**Tema 3: Análisis y síntesis de circuitos lógicos combinacionales**

**Fundamentos del álgebra de Boole**

* Conjunto A con dos operaciones binarias (+, \*), equivale a (OR, AND) que verifica los postulados siguientes:
  + Es un conjunto cerrado
  + Elementos neutros (a+0=a, a\*1=a)
  + Elemento complementario (a+ā=0, a\*ā=1)
  + Conmutatividad (a+b=b+a, a\*b=b\*a)
  + Asociatividad (a+[b+c]=[a+b]+c, a\*[b\*c]=[a\*b] \*c, )
  + Propiedad distributiva (a\*(b+c) = ab+ac)
* **Teoremas:**
  + a+a=a, a\*a=a
  + a+1=1, a\*0=0
  + a+ab=a, a(a+b)=a
  + a+āb=a+b, a(ā+b) = ab
  + ab+aЂ=a, (a+b)(a+Ђ)=a ¬
  + ¬(a+b) = ¬a\*¬b, ¬(a\*b) = ¬a+¬b
  + ab+aЂc = ab+ac
  + (a+b)(a+Ђ+c) = (a+b)(a+c)

**Funciones booleanas**

* Funciones de n variables que toman valor 0 o 1 y cuyas salidas toman también valores 0 o 1
* Se representan mediantes tablas de verdad o formas algebraicas
  + Forma algebraica: F(X2,X1,X0) = X2\*X0 + X2\*X1
* Función **incompletamente especificada:** Si para alguna entrada la salida es de valor desconocido o irrelevante.
  + Ejemplo: una función cuya entrada es la tirada de un dado de 6 caras requiere 3 variables, sin embargo, para el valor 1112 = 710, la salida es irrelevante.
* **Expresiones equivalentes:** Expresiones algebraicas que representan la misma función
* **Formas suma de productos (SOP)**: Expresión donde se realiza el OR de términos AND de la función.
  + **Mintérmino** (mi): número binario representado por los dígitos del término producto que contiene todas las variables de la función SOP
* **Forma POS:** AND de términos OR, producto de sumas.
  + **Maxtérmino** (Mi): Lo mismo aplicado a los sumandos de la función POS.
  + **¬mi = Mi , ¬Mi = mi**
* **Formas canónicas:**
  + Forma SOP o POS única para la función.
  + Cada término AND o OR incluye a todas las variables de la función una sola vez negadas o sin negar
  + Cada término AND o OR aparece una única vez en la expresión
* **Literales:** Cada una de las variables que aparece en un término suma o término producto
* Obtener función SOP a partir de tabla:

| a | b | c | f |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Forma SOP: f(a,b,c) = m0 + m2 = ∑m(0,2) = ¬a\*¬b\*¬c + ¬a+b+¬c

Donde hay 0s se escribe ¬a, donde hay 1 se escribe a.

Forma POS: f(a,b,c) = M1\*M3\*M4\*M5\*M6\*M7 = ∏M(1,3,4,5,6,7)

= (a+b+¬c)\*(a+¬b+¬c)........

Donde hay 0s se escribe a, donde hay 1 se escribe ¬a.

Donde hay una indiferencia, se puede expresar como mintérmino o maxtérmino.

**Simplificación de expresiones lógicas**

* Aplicar álgebra de Boole y leyes de De Morgan
  + Ejemplo: F(A,B,C) = B\*(A+C)+ABC = B\*(A+C)
* Analizar las adyacencias mediante diagramas de Karnaugh
  + Dos estados de entrada son adyacentes cuando entre ellos solo cambia una de las variables.

| b \ a | 0 | 1 |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 0 | 1 |

* + Ejemplo: tabla de Karnaugh de ab + a¬b:

Dos valores adyacentes se agrupan, resultando en que **ab + a¬b = a**

* Obtener función a partir de tabla de Karnaugh:

| ab \ c | 0 | 1 |
| --- | --- | --- |
| 00 | 0 | X |
| 01 | 0 | 1 |
| 11 | 0 | 1 |
| 10 | 0 | X |

Para obtener la fórmula SOP, nos fijamos en los implicantes[[1]](#footnote-0) de 1s:

f(a,b,c) = b\*¬c + a\*b

Para obtener la fórmula POS, nos fijamos en los implicantes de 0s y escribimos la inversa de la fórmula correspondiente.

f(a,b,c) = (b) \* (a + ¬c)

* En una tabla de Karnaugh, las indeterminaciones (representadas con X) se pueden considerar como 0s o 1s para poder expandir los implicantes todo lo posible.

**Puertas lógicas**

* Circuitos que realizan funciones lógicas básicas.

| Puerta | Símbolo | Puerta | Símbolo |
| --- | --- | --- | --- |
| NOT |  |  | |
| AND |  | NAND |  |
| OR |  | NOR |  |
| EXOR |  | NEXOR |  |

* XOR: a⊕b = ¬a\*b + a\*¬b (a ou b, pero non ambas).
  + ¬(a⊕b) = ¬a⊕b = a⊕¬b
  + a⊕0 = a
  + a\*(b⊕c) = (a\*b)⊕(a\*c)
* Las leyes de De Morgan demuestran que ¬(A+B) = ¬A\*¬B, por lo que una puerta NOR es equivalente a una AND con las entradas complementadas.
* Además, ¬(A\*B) = ¬A+¬B, por lo que una puerta NAND equivale a una OR con las entradas complementadas.

**Conjuntos completos de puertas**

* Aquel con el que se puede implementar cualquier función lógica.
  + AND y NOT
  + OR y NOT
  + NAND
  + NOR
* Con una puerta NAND, se puede crear cualquier otra puerta lógica:
  + AND: negando la NAND
  + OR: Conectando dos NAND a otra NAND
  + NOT: conectando las dos entradas de la NAND entre sí para crear una única
* Lo mismo con una NOR:
  + AND: Conectando dos NOR a otra NOR
  + OR: Negando la NOR
  + NOT: mismo método previo

1. Implicante: conjunto formado por celdas adyacentes

   Implicante primo: aquel que no está contenido en otro implicante

   **Implicante primo esencial:** aquel que además cubre un mintérmino que no puede cubrir ningún otro. [↑](#footnote-ref-0)