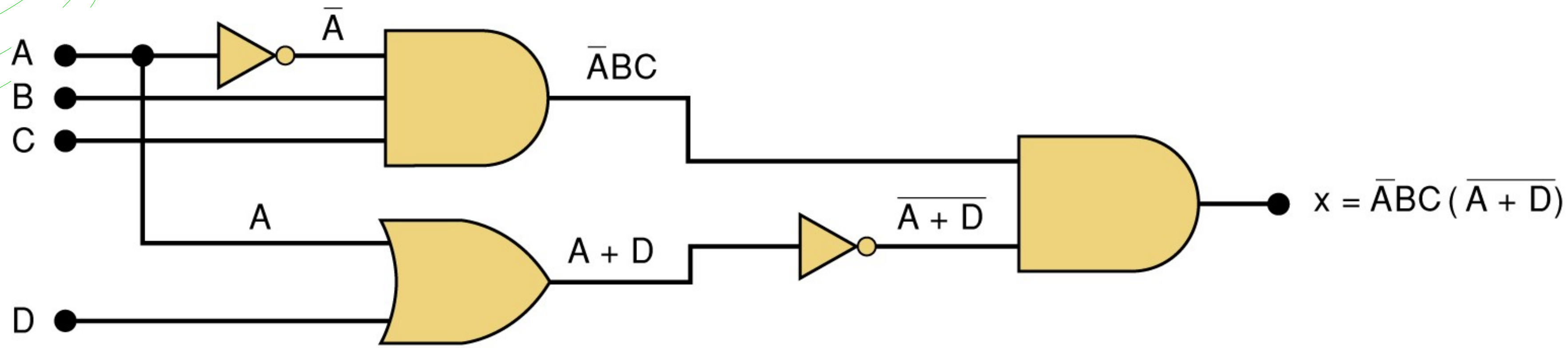
- Determinaremos las tablas de verdad de ambos circuitos.

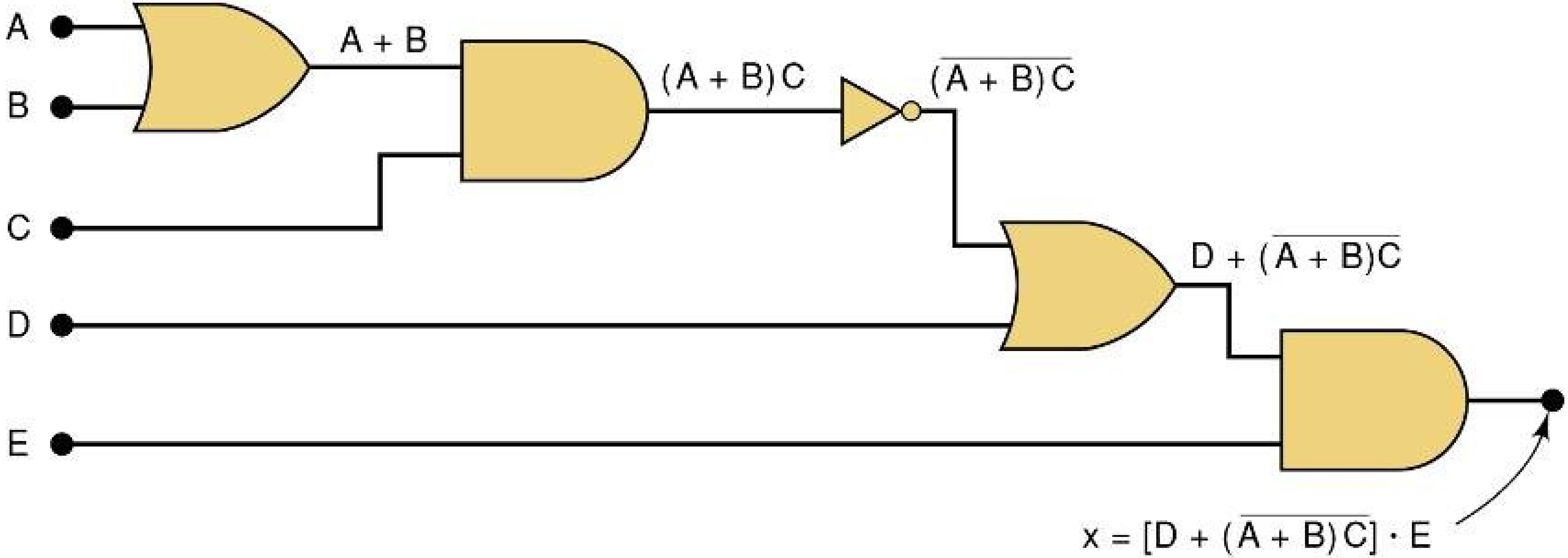
x=A•B+C			
A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x=(A+B)•C			
A	B	C	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Se ve claramente que son distintas

Ejemplos. Se animan a hacer la TdV?





Evaluar expresiones booleanas

- **Reglas para la evaluación de expresiones booleanas:**
  - **Realizar todas las inversiones de términos simples.**
  - **Realizar todas las operaciones dentro de paréntesis.**
  - **Realizar las operaciones AND antes que las OR a menos que los paréntesis indiquen otra cosa.**
  - **Si una expresión tiene una barra sobre ella, realizar todas las operaciones dentro de la expresión y luego invertir el resultado.**

Teoremas de Boole (Propiedades)

Una Variable

1. $x.0 = 0$
2. $x.1 = x$
3. $x.x = x$
4. $x.\overline{x} = 0$
5. $x+0 = x$
6. $x+1 = 1$
7. $x+x = x$
8. $x+\overline{x} = 1$

Dos Variables

9. $x + y = y + x$ (conmutativa)
10. $x.y = y.x$ (conmutativa)
11. $x + (y + z) = (x + y) + z = x + y + z$ (asociativa)
12. $x. (y . z) = (x . y) .z = x.y.z = xyz$ (asociativa)
13. a) $x (y + z) = x y + x z$ (distributiva)

    b) $(w + x) (y + z) = wy + wz + xy + xz$ (distributiva)
14. $x + xy = x$
15. a) $x + \overline{x} y = x + y$ b) $\overline{x} + xy = \overline{x} + y$

Teoremas de De Morgan

$$\overline{x+y}=\overline{x}\,.\,\overline{y}$$

$$\overline{x\,.\,y}=\overline{x}+\overline{y}$$

Son muy útiles para simplificar expresiones booleanas.
Puede demostrarse fácilmente que además:

$$\overline{x+y+z}=\overline{x}\,.\,\overline{y}\,.\,\overline{z}$$

$$\overline{x\,.\,y\,.\,z}=\overline{x}+\overline{y}+\overline{z}$$

Ejemplos:

$$z=\overline{\left(\overline{A}+C\right)\left(B+\overline{D}\right)}=\overline{\left(\overline{A}+C\right)}+\overline{\left(B+\overline{D}\right)}=\overline{\overline{A}}\,\overline{C}+\overline{B}\,\overline{\overline{D}}=A\,\overline{C}+\overline{B}\,D$$

$$z=\overline{A+\overline{B}+C}=\overline{A}\,\overline{\overline{B}}\,\overline{C}=\overline{A}\,B\,\overline{C}$$

$$z=\overline{\left(A+BC\right)\left(D+EF\right)}=\overline{\left(A+BC\right)}+\overline{\left(D+EF\right)}=\overline{A}\,\overline{\left(BC\right)}+\overline{D}\,\overline{\left(EF\right)}$$

$$=\overline{A}\left(\overline{B}+\overline{C}\right)+\overline{D}\left(\overline{E}+\overline{F}\right)=\overline{A}\,\overline{B}+\overline{A}\,\overline{C}+\overline{D}\,\overline{E}+\overline{D}\,\overline{F}$$

Formas canónicas

- Se definen dos formas canónicas de expresión booleana
  - Conjunción de maxitérminos o “Producto de Sumas”
    - Ejemplo (A+\B)(\A+ B)
  - Disyunción de minitérminos o “Suma de Productos"
    - Ejemplo A\B + \AB
- Podemos llevar cualquier expresión booleana a una forma canónica, operando sobre ella mediante los teoremas de Boole y De Morgan.
- En los ejemplos de la transparencia anterior se han llevado las expresiones a la forma de una “suma de productos”.