CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Arquitectura de ordenadores

3-2. Principios Básicos II

Ignacio Calles González ignacio.gonzalez@ext.live.u-tad.com
Tiago Manuel Louro Machado de Simas tiago.louro@u-tad.com
Francisco Javier García Algarra javier.algarra@u-tad.com
Carlos M. Vallez Fernández carlos.vallez@u-tad.com

2023-2024

Índice

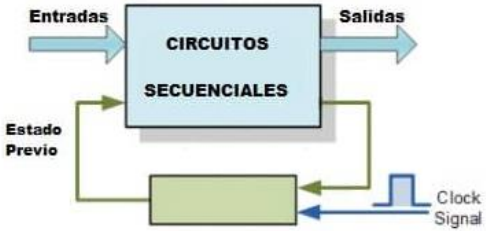
1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito data su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

1. Introducción a circuitos

Como hemos comentado en el apartado anterior, las puertas lógicas se agrupan para formar circuitos. Atendiendo a sus características, estos circuitos se pueden diferenciar en:

- **Circuitos combinacionales:** Aquellos en los que las salidas dependen tan sólo del estado actual de sus entradas. No dependen del estado anterior. No hay realimentación.
- **Circuitos secuenciales:** Aquellos en los que las salidas dependen del estado actual de sus entradas y del estado anterior.

En los siguientes capítulos nos centraremos en explicar cada uno de estos subtipos de circuitos.



Índice

1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito data su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

2. Circuitos combinaciones definición

- Un circuito combinacional **se compone de un conjunto de puertas interconectadas cuya salida depende del valor de las entradas en ese mismo instante.**
- El (ínfimo) retardo entre la aparición de las entradas y la aparición de las salidas es el causado por los retardos de las puertas lógicas del circuito.
- Se dice que un circuito combinacional consiste en n entradas por m salidas, y puede definirse mediante:
  - **Su tabla de verdad**: aúna las 2^n combinaciones posibles de las n entradas con todos los valores de las m salidas.
  - **Su símbolo**: el diagrama del circuito.
  - **Su función booleana**: la función que expresa los valores de la salida en función de las entradas.

Índice

1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito dada su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

3. Implementación de un circuito dada su tabla de verdad (I)

Según vayáis construyendo circuitos comprobareis que si no se sigue cierto orden y criterio puede quedarnos una maraña de conexiones y puertas difícilmente entendible.

Es por ello que en las siguientes diapositivas os explicamos una serie de pasos que si seguís harán que podáis generar circuitos fácilmente comprensibles.

Son los pasos a seguir tanto en las prácticas que hagáis como en los futuros exámenes.

3. Implementación de un circuito dada su tabla de verdad (II)

Paso 1: Comprobar si tenemos la función lógica. Si solo disponemos de la tabla de verdad ¿cómo se consigue su función lógica equivalente?

Se buscan en la tabla las combinaciones que dan F = 1.

Cada combinación se separa mediante una suma para escribir la **función booleana**.

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

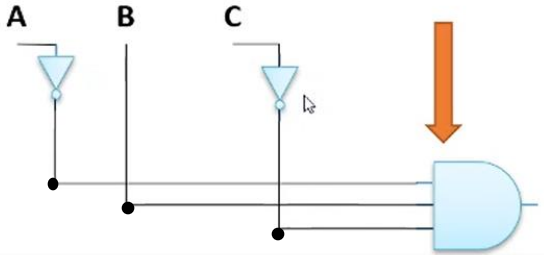
$$F = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$

3. Implementación de un circuito dada su tabla de verdad (III)

Paso 2: Una vez tenemos la función lógica, es importante tener para cada variable una línea vertical con su valor y su valor negado (salvo que veamos que no lo vamos a utilizar como es el caso de la variable B en el ejemplo). De esas líneas sacaremos líneas horizontales marcando con un punto de dónde salen para evitar confusiones.

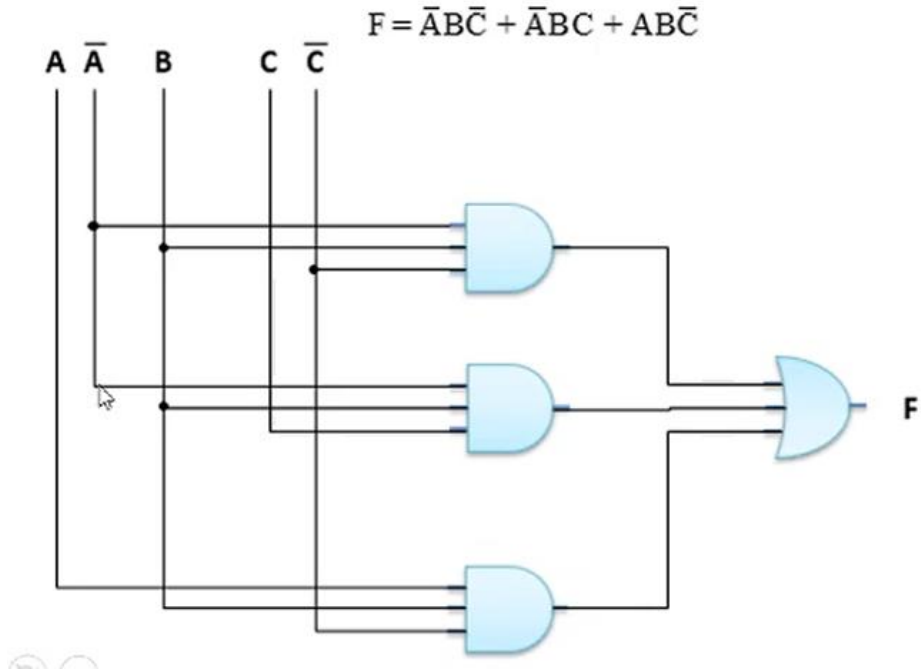
- Se diseña un **circuito** con puertas AND, OR y NOT que implemente dicha función:

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



3. Implementación de un circuito dada su tabla de verdad (III)

Paso 3: Hay que repetir esto para cada sumando de la función. Fijarse que los elementos de un mismo sumando se están multiplicando por tanto se unen con una puerta and mientras que los resultados de cada sumando se suman con los de otros sumando por tanto es una puerta OR



Índice

1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito data su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

4. Complejidad de un circuito. Formas de simplificación

- La **complejidad** de un circuito **viene dada por la cantidad de puertas y conexiones** que lo compongan.

- Es aconsejable simplificar la función booleana, para así simplificar el circuito que la codifica. En definitiva, lo que obtenemos es un circuito equivalente, pero más sencillo.

- Formas de simplificación:
  - 1. Simplificación Algebraica
  - 2. Mapas de Karnaugh

4.1 Simplificación Algebraica (I)

- Básicamente se aplican las propiedades de las operaciones lógicas AND y OR sucesivamente hasta reducir la función a su mínima expresión.

- Por ejemplo si nos dieran como función booleana de un circuito: f=NOT(A + A · B), si conocemos las propiedades del Álgebra de Boole rápidamente podemos ver aquí la propiedad de Absorción (A + A · B= A).

- Por tanto, nuestra función simplificada equivalente sería f= NOT(A). Vamos a comprobarlo con las tablas de verdad:

4.1 Simplificación Algebraica (II)

A	B	S1=A · B	S2= A+ S1	NOT(S2)	NOT (A)
0	0	0	0	1	1
0	1	0	0	1	1
1	0	0	1	0	0
1	1	1	1	0	0

Vemos que si vamos calculando cada parte de la expresión paso a paso al final llegamos a los mismos valores que si hiciésemos NOT (A) directamente.

4.2 Mapas de Karnaugh (I)

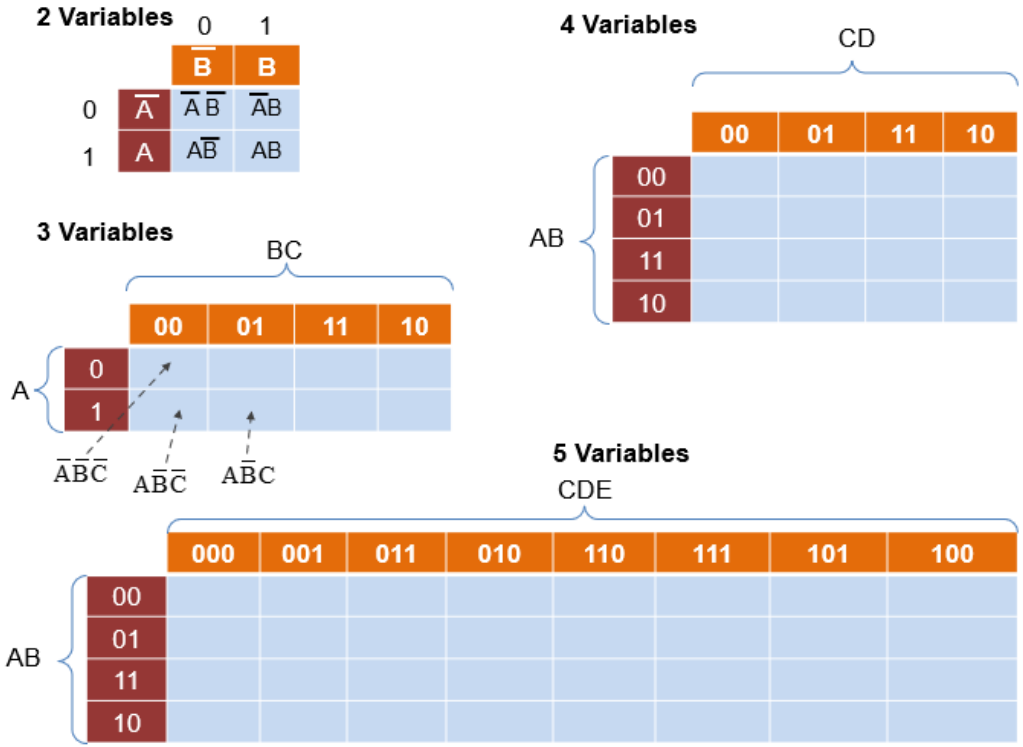
- Los mapas de Karnaugh aportan un método sistemático (una serie de pasos) que nos permiten realizar la simplificación de una función booleana.

- Un mapa de Karnaugh es una matriz o tabla donde cada celda representa una combinación de los valores binarios de cada variable de entrada de nuestra función Booleana.

4.2 Mapas de Karnaugh (II)

- Los mapas de Karnaugh se pueden usar con dos o más variables. A continuación, se muestran algunos esquemas con 2, 3, 4 y 5 variables:

Nota Importante: Hay que fijarse que cada celda difiere de la contigua en un solo valor (solo cambia el valor de una variable). Es decir, en el ejemplo de las 3 variables el orden de los valores de BC es 00, **01, 11** y 10. No se puede poner 00, **01, 10**, 11 porque entonces entre la 2ª columna y la 3ª estarían cambiando los valores de B y C.



4.2 Mapas de Karnaugh (III)

- El número total de celdas de un mapa de Karnaugh es igual al número total de combinaciones posibles de las variables de entrada. Así por ejemplo si tenemos 3 variables tendremos $2^3 = 8$ celdas. Por extensión, para n entradas tendremos 2^n celdas.

- Llegados a este punto, conociendo el número de variables que tiene la fórmula booleana o la tabla que nos dan, debemos ser capaces de construir la estructura de la Tabla de Karnaugh. En el siguiente apartado vamos a centrarnos en cómo rellenarla con valores y en aprender los pasos necesarios para poder simplificar la fórmula booleana

4.3 Mapas de Karnaugh. Método sistemático

- Primeramente debemos aclarar que la simplificación de una fórmula booleana por mapas de Karnaugh se puede realizar de dos formas:
    - Minimización de una suma de productos
    - Minimización de un producto de sumas

4.3.1 Minimización de suma de productos

- Diremos que nuestra fórmula booleana es una suma de productos si cada uno de sus términos están formados exclusivamente por productos y éstos a su vez se relacionan entre ellos mediante la operación de suma.
- La suma de productos se conoce como SOP (por su abreviación en inglés de “Sum of products”). Un par de ejemplos serían:

$$f1 = A + (B \cdot C)$$

$$f2 = (A \cdot B \cdot C) + (C \cdot NOT(D))$$

4.3.1 Minimización de suma de productos

Las funciones **SOP** presentan la siguiente **forma de codificar las variables**: Si tenemos **una variable** v , su valor **se corresponderá con 1** y **si la tenemos negada (** $\overline{v}$ **) se corresponderá con 0**

Para describir los pasos a realizar vamos a partir de una función Booleana (como bien sabemos a estas alturas, sería similar si nos dieran la tabla de verdad asociada a esa expresión Boolena).

4.3.1 Minimización de suma de productos

Paso 1: Verificamos que la expresión que nos han dado es una suma de productos (SOP) y montamos el esqueleto del mapa de Karnaugh. Supongamos que nos han dado la función:

F= (NOT(A) · B · NOT(C)) + (NOT(A) · B · C) + (A · B · NOT(C))

Podemos escribirla como:

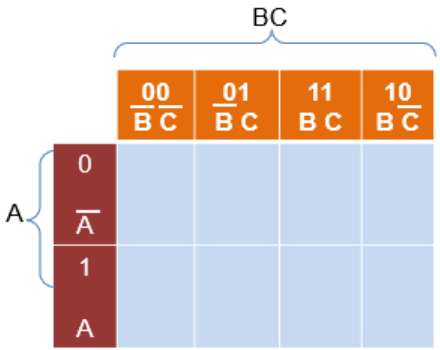
$$F= \bar{A}\, B\, \bar{C} + \bar{A}\, B\, C + A\, B\, \bar{C}$$

4.3.1 Minimización de suma de productos

Vemos que tiene 3 variables (A, B y C) por tanto el mapa de Karnaugh tendrá $2^3 = 8$ celdas.

Comprobamos que se tarta de una SOP por tanto las variables valen 1 y las negadas 0.

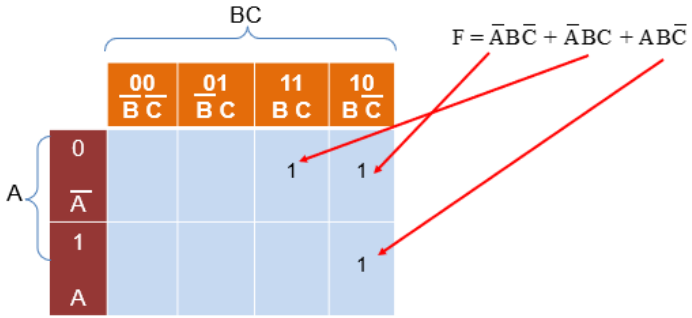
El esqueleto será:



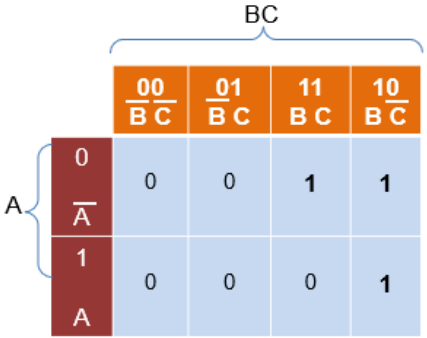
Es **muy importante** tener en cuenta que estamos en una suma de productos y que **por tanto la variable negada toma el valor cero mientras que la normal toma el valor 1**.

También hay que recordar (muy importante porque luego se olvida y da problemas) que **entre una celda y su adyacente solo cambia el valor de una variable** como vimos en el apartado anterior.

4.3.1 Minimización de suma de productos



Paso 2: Se coloca, (en la celda correspondiente) un 1 en el mapa de Karnaugh por cada término de la suma de productos.

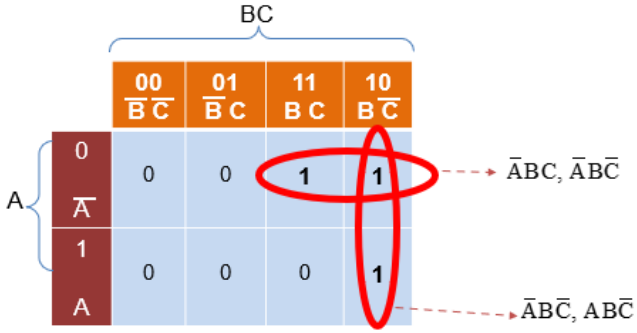


Se pueden colocar ceros en las celdas que no tengan un 1 tras completar el paso 2 o dejarlas así vacías:

4.3.1 Minimización de suma de productos

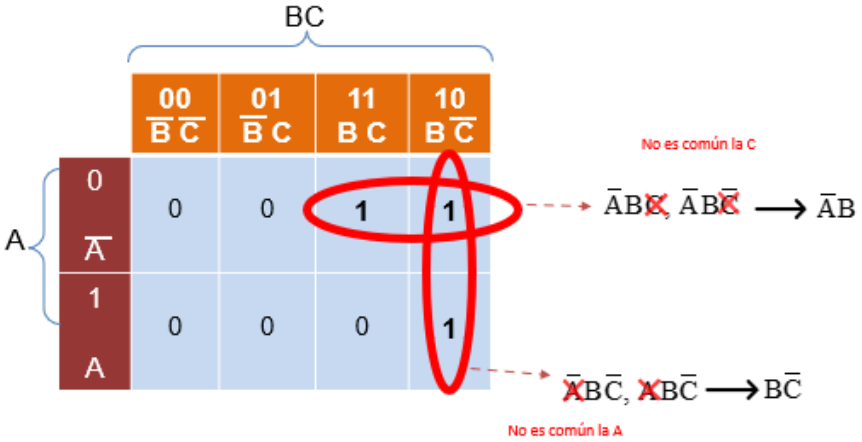
Paso 3: Una vez construido el mapa, se agrupan las celdas con valor 1 adyacentes, formando grupos cuyo tamaño sea potencia de 2 (2n): 2, 4, 8, 16…

Debemos entender que dos casillas adyacentes representan dos combinaciones que sólo difieren en el valor de una única variable.



4.3.1 Minimización de suma de productos

- **Paso 3 continuación:** Las dos combinaciones se pueden fundir en una, eliminando dicha variable.



Por lo tanto la función Booleana simplificada equivalente a la dada sería

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

4.3.1 Minimización de suma de productos

- **Paso 3:** I**MPORTANTE** a la hora de realizar las agrupaciones de valores 1 hay que tener en cuanta que:
  - Las agrupaciones tienen que contener una cantidad de “1” que sean potencias de 2. Es decir 2 números “1”, 4 números “1” etc. No se permiten grupos de 3 números 1, o de 5 etc.
  - Al final todos los 1 tienen que estar al menos en un grupo
  - Cuantos más números “1” Junte una agrupación, más simplificada nos quedará la función booleana.
  - Puede haber agrupaciones que solo tengan un 1 porque no tiene otro adyacentes con los cuales agruparlos.
  - Hay que entender la “adyacencia” de una forma amplia. Eso implica que por ejemplo la ultima celda de una fila es adyacente con la primera celda de la misma fila, como si la matriz estuviera doblada y formase un cilindro. Se muestra a continuación una imagen con las simplificaciones válidas en un mapa de Karnaugh:

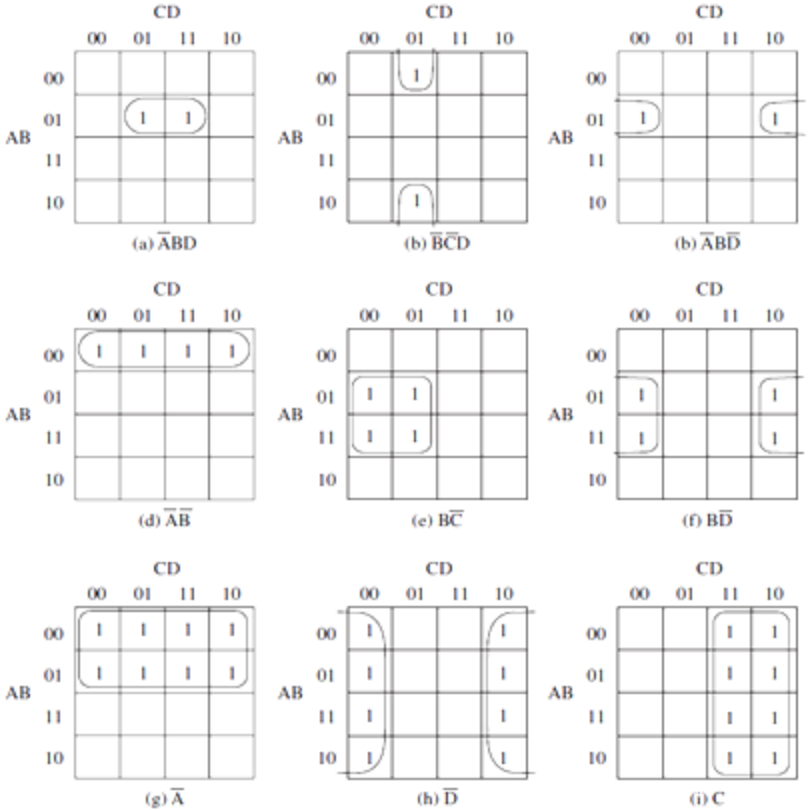
4.3.1 Minimización de suma de productos

- **Paso 3:**

La explicación matemática de por qué hacemos y podemos hacer estas simplificaciones se encuentra en las propiedades booleanas antes vistas:

A+ A = A

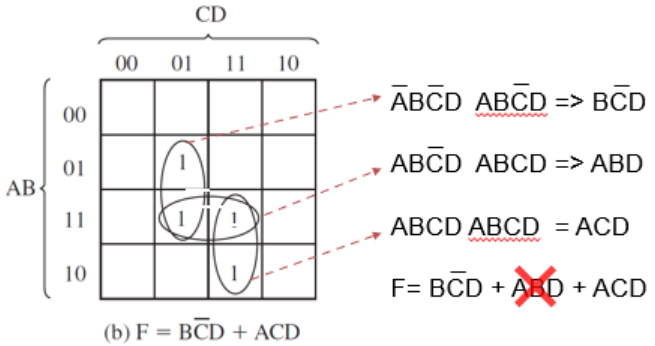
A + NOT(A) = 1



4.3.1 Minimización de suma de productos

- **Paso 4:** Al terminar de hacer los pasos anteriores, si tenemos conjuntos solapados se deben eliminar para tener una expresión más simplificada.

  Tomemos otro ejemplo. Imaginemos que nuestro mapa de Karnaugh queda así:



(b) $F = B\bar{C}D + ACD$

Acabamos de ver cómo simplificar una función booleana de tipo SOP mediante mapas de Karnaugh. Si nos dieran de partida una tabla de verdad, deberíamos transformar dicha tabla en su función Booleana y posteriormente aplicar los mismos pasos anteriormente comentados.

4.3.2 Minimización de producto de sumas

- Diremos que nuestra **fórmula booleana es un producto de sumas si cada uno de sus términos están formados exclusivamente por sumas y éstos a su vez se relacionan entre ellos mediante la operación de producto**. El producto de sumas se conoce como POS (por su abreviación en inglés de “Product of sums ”). Un par de ejemplos serían:

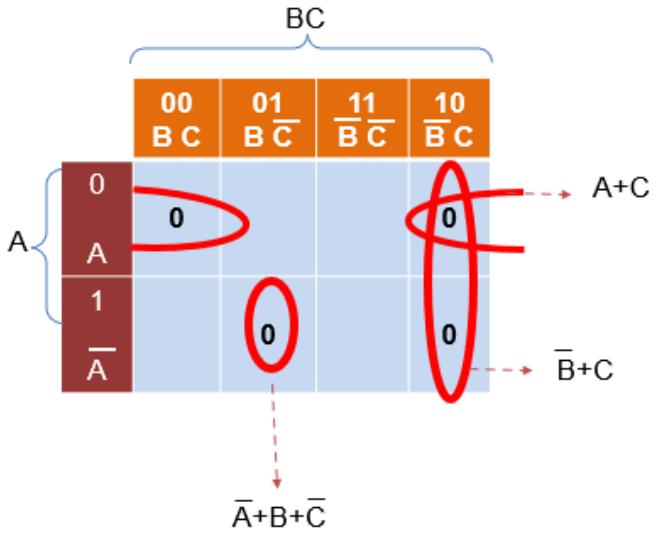
  $$f1 = (A + B) \cdot (A + C)$$
  $$f2 = (A + B + NOT(C)) \cdot (NOT(A) + B)$$

- Las funciones **POS presentan** la siguiente forma de codificar las variables: Si tenemos **una variable v , su valor se corresponderá con 0** y si la tenemos negada **(NOT(v)) se corresponderá con 1** (**justo al revés que las SOP**)
- Los pasos en POS son los mismos que en el caso de SOP teniendo en cuenta la correspondencia entre valores de variables y ceros y uno y además que ahora agruparemos ceros y no unos.

4.3.2 Minimización de producto de sumas

Ejemplo de problema resuelto:

$$F= (A+B+C) \cdot (A+\overline{B}+C) \cdot (\overline{A}+\overline{B}+C) \cdot (\overline{A}+B+\overline{C})$$
$$0\ 0\ 0 \quad 0\ 1\ 0 \quad 1\ 1\ 0 \quad 1\ 0\ 1$$



En POS:
A= 0
$\overline{A}$= 1

En los términos ponemos 0 en vez de 1

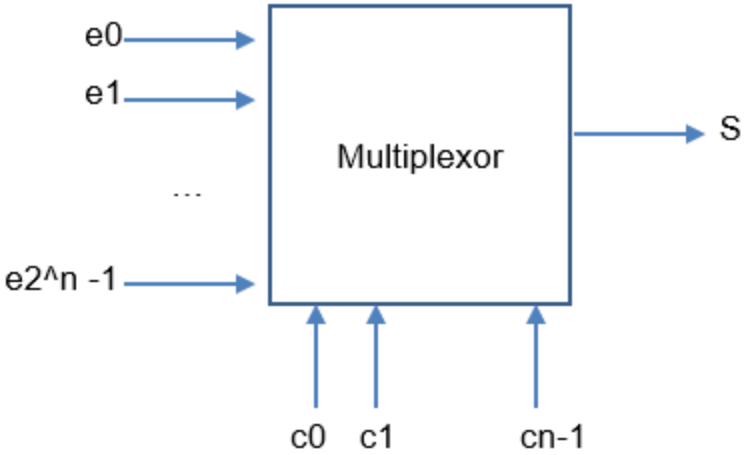
F simplificada= $(A+ C) \cdot (\overline{B}+C) \cdot (\overline{A}+B+\overline{C}) \cdot$

Índice

1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito data su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

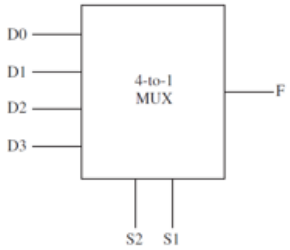
5 Circuitos Combinacionales típicos: Multiplexor (I)

- Es un circuito que conecta varias entradas con una única salida. En un instante dado, un mecanismo de selección elige cuál de las entradas es la que pasará a la salida.
- Se denominan multiplexor (MUX) n a 1, donde n es el número de entradas.

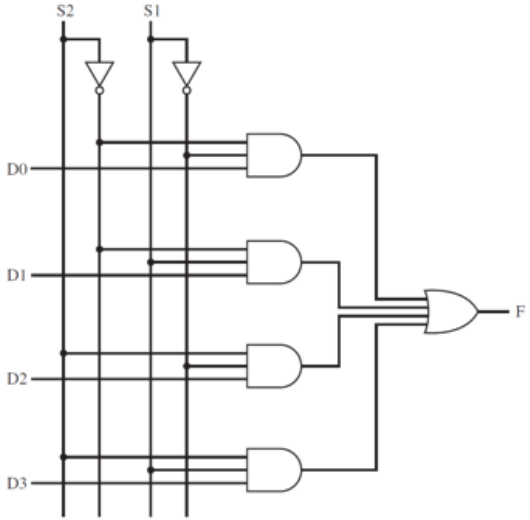


5 Circuitos Combinacionales típicos: Multiplexor (II)

- Las entradas son en potencias de 2. Por tanto, son circuitos con 2^n entradas de información, n entradas de selección y una salida. Así por ejemplo un multiplexor con 4 entradas de datos, tiene 2 entradas de selección y una única salida..



S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3



5 Circuitos Combinacionales típicos: Multiplexor (III)

- Se usan en circuitos digitales para controlar el enrutamiento de señales y datos.
- Si existe un registro cuyo valor puede aumentar debido a 4 causas diferentes, un multiplexor 4 a 1 elige en cada instante cuál de esas causas es la que va a modificar el registro

- ¿Cuál sería la tabla de verdad completa y la función booleana de un Multiplexor 4x1?

5 Circuitos Combinacionales típicos: Multiplexor (IV)

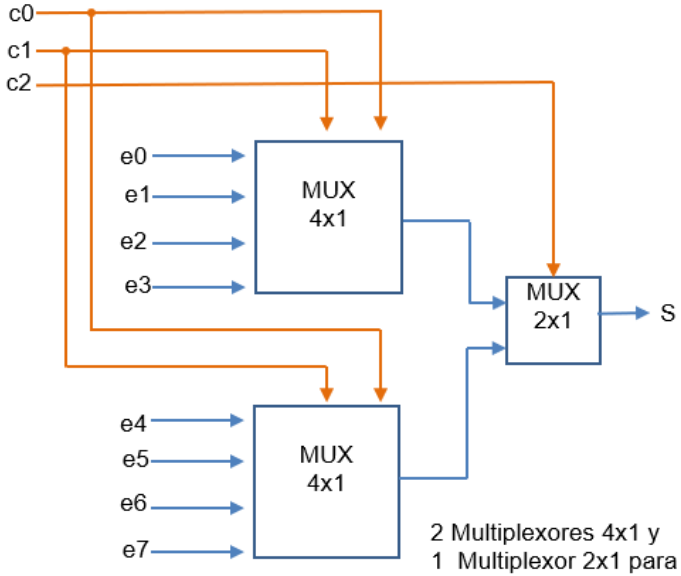
- La * implica que esa variable puede tomar cualquier valor

e3	e2	e1	e0	c1	c0	S
*	*	*	0	0	0	0
*	*	*	1	0	0	1
*	*	0	*	0	1	0
*	*	1	*	0	1	1
*	0	*	*	1	0	0
*	1	*	*	1	0	1
0	*	*	*	1	1	0
1	*	*	*	1	1	1

$$S = e3 \cdot c1 \cdot c0 + e2 \cdot c1 \cdot \overline{c0} + e1 \cdot \overline{c1} \cdot c0 + e0 \cdot \overline{c1} \cdot \overline{c0}$$

5 Circuitos Combinacionales típicos: Multiplexor (IV)

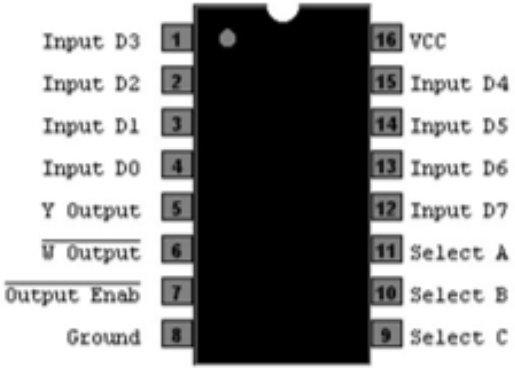
- Es bastante común que para obtener multiplexores de más de 8 entradas se empleen acopamientos de multiplexores de 2, o de 4 o de 8 entradas.



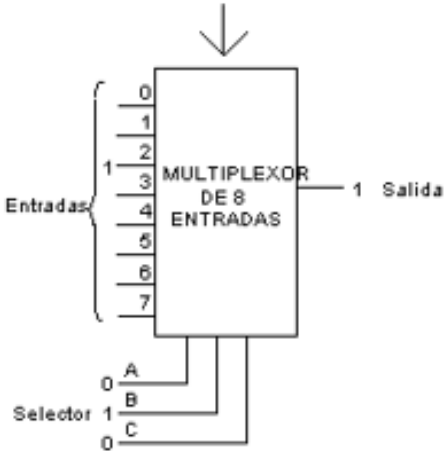
2 Multiplexores 4x1 y
1 Multiplexor 2x1 para implementar un circuito equivalente a un multiplexor 8x1

5 Circuitos Combinacionales típicos: Multiplexor (V)

- Ejemplo de un multiplexor 8x1 implementado en un chip:



Multiplexor (74LS151)

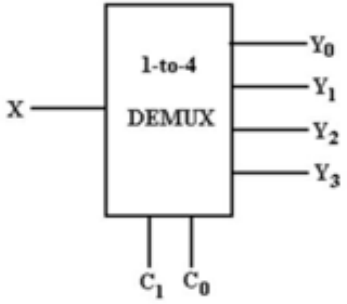


5 Circuitos Combinacionales típicos: Demultiplexor (I)

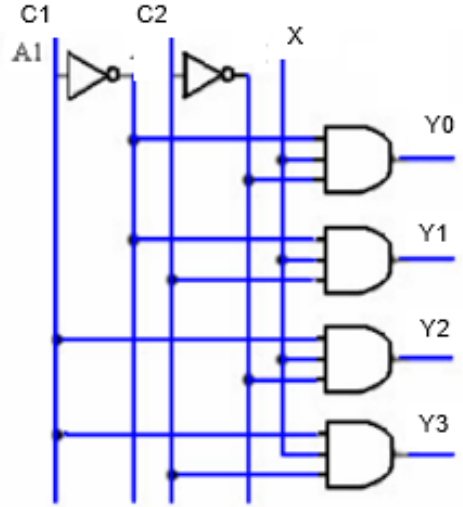
- Es el inverso del multiplexor: conecta varias salidas (Y) con una única entrada (X). En un instante dado, un mecanismo de selección (C) elige cuál de las salidas es la que se activa.
- Se denominan demultiplexor (DEMUX) 1 a n, donde n es el número de salidas.
- Las salidas son en potencias de 2, 2^y, donde y es el número de señales de selección (C).

5 Circuitos Combinacionales típicos: Demultiplexor (II)

- Suele usarse como enrutador de señales

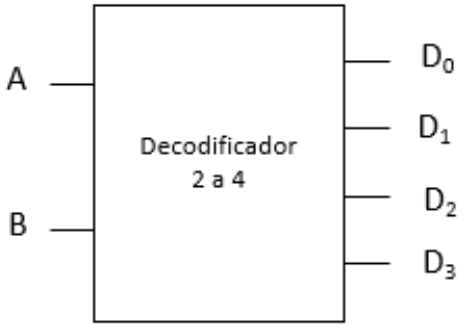


C_0	C_1	Y_0	Y_1	Y_2	Y_3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X



5 Circuitos Combinacionales típicos: Decodificador (I)

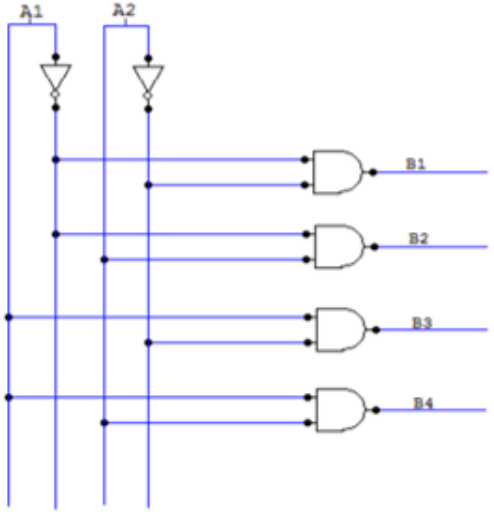
- Es un circuito que dispone n señales de entrada y 2n señales de salida (D). En un instante dado y, dependiendo del valor de las entradas, sólo una de las salidas es seleccionada.



A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

5 Circuitos Combinacionales típicos: Decodificador (II)

- Uno de sus usos más frecuentes es para decodificar direcciones:
  - Hay que elegir uno entre cuatro módulos de memoria disponibles para almacenar un dato.
  - El módulo de destino se codifica en las entradas del decodificador (A y B). El decodificador transforma ese código en la dirección del módulo de destino correcto (D0, D1, D2, D3).
- El decodificador funciona como un demultiplexor si contamos los selectores como una entrada.



5 Circuitos Combinacionales típicos: Sumador (I)

- La suma binaria funciona de la misma forma que la suma decimal:

  0 + 0 = 0

  0 + 1 = 1

  1 +1 = 10 (el 1 sale del acarreo)

- El acarreo (“Carry” en inglés) es lo que aprendimos en primaria como “lo que me llevo”.
- Podemos definir la suma binaria de dos bits como dos funciones las que llamamos S(suma) y C (Carry).

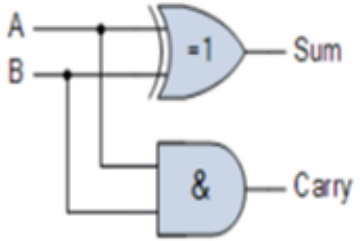
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$C = x \cdot y$

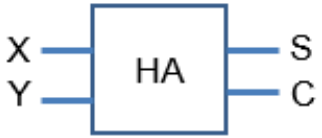
$S = x \cdot \bar{y} + \bar{x} \cdot y = X \text{ XOR } Y = x \oplus y$

5 Circuitos Combinacionales típicos: Sumador (II)

- Así podemos construir la semisuma como:



De forma simplificada se representa como

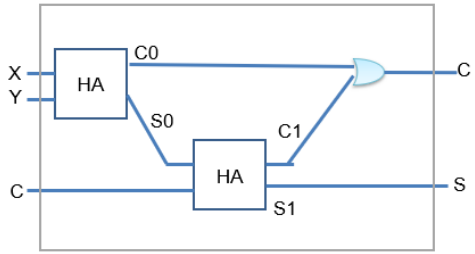


Half adder

El semisumador solo sirve para 2 bits ¿qué ocurre si queremos sumar números mayores? Emplearemos el sumador completo en serie.

5 Circuitos Combinacionales típicos: Sumador (III)

- Para sumar dos números binarios de M cifras se sigue el mismo procedimiento que empleamos en a suma decimal: sumamos las dos cifras de la posición j , con el posible acarreo de la posición j-1 y producimos una suma Sj y un acarreo Cj.
- El sumador completo necesita 3 entradas x, y, c y produce dos salidas S y C. Si nos fijamos en la forma de operar la suma es la suma de x+y con c y se produce acarreo bien si x+y lo genera o si x+y+c lo produce. A partir de esta observación el sumador completo se puede construir con dos semisumadores:



Full adder

5 Circuitos Combinacionales típicos: Sumador (IV)

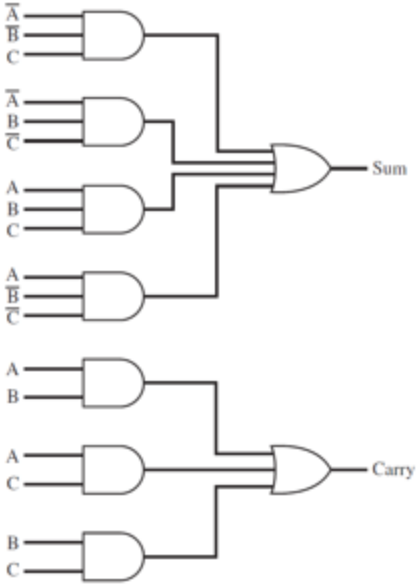
- Cuando se suma teniendo en cuenta el acarreo, la tabla es la siguiente (Carry in es el acarreo en la entrada):

C_{in}	A	B	Suma	C_{out}
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

5 Circuitos Combinacionales típicos: Sumador (V)

$$\text{Suma} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$$

$$C_{out} = AB + AC + BC$$

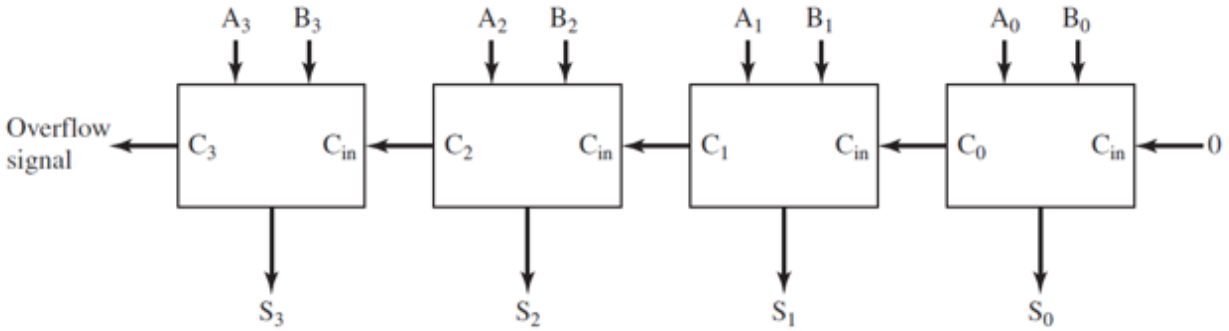


C_{in}	A	B	Suma	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

5 Circuitos Combinacionales típicos: Sumador (V)

Y la suma de M cifras se consigue conectando M sumadores completos en serie. Ejemplo:

- Sumador de ocho bits: cuatro sumadores de 2 bits encadenados secuencialmente.
- Para que un sumador ejecute su operación, necesita esperar a que el acarreo del anterior esté disponible: cuantos más bits, más se retrasa la operación.



Índice

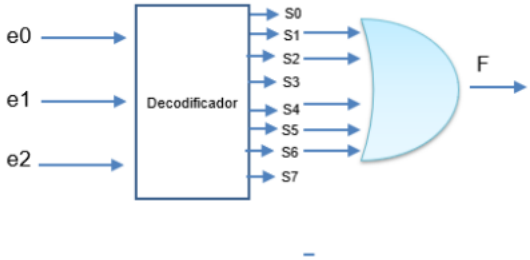
1. Introducción a circuitos
2. Circuitos combinacionales definición
3. Implementación de un circuito data su tabla de verdad
4. Complejidad de un circuito simplificación
5 Circuitos Combinacionales Típicos
6. Construcciones usando multiplexores y demultiplexores

6 Construcción de funciones lógicas usando decodificadores y multiplexores

Cualquier función de n variables puede implementarse como la salida de un decodificador de n a 2^n y puertas OR. Además a veces resulta más económico construir una función con decodificadores que hacerlo directamente con puertas.

Ejemplo: A partir de un decodificador de 3 entradas generar la función definida por la siguiente tabla de verdad:

e2	e1	e0	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Equivale a decir que F =∑ (M1, M2, M4, M5 , M6) Siendo MX la salida x del decodificador.

De la misma forma que acabamos de ver con decodificadores, con multiplexores se pueden también construir funciones lógicas. La diferencia es que para generar la función no vamos a necesitar añadir puertas lógicas.