Tema 2: Sistemas digitales.

2.1- Introducción.

2.2- Sistemas combinacionales.

- *Formas de descripción: Función booleana y Tabla de verdad.*
- *Sistemas combinacionales elementales: puertas lógicas.*
- *Sistemas combinacionales específicos: MUX y DECO.*
- *Diseño de circuitos combinacionales: mediante puertas lógicas y mediante PLAs (Sumador Elemental).*
- *Diseño de circuitos combinacionales complejos (SUM4, ALU16).*

2.3- Sistemas secuenciales: biestables y registros.

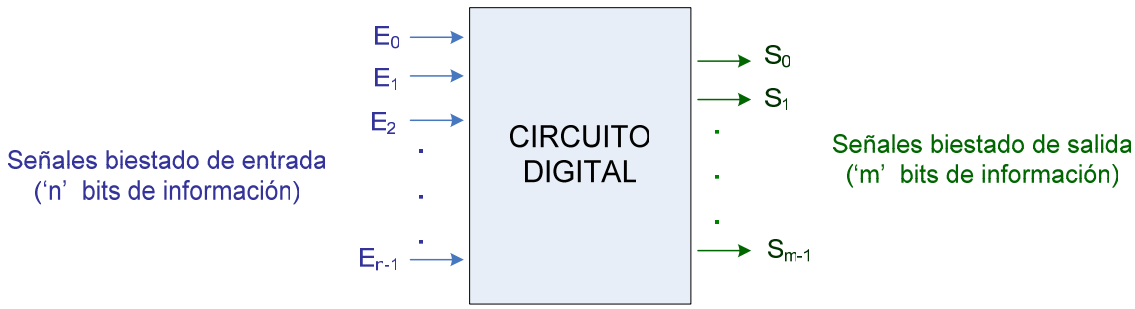
2.4- Interconexión de componentes: buses y lógica triestado.

2.5- Sincronización de dispositivos digitales: relojes.

2.6- Memorias: ROM y RAM estática.

2.1- Introducción

- Los sistemas diseñados para trabajar con información digital se denominan *sistemas digitales*. Un sistema digital puede considerarse como un bloque funcional que recibe un conjunto de señales biestado de entrada, las procesa, y genera un conjunto de señales biestado de salida.



- Los sistemas digitales se pueden clasificar en:

  - ***Combinacionales***: la salida depende únicamente de las entradas, NO de su estado interno en ese momento (→ NO tienen memoria).
    Esto significa que, ante una misma combinación de entrada, siempre generarán la misma salida.

  - ***Secuenciales***: la salida no sólo depende de las entradas, sino también del estado interno en el que se encuentre el sistema digital (→ SÍ tienen memoria).
    Esto significa que, ante una misma combinación de entrada, pueden no generar siempre la misma salida.

- **Introducción: Operadores Booleanos**
  - Pueden representarse de varias formas:

    a) a menudo se representan simplemente como AND (Y), OR (O) y NOT (Negación).

    b) en electrónica digital (puertas lógicas) también se emplean la X-OR (O-exclusiva) y las negadas NAND (NO-Y), NOR (NO-O) y X-NOR (equivalencia).

    c) En matemática a menudo se utiliza:

    - + (en lugar de OR)
    - * (en lugar de AND)

    Ya que dichas operaciones son de alguna manera análogas a la suma y el producto en otras estructuras algebraicas.

    - NOT → se representa como una línea o una comilla sobre la expresión que se pretende negar (NO-A sería Ā o A').

– Tablas de verdad asociadas a los operadores booleanos:

A	B	Suma Lógica (OR, +)	Producto Lógico (AND, *)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

A	Negación (NOT, $\bar{A}$)
0	1
1	0

Salida igual a 0 sólo en caso de que todas las entradas estén a 0.

Salida igual a 1 sólo en caso de que todas las entradas estén a 1.

- **FORMAS DE DESCRIPCIÓN DE UN CIRCUITO COMBINACIONAL**

Existen dos formas: con una *función booleana* o con una *tabla de verdad*.

a) ***Función booleana***: expresa la combinación de variables de tipo booleano (0, 1) mediante operadores booleanos (OR, AND, NOT). Describe cada salida como una función booleana de las entradas.

$$S_0 = f_0 (E_0, E_1, E_2, \ldots, E_i, \ldots, E_{n-1})$$

. . .

$$S_{m-1} = f_{m-1} (E_0, E_1, E_2, \ldots, E_i, \ldots, E_{n-1})$$

Ejemplo de función booleana: $f(A, B, C) = (A * B) + \overline{(B * C * A)} + C$

Las funciones booleanas son una forma de descripción de tipo matemático basada en el ***Álgebra de Boole***, que se define como un conjunto de leyes y reglas de operación sobre variables lógicas.

Propiedades del Álgebra de Boole:

- Conmutativa: $a + b = b + a$
- Asociativa: $(a + b) + c = a + (b + c)$; $(a * b) * c = a * (b * c)$
- Idempotencia: $a + a = a$ // $a * a = a$; $a + \overline{a} = 1$ // $a * \overline{a} = 0$
- Distributiva: $a * (b+c) = (a*b) + (a*c)$; $a + (b*c) = (a+b) * (a+c)$
- Leyes de De Morgan: $\overline{(a+b)} = \overline{a} * \overline{b}$ // $\overline{(a * b)} = \overline{a} + \overline{b}$
  $\neq \overline{a} + \overline{b}$ $\neq \overline{a} * \overline{b}$

b) ***Tabla de verdad***: dadas 'n' entradas, define para todas y cada una de las combinaciones de entrada posibles (2^n), las combinaciones de salida que les corresponden.

Ejemplo:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Operaciones binarias

A	B	Suma Lógica (OR, +)	Producto Lógico (AND, *)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Operación unitaria

A	Negación (NOT)
0	1
1	0

CÓMO PASAR DE UNA FORMA DE DESCRIPCIÓN A LA OTRA

i) ***De Función booleana a Tabla de verdad***: para todas las combinaciones posibles de las variables de entrada, se calcula el valor de cada salida utilizando la función booleana dada.

Ejemplo:

$X = A + B + C$

$Y = A * B + A * \overline{C}$

A	B	C	X	Y
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

ii) ***De Tabla de verdad a Función booleana***:

- Un **MINTERM** es un producto en el que intervienen las 'n' variables correspondientes a una función booleana, pudiendo éstas aparecer en forma natural o complementada.
- La función booleana representativa de una tabla de verdad se obtiene sumando los MINTERMS correspondientes a las combinaciones de entrada que generan un valor igual a "1" en la salida.

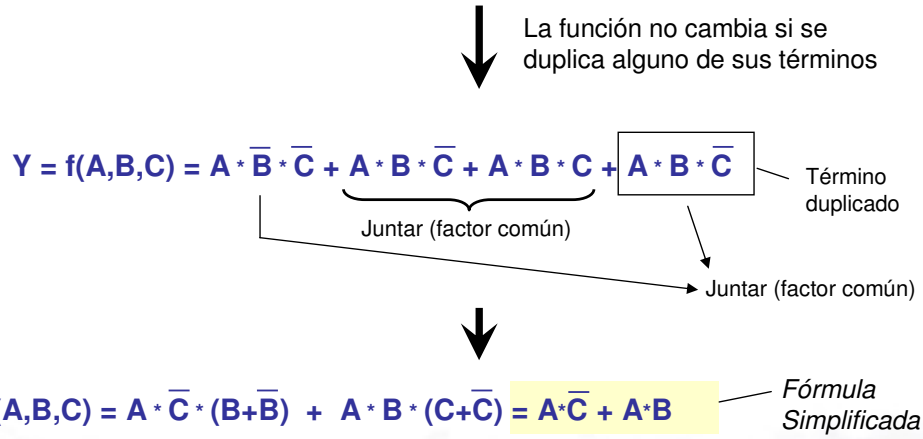
	A	B	C	S
	0	0	0	0
	0	0	1	0
$MINTERM_2$	0	1	0	1
	0	1	1	0
	1	0	0	0
$MINTERM_5$	1	0	1	1
	1	1	0	0
$MINTERM_7$	1	1	1	1

→ $S = \overline{A} * B * \overline{C} + A * \overline{B} * C + A * B * C$

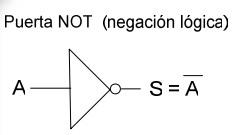
2.2- Sistemas Combinacionales: formas de descripción

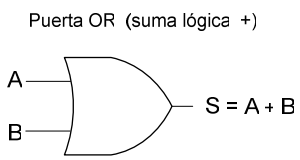
- Cuando una función booleana está representada como una suma de MINTERMS, se dice que la función está expresada en **forma CANÓNICA**.
- En muchos casos, la función canónica puede **simplificarse** para obtener una expresión matemática más sencilla que conlleve a los mismos valores de salida.

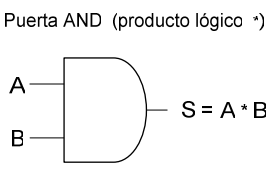
Ejemplo: $Y = f(A,B,C) = A * \overline{B} * \overline{C} + A * B * \overline{C} + A * B * C$



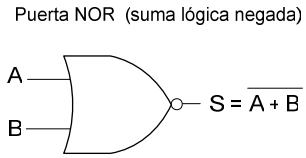
2.2- Sistemas Combinacionales: puertas lógicas

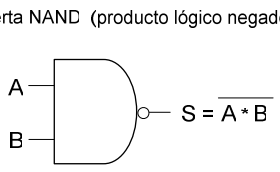




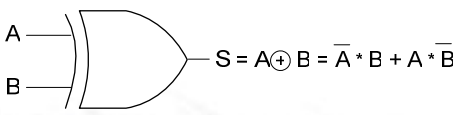


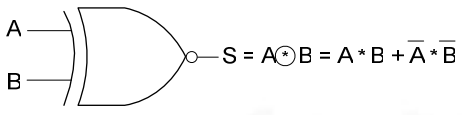
A	B	$\overline{A}$	A+B	A * B
0	0	1	0	0
0	1	0	1	0
1	0		1	0
1	1		1	1





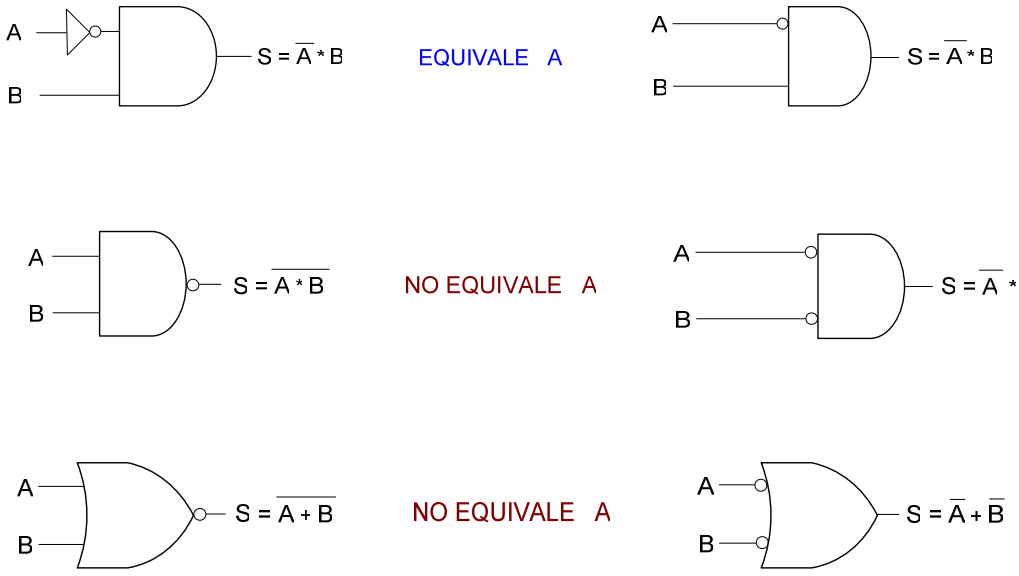
A	B	$\overline{A+B}$	$\overline{A * B}$
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0



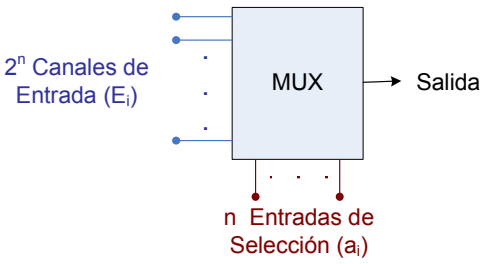


A	B	A⊕B	A⊛B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

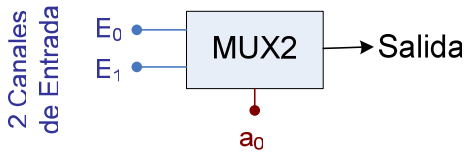
Observaciones:



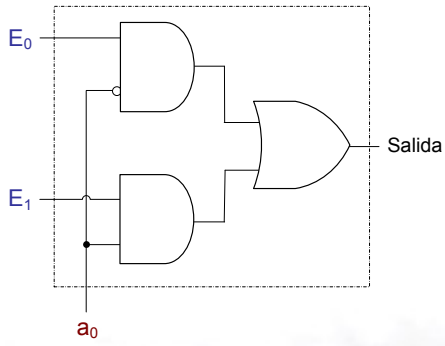
Funcionalidad: Conmutador electrónico. Selecciona un canal entre los 2^n canales de entrada posibles (E_i), llevando su valor a la salida (S). El canal seleccionado será el indicado en las entradas de selección (a_i).



Ejemplo MUX2:

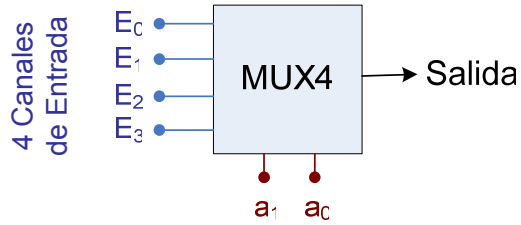


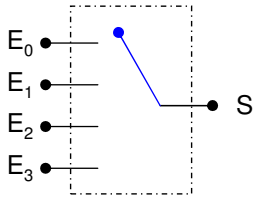
a_0	E_1	E_0	S
0	x	0	0
0	x	1	1
1	0	x	0
1	1	x	1



2.2- Sistemas Combinacionales: Multiplexador (MUX)

Ejemplo MUX4:

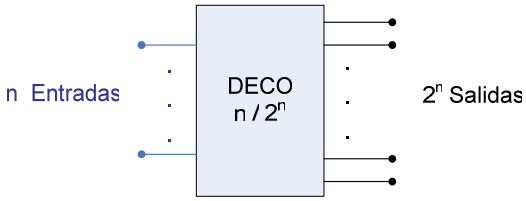




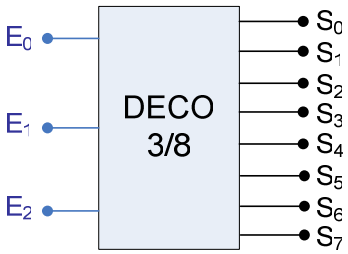
	a_1	a_0	E_3	E_2	E_1	E_0	S
Selección E_0	0	0	x	x	x	0	0
	0	0	x	x	x	1	1
Selección E_1	0	1	x	x	0	x	0
	0	1	x	x	1	x	1
Selección E_2	1	0	x	0	x	x	0
	1	0	x	1	x	x	1
Selección E_3	1	1	0	x	x	x	0
	1	1	1	x	x	x	1

2.2- Sistemas Combinacionales: Decodificador (DECO)

Funcionalidad: cada combinación de entrada (E_i) activa una y sola una línea de salida (S_i). Se activará la salida correspondiente al nº binario representado por los bits de entrada.



Ejemplo DECO 3/8:



	e_2	e_1	e_0	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
Activación S_0	0	0	0	0	0	0	0	0	0	0	1
	0	0	1	0	0	0	0	0	0	1	0
	0	1	0	0	0	0	0	0	1	0	0
	0	1	1	0	0	0	0	1	0	0	0
Activación S_4	1	0	0	0	0	0	1	0	0	0	0
	1	0	1	0	0	1	0	0	0	0	0
	1	1	0	0	1	0	0	0	0	0	0
Activación S_7	1	1	1	1	0	0	0	0	0	0	0

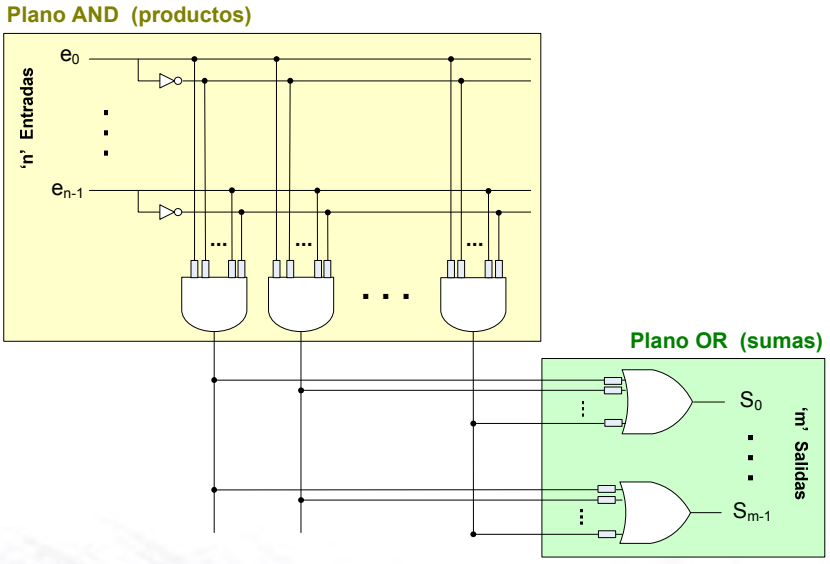
a) Diseño de circuitos combinacionales mediante Puertas Lógicas.

Pasos
- a.1) Determinar las entradas y las salidas del circuito.
- a.2) Determinar las ecuaciones booleanas que expresan cada una de las salidas en función de las entradas, en la forma más simplificada posible *(partir de la tabla de verdad)*.
- a.3) Construir mediante puertas lógicas el circuito que cumpla las ecuaciones booleanas obtenidas.

b) Diseño de circuitos combinacionales mediante Arrays Lógicos Program.

Pasos
- b.1) Partiendo de la tabla de verdad, construir todos los MINTERMS que participen en las salidas (construcción del plano AND).
- b.2) Generar cada una de las salidas de la tabla sumando los MINTERMS que corresponda (construcción del plano OR).

– Un Array Lógico Programable (o PLA) es una estructura de puertas lógicas orientadas a implementar ecuaciones lógicas canónicas.
– Dos planos: AND (productos de las entradas) y OR (suma para cada salida).



La estructura interna de un PLA está compuesta de fusibles.

Programar el PLA consistirá en dejar activos/desactivos los fusibles que corresponda.

Los PLA son especialmente útiles para diseñar circuitos combinacionales con muchas entradas y salidas.

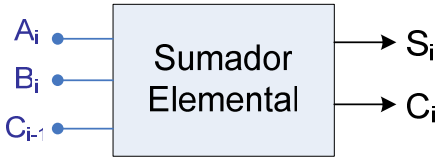
Ejemplo: Diseño de un Sumador Elemental (1 bit)

a) Diseño mediante puertas lógicas.

a.1) Determinar entradas y salidas del circuito (→ Funcionalidad: sumar dos números de 1 bit).

$$\begin{array}{r} {}^{1}0\,{}^{1}0\,1\,0 \\ +\;\; 0\;1\;1\;1 \\ \hline 1\;0\;0\;1 \end{array}$$

Esquema de Entradas y Salidas



a.2) Obtener las ecuaciones booleanas de las salidas (a partir de la tabla de verdad).

A_i	B_i	C_{i-1}	S_i	C_i
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

$$S = m_1 + m_2 + m_4 + m_7 = \overline{A}\,\overline{B}\,C_{-1} + \overline{A}\,B\,\overline{C_{-1}} + A\,\overline{B}\,\overline{C_{-1}} + A\,B\,C_{-1} =$$

$$= (\overline{A}\,\overline{B} + A\,B)\,C_{-1} + (\overline{A}\,B + A\,\overline{B})\,\overline{C_{-1}} =$$

(Equivalencia: $\overline{A}\,\overline{B} + A\,B$; O-Exclusiva: $\overline{A}\,B + A\,\overline{B}$)

$$= (A \circledast B)\,C_{-1} + (A \oplus B)\,\overline{C_{-1}} =$$

$$\circledast = \overline{\oplus}$$

$$= (\overline{A \oplus B})\,C_{-1} + (A \oplus B)\,\overline{C_{-1}} =$$

$$= \boxed{(A \oplus B) \oplus C_{-1}}$$

$\overline{X}\,Y + X\,\overline{Y} = X \oplus Y$

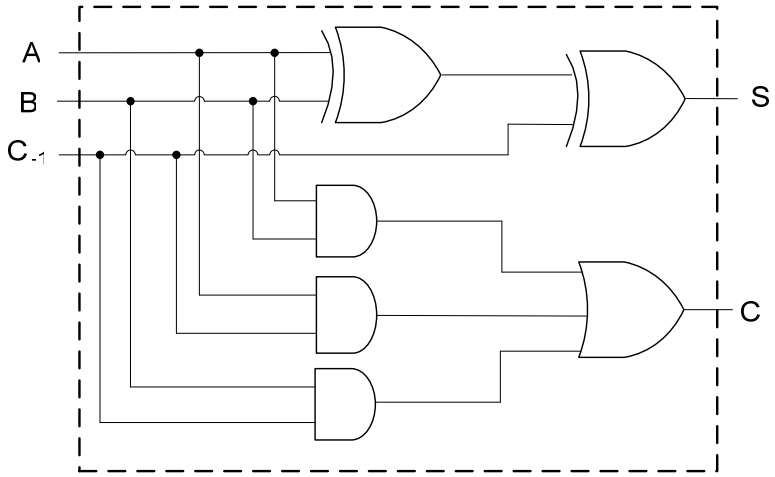
$$C = m_3 + m_5 + m_6 + m_7 = \overline{A}\,B\,C_{-1} + A\,\overline{B}\,C_{-1} + A\,B\,\overline{C_{-1}} + A\,B\,C_{-1} =$$

$$= \overline{A}\,B\,C_{-1} + A\,\overline{B}\,C_{-1} + A\,B\,\overline{C_{-1}} + A\,B\,C_{-1} + A\,B\,C_{-1} + A\,B\,C_{-1} =$$

$$= (\overline{A} + A)\,B\,C_{-1} + (\overline{B} + B)\,A\,C_{-1} + (\overline{C_{-1}} + C_{-1})\,A\,B =$$

$$= \boxed{B\,C_{-1} + A\,C_{-1} + A\,B}$$

a.3) Construir el circuito mediante puertas lógicas.



$S_i = (A_i \oplus B_i) \oplus C_{i-1}$

$C_i = A_i B_i + B_i C_{i-1} + A_i C_{i-1}$

b) Diseño mediante PLA.

b.1) Construir los MINTERMs que participen en las salidas.

A_i	B_i	C_{i-1}	S_i	C_i		
0	0	0	0	0		
0	0	1	1	0	→	$\overline{A_i}\,\overline{B_i}\,C_{i-1}$
0	1	0	1	0	→	$\overline{A_i}\,B_i\,\overline{C_{i-1}}$
0	1	1	0	1	→	$\overline{A_i}\,B_i\,C_{i-1}$
1	0	0	1	0	→	$A_i\,\overline{B_i}\,\overline{C_{i-1}}$
1	0	1	0	1	→	$A_i\,\overline{B_i}\,C_{i-1}$
1	1	0	0	1	→	$A_i\,B_i\,\overline{C_{i-1}}$
1	1	1	1	1	→	$A_i\,B_i\,C_{i-1}$

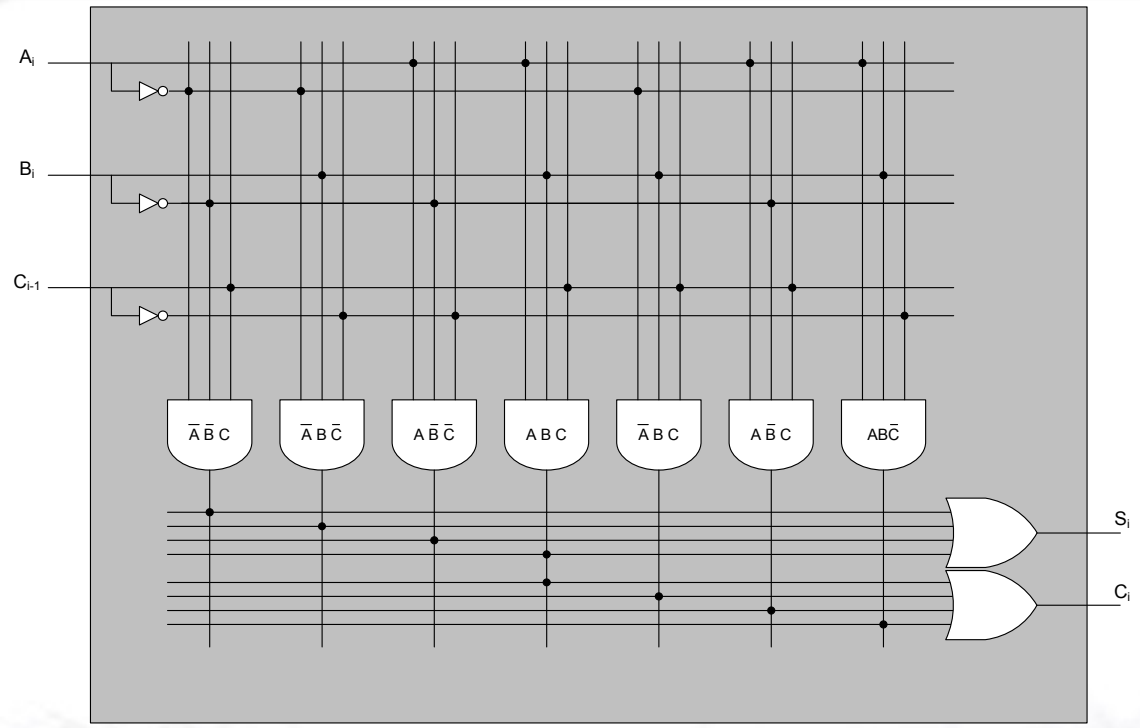
MINTERMS a generar (plano AND)

b.2) Generar cada una de las salidas en base a los MINTERMS identificados.

$S_i = \overline{A_i}\,\overline{B_i}\,C_{i-1} + \overline{A_i}\,B_i\,\overline{C_{i-1}} + A_i\,\overline{B_i}\,\overline{C_{i-1}} + A_i\,B_i\,C_{i-1}$

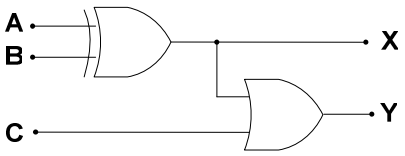
$C_i = \overline{A_i}\,B_i\,C_{i-1} + A_i\,\overline{B_i}\,C_{i-1} + A_i\,B_i\,\overline{C_{i-1}} + A_i\,B_i\,C_{i-1}$

SALIDAS (plano OR)



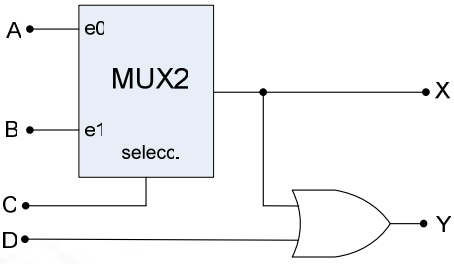
Ejercicios propuestos:

1. Se quiere diseñar el circuito de la figura adjunta utilizando un array lógico programable. ¿Cuántas puertas AND y OR deberá tener el PLA?



***Solución**: 6 puertas AND*
2 puertas OR

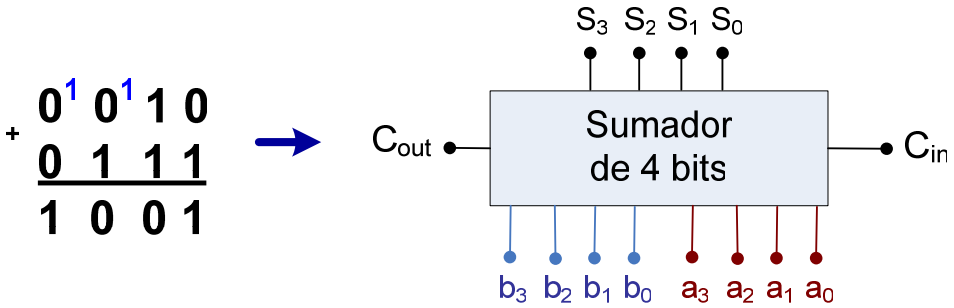
2. Si se rediseña el circuito de la figura adjunta utilizando un array lógico programable. ¿Cuántas puertas AND y OR deberá tener el PLA?



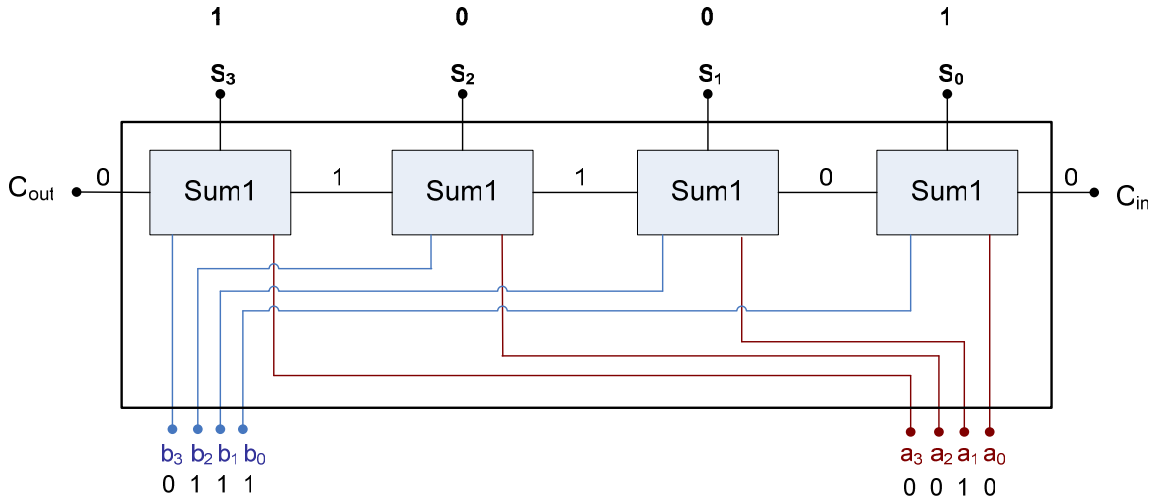
***Solución**: 12 puertas AND*
2 puertas OR

Circuito 1: Diseño de un Sumador de 4 bits *(basado en Sumador Elemental)*

Determinar entradas y salidas del circuito (→ Funcionalidad: sumar dos números de 4 bits).



El sumador de 4 bits se construye a partir de cuatro sumadores elementales.



Circuito 2: Unidad Aritmético-Lógica (ALU)

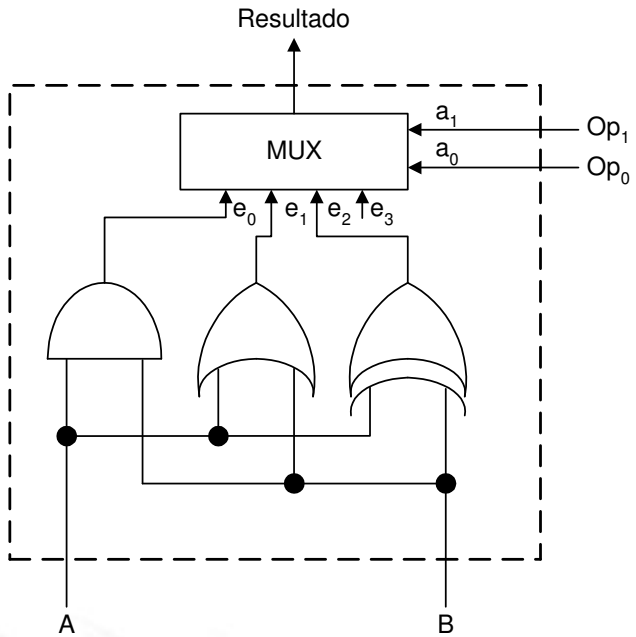
Características

- Tamaño de los operandos: 16 bits.
- Operaciones:
  - Aritméticas: SUMA. RESTA
  - Lógicas: AND, OR, XOR

Ej. 4 bits:	0 1 0 1 (A)
	0 0 1 1 (B)
(Suma)	1 0 0 0
(Resta)	0 0 1 0
(AND)	0 0 0 1
(OR)	0 1 1 1
(XOR)	0 1 1 0

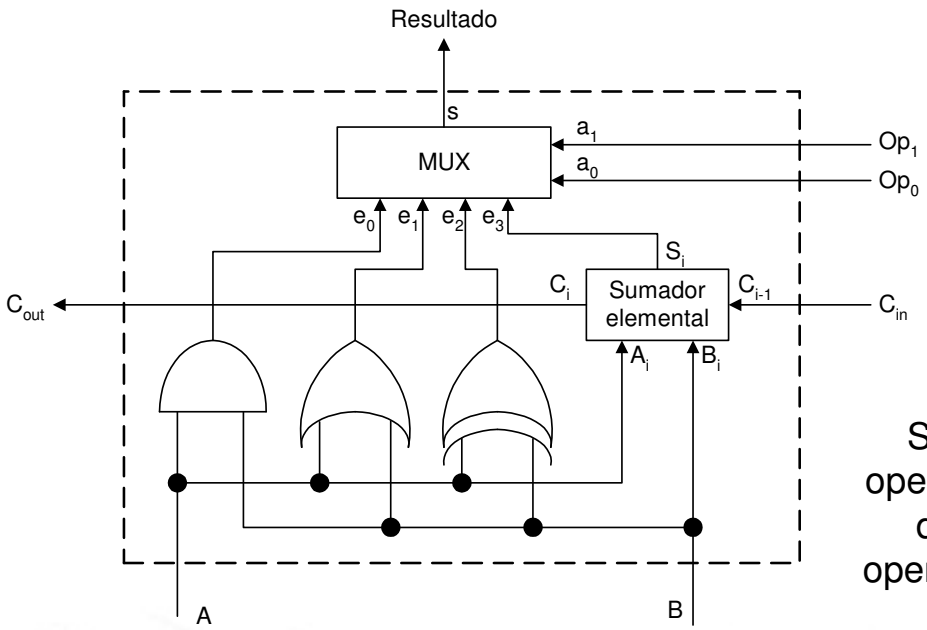
- Debe generar 4 bits informativos del resultado:
  - Bit de Zero (ZF) → indica si el resultado es 0.
  - Bit de Carry (CF) → desbordamiento de Naturales.
  - Bit de Overflow (OF) → desbordamiento de Enteros.
  - Bit de Signo (SF) → indicativo del signo del resultado.

• CONSTRUCCIÓN DE LA ALU ELEMENTAL (1 de 3)



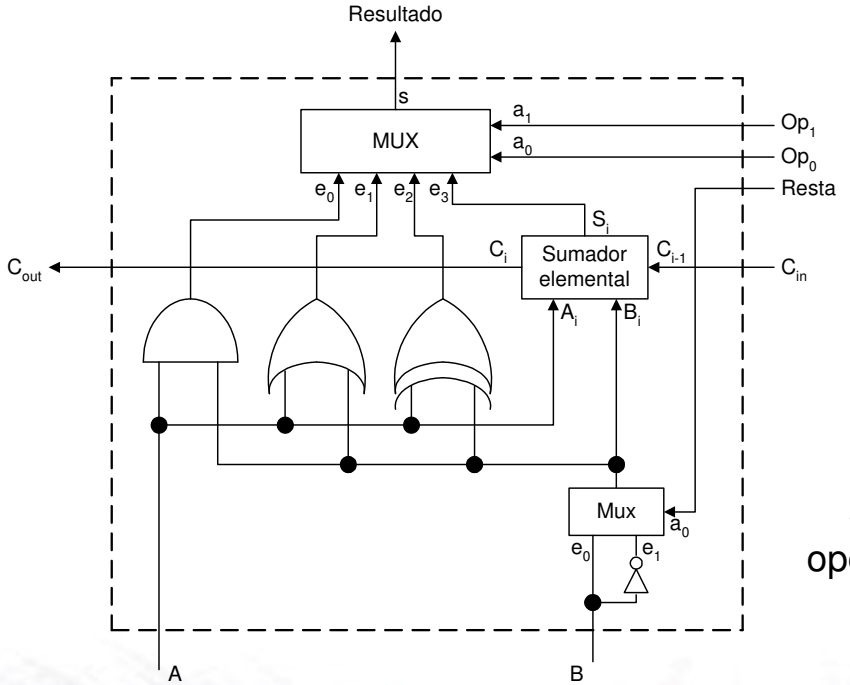
Implementación de las operaciones lógicas.

• CONSTRUCCIÓN DE LA ALU ELEMENTAL (2 de 3)



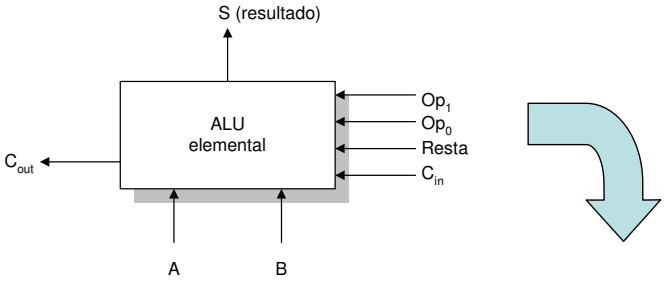
Se incorpora la operación aritmética de suma a las operaciones lógicas.

• CONSTRUCCIÓN DE LA ALU ELEMENTAL (3 de 3)

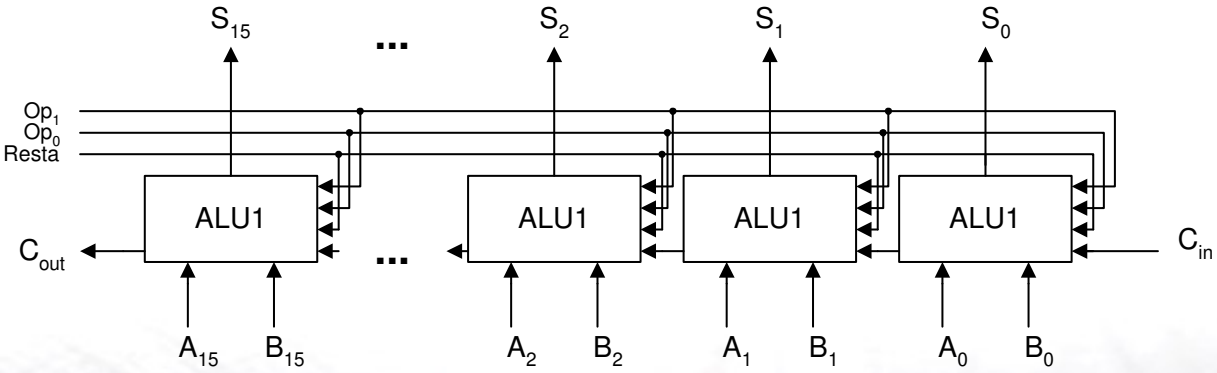


Se incorpora la operación aritmética de resta.

• CONSTRUCCIÓN DE LA ALU DE LA CPU TEÓRICA (1 de 3)



Agrupando 16 ALUs elementales y conectándolas adecuadamente:

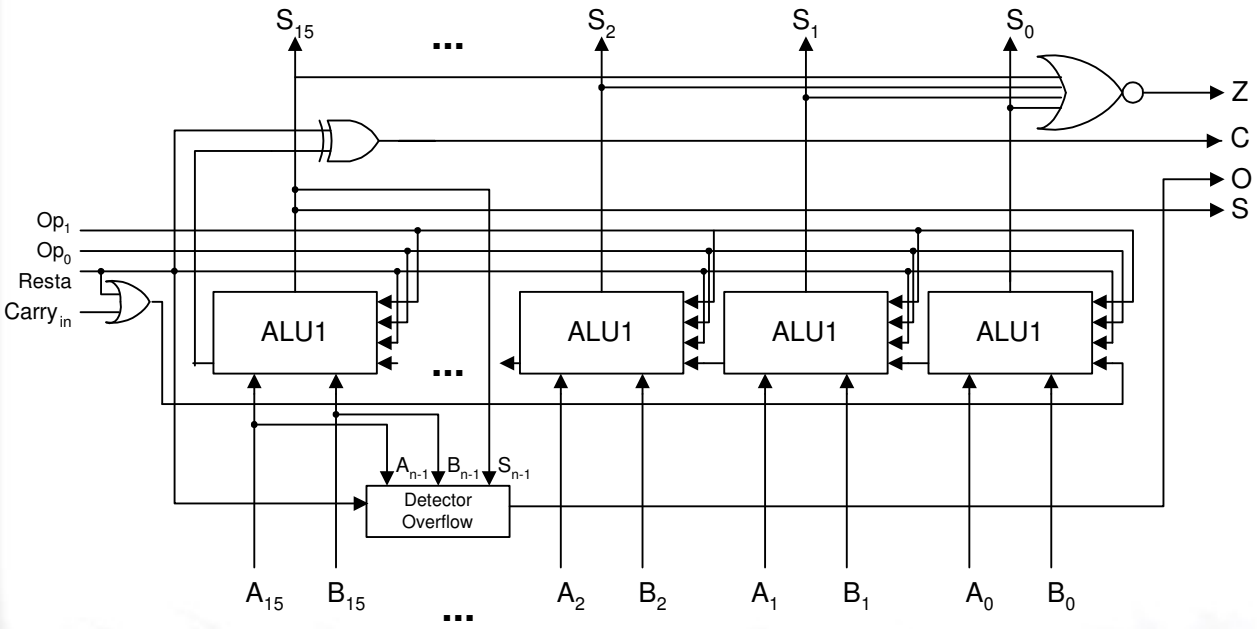


• CONSTRUCCIÓN DE LA ALU DE LA CPU TEÓRICA (2 de 3)

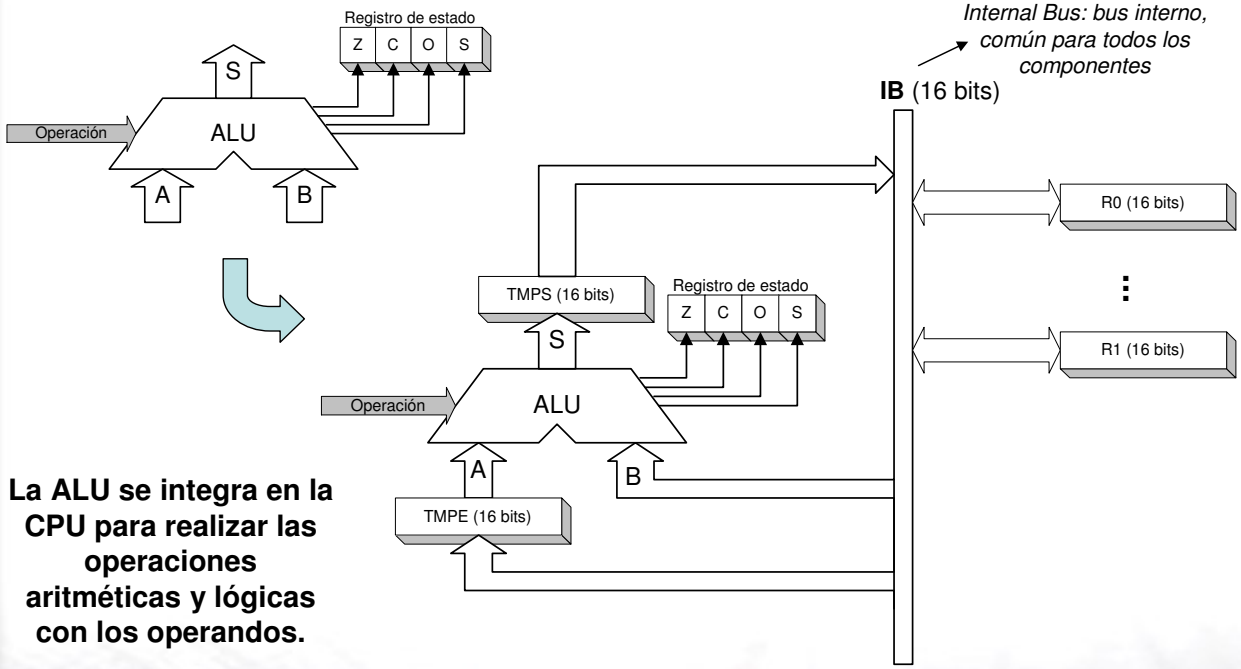
Detección del Flag de Overflow (OF), para Sumas y Restas.

Resta	Signo *A*	Signo *B*	Signo Resultado	Overflow
0	0	0	0	No
0	0	0	1	**Sí**
0	0	1	0	No
0	0	1	1	No
0	1	0	0	No
0	1	0	1	No
0	1	1	0	**Sí**
0	1	1	1	No
1	0	0	0	No
1	0	0	1	No
1	0	1	0	No
1	0	1	1	**Sí**
1	1	0	0	**Sí**
1	1	0	1	No
1	1	1	0	No
1	1	1	1	No

• CONSTRUCCIÓN DE LA ALU DE LA CPU TEÓRICA (3 de 3)



• INTEGRACIÓN DE LA ALU EN LA CPU TEÓRICA



La ALU se integra en la CPU para realizar las operaciones aritméticas y lógicas con los operandos.

Ejercicios propuestos:

1. **Se desea construir una ALU que sea capaz de operar con números enteros expresados en C-2. El rango de los números que debe manejar es [-256, 255]. Se sabe que dicha ALU estará constituida a partir de sumadores elementales, puertas lógicas de diversos tipos y multiplexadores de 2 y 4 canales de entrada. ¿Cuántos sumadores elementales serán necesarios para construir dicha ALU?**

   (Solución: 9 sumadores elementales)

2. **En el diseño de una ALU de 4 bits se ha cometido un error al conectar las líneas. El error ha consistido en que en la entrada de Carry de la ALU de peso 3 se ha conectado la línea de Resta. El resto del diseño ha sido correcto, y análogo al visto en clase para la ALU de la CPU elemental. Determinar el valor de la señal $Carry_{out}$ así como el resultado que producirá esta ALU al realizar la operación aritmética Ah – Dh (responder en hexadecimal).**

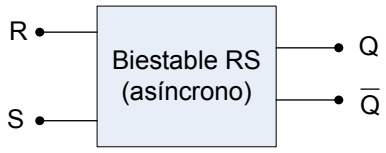
   [Solución: $Carry_{out}$ = 0, Resultado = 5h (0101)]

- Elemento de memoria más básico → el que tiene la capacidad de almacenar 1 bit de información.
- Los circuitos digitales que tienen esta capacidad se denominan Biestables (o flip-flops).
- La diferencia entre estos sistemas y los sistemas combinacionales vistos anteriormente es que los sistemas secuenciales tienen “memoria” (capacidad de recordar estados anteriores).
- Se verán biestables de tipo asíncrono y síncrono (por flanco).
- Tipos de biestables a estudiar:
  - **Biestable RS asíncrono**
  - **Biestable RS síncrono (por flanco).**
  - **Biestable D síncrono (por flanco).**

Biestable RS asíncrono

- Funcionalidad: almacenar 1 bit de información.

Esquema de Entrada/Salidas



R → Reset (puesta a 0)
S → Set (puesta a 1)

Estructura interna

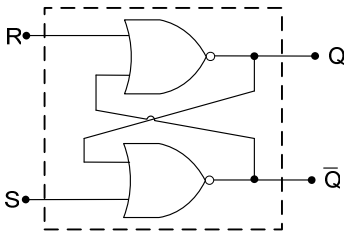


Tabla de verdad

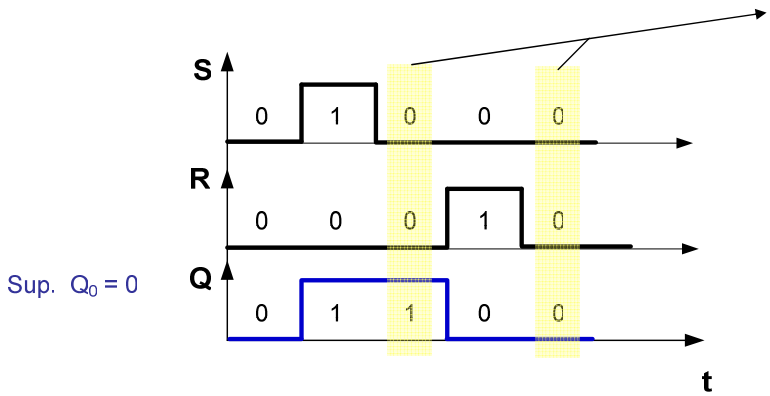
R	S	Q_T
0	0	Q_{T-1}
0	1	1
1	0	0
1	1	prohibida

$R=S=0 \rightarrow Q_T = Q_{T-1}$

↓

Concepto de memorización

- *Funcionamiento del Biestable RS asíncrono*: se estudiará utilizando un cronograma, que es un esquema en el que se representa la evolución temporal de las entradas, salidas y estados internos de un circuito.



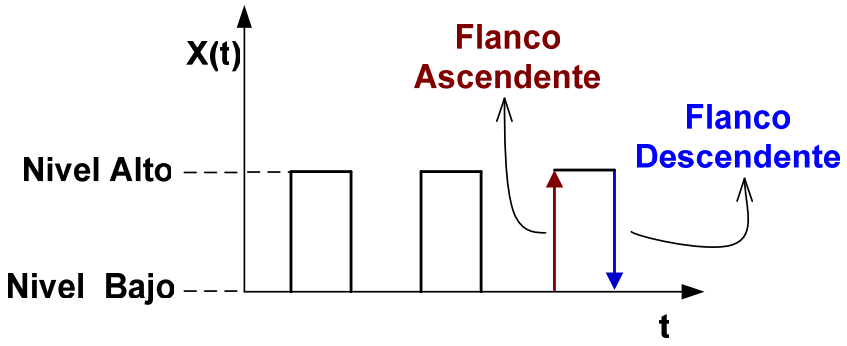
Se ve cómo una misma combinación de valores de entrada (R=S=0) puede dar lugar a distintos valores de salida en función del estado anterior.

- Al tratarse de un circuito asíncrono, tanto el estado interno como las salidas del circuito pueden modificarse en cualquier momento, en base a los valores de las señales de entrada.

Biestables Síncronos

- El diseño de circuitos digitales complejos, como p.e. un microprocesador, utilizando elementos de memoria asíncronos es muy complejo → su diseño se basa en elementos de memoria síncronos.
- Diferencia entre los elementos de memoria asíncronos y síncronos: estos últimos poseen una entrada adicional denominada *Entrada de Control*.
- Los estados internos y la salida de un biestable síncrono sólo evolucionarán cuando la entrada de control lo permita.

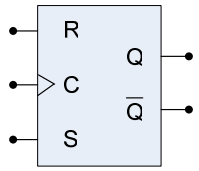
Control
- Por nivel
  - **alto**
  - **bajo**
- Por flanco
  - **Ascendente → Paso de nivel bajo a nivel alto.**
  - **Descendente → Paso de nivel alto a nivel bajo.**



- Un **flanco** es el instante del tiempo en el que se produce la transición del estado de una señal.
- Los biestables controlados por flanco se activarán o bien cuando se produzcan flancos **ascendentes** (↑) en su entrada de control, o bien cuando se produzcan flancos **descendentes** (↓).

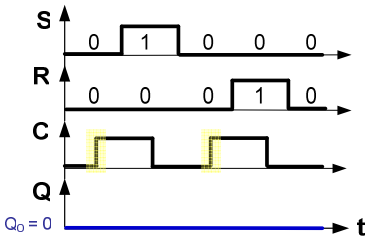
Biestable RS Síncrono (por flanco)

Por flanco ascendente

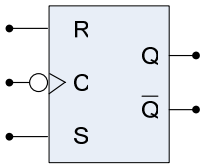


C	R	S	Q_T
↑	0	0	Q_{T-1}
↑	0	1	1
↑	1	0	0
↑	1	1	prohibida
Otro caso	X	X	Q_{T-1}

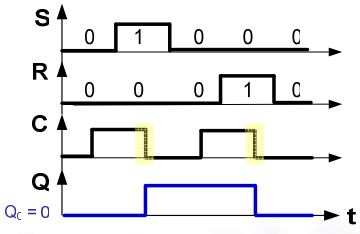
***Ejemplos*:**



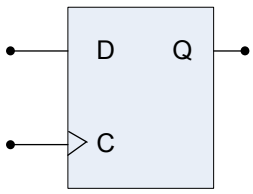
Por flanco descendente



C	R	S	Q_T
↓	0	0	Q_{T-1}
↓	0	1	1
↓	1	0	0
↓	1	1	prohibida
Otro caso	X	X	Q_{T-1}

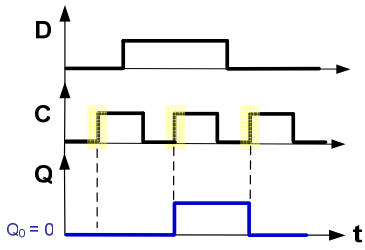


Biestable D (síncrono por flanco ascendente)



C	D	Q_T
↑	0	0
↑	1	1
Otro caso	X	Q_{T-1}

***Ejemplo*:**



- Un biestable tipo D sirve para almacenar 1 bit de información.
- El bit a almacenar se sitúa en la entrada D, y se lleva un flanco ascendente a la entrada de control.
- Como consecuencia, el valor que haya en la entrada D pasará a la salida Q.

Ejemplo de cronograma de biestable D:

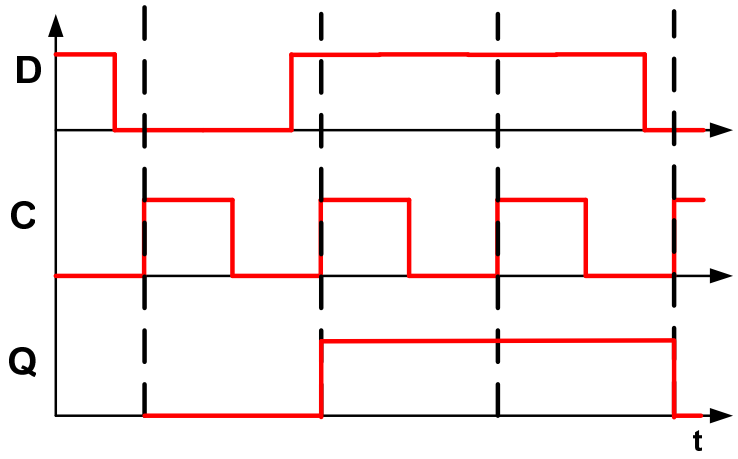


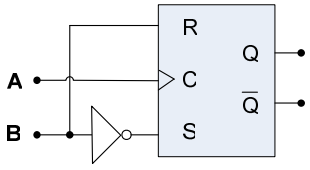
TABLA DE VERDAD RESUMIDA

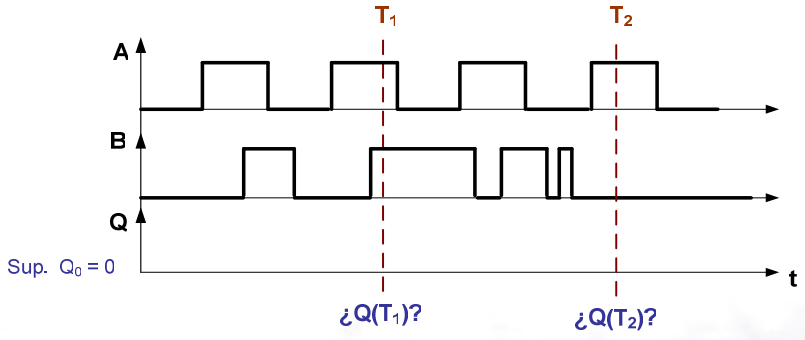
C	D	Q
↑	0	0
↑	1	1
OTRO CASO	X	NO CAMBIA

Ejercicios propuestos:

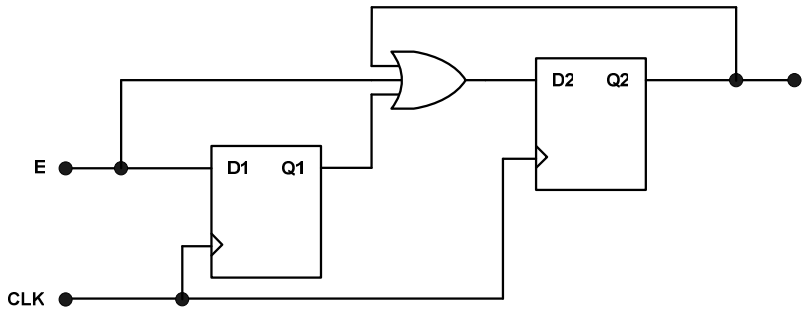
❶ **En el circuito de la figura, las entrada A y B evolucionan tal y como muestra el cronograma adjunto. Sobre dicho cronograma se han marcado dos instantes de tiempo, denominados T_1 y T_2. ¿Qué valor tendrá la salida Q en cada uno de dichos instantes?**

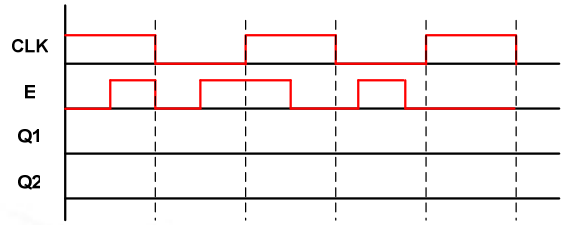
Solución: $Q(T_1) = 1$, $Q(T_2) = 1$



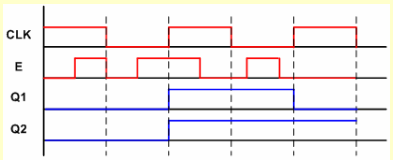


2 **Dado el circuito de la figura, completar el cronograma dibujando la evolución de las salidas Q_1 y Q_2.**



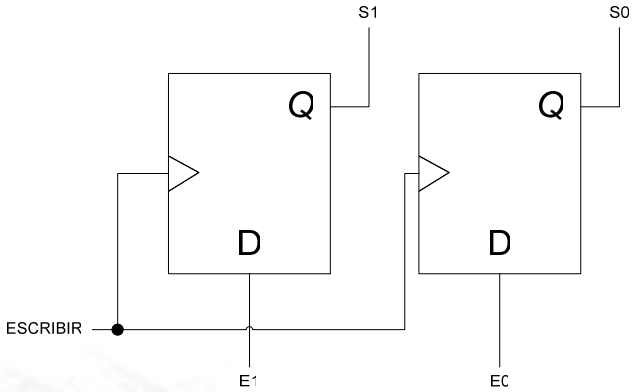


Solución:



- Biestable D → almacena 1 bit de información digital.
- 'n' Biestables D → almacenarán 'n' bits de información digital.
- Registro: Conjunto de 'n' biestables D con las entradas de control conectadas entre sí. El objetivo es almacenar un dato de 'n' bits.

Ejemplo: registro de 2 bits

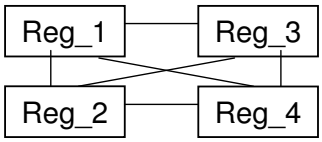


- Diferenciar líneas de datos de líneas de control.
- Si no hay flanco ascendente en la señal ESCRIBIR, E0 y E1 pueden cambiar tantas veces como sea, sin que S0 y S1 varíen.

- **INTRODUCCIÓN A LA PROBLEMÁTICA DE LA INTERCONEXIÓN**

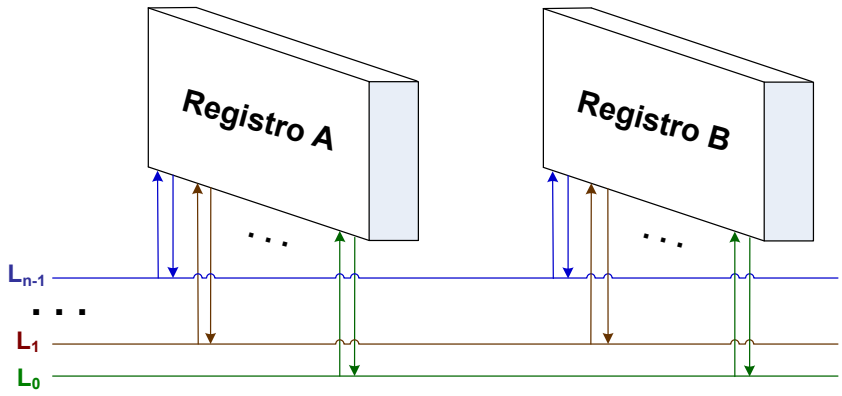
Los sistemas digitales complejos *(p.e. los microprocesadores)* están formados por múltiples elementos de almacenamiento *(registros)* y circuitos combinacionales.

¿Cómo interconectar todos estos dispositivos? *Ej. Conexión 4 registros:*



Si nº de elementos ↑, entonces la complejidad de las conexiones sería inadmisible → ***Solución***: **Interconexión mediante BUS.**

- **CONCEPTO DE BUS**: conjunto de 'n' cables o líneas eléctricas de transmisión, que permiten transportar datos de 'n' bits entre múltiples dispositivos.

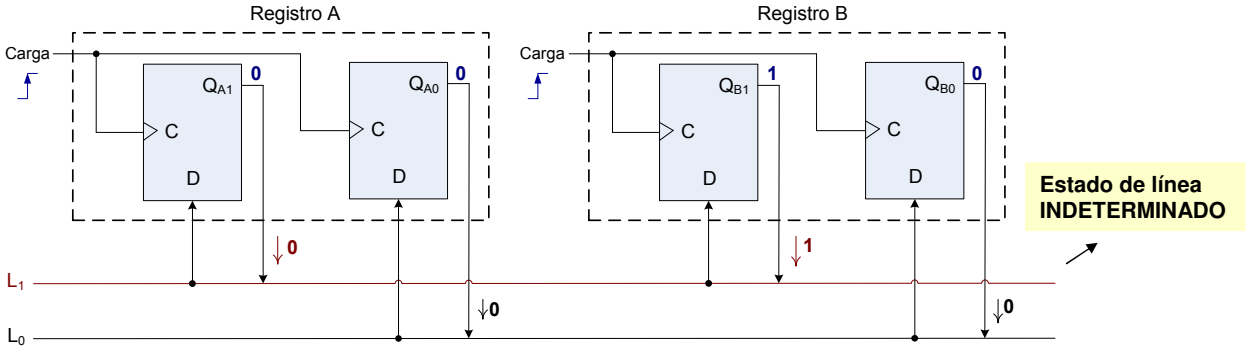


Bus de 'n' líneas
- L_0: línea de peso 0 → transporta el bit de peso 0
- L_1: línea de peso 1 → transporta el bit de peso 1
- . . .
- L_{n-1}: línea de peso n-1 → transporta el bit de peso 'n-1'

- **Nueva Problemática**: ¿Cómo gestionar el acceso al Bus?

  Todos los registros están conectados al mismo bus, y todos pueden mover datos a través de ellas → un único camino para múltiples dispositivos.

  Ejemplo gráfico: { Bus de 2 bits.
  Dispositivos conectados: dos registros de 2 bits.



***Solución*: utilización de registros triestado.**

- **LÓGICA TRIESTADO**

  Los dispositivos de tipo triestado son aquellos que pueden trabajar con 3 estados posibles:

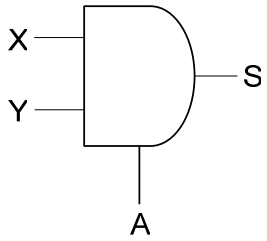
  { ALTO → Equivale al "1" lógico.
  BAJO → Equivale al "0" lógico.
  ALTA IMPEDANCIA (Z) → Equivale a Desconectado.

  Los dispositivos triestado disponen de una entrada especial denominada *Entrada de Habilitación* (A), de forma que:

  { Si A=1 → el elemento triestado funciona normalmente, apareciendo en su salida el nivel lógico "0" o "1".

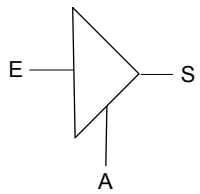
  Si A=0 → la salida del elemento triestado se pondrá en estado de Alta Impedancia (Z).

Ejemplo: Puerta AND triestado



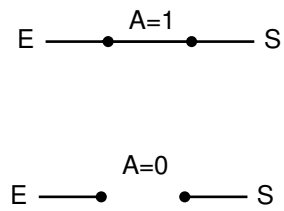
A	X	Y	S	
1	0	0	0	Si A=1 → salida de una puerta AND normal
1	0	1	0	
1	1	0	0	
1	1	1	1	
0	-	-	Z	Si A=0 → Salida en Alta Impedancia, independientemente del valor de las entradas.

Concepto de Buffer Triestado: dispositivo que permite convertir la salida de cualquier otro dispositivo en triestado.



Si A=1 → S = E

Si A=0 → S = Z



(Estado de Alta Impedancia)

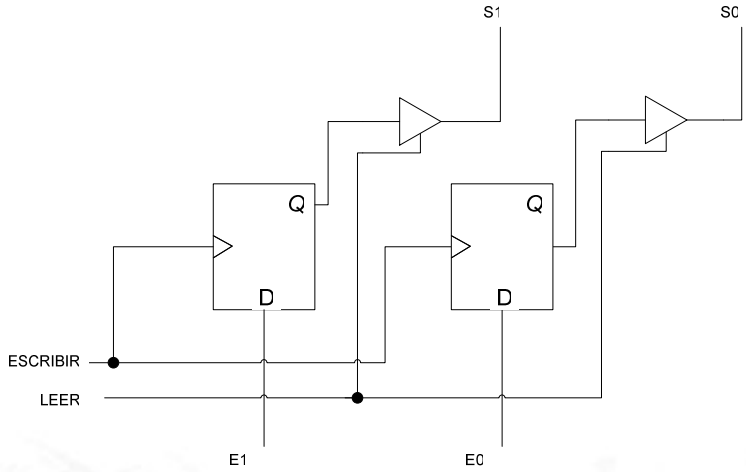
A	E	S	
1	0	0	
1	1	1	
0	-	Z	

Utilizando este tipo de dispositivo pueden construirse **registros triestado**.

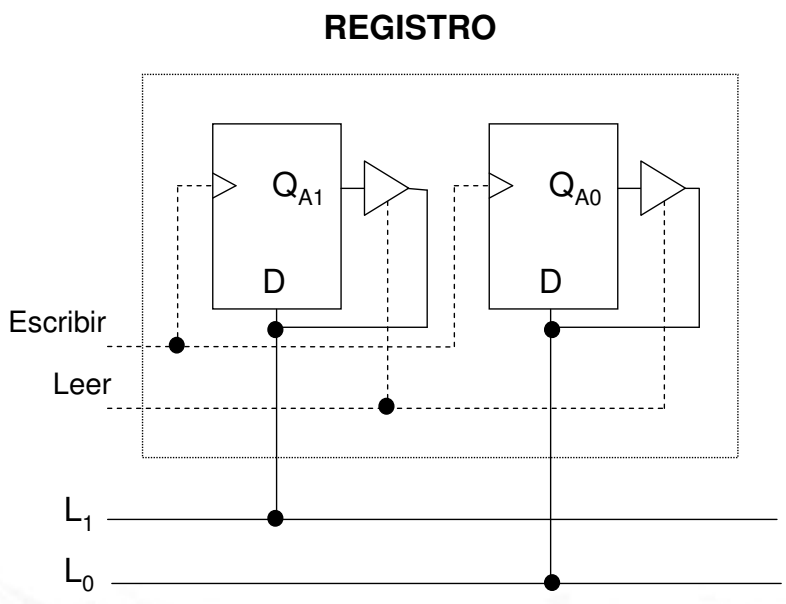
- Ejemplo de registro triestado de 2 bits (1 de 2)

Señal LEER → Activa/desactiva los buffer triestado.

- Si LEER=1, entonces los valores E_0 y E_1 pasarán a S_0 y S_1 respectivamente cuando se produzca un flanco ascendente en la señal ESCRIBIR (E_0=S_0, E_1=S_1).
- Si LEER=0, entonces los buffer triestado quedarán desconectados (→ S_0= S_1= Z).

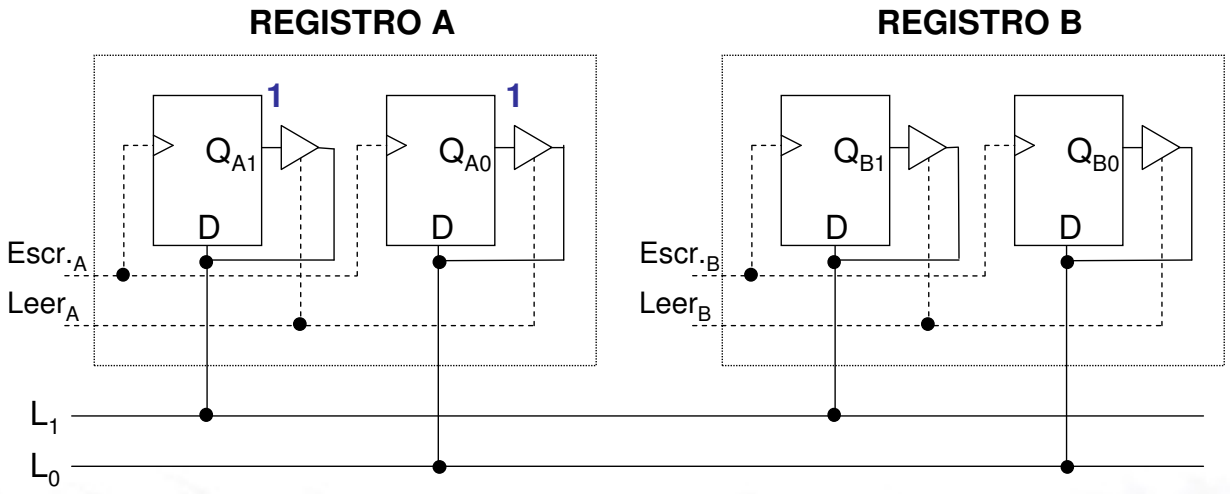


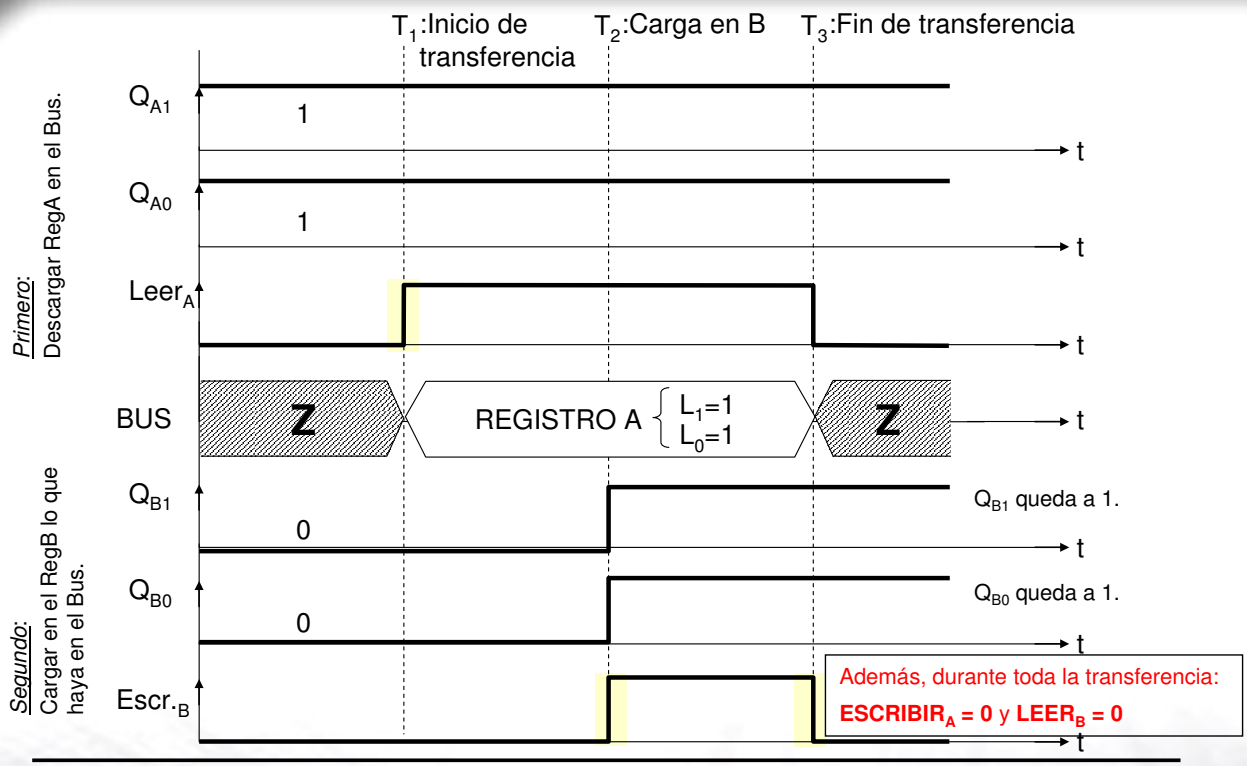
- Ejemplo de registro de 2 bits (2 de 2)



- Ejemplo de transferencia en un bus
  - Bus de 2 bits.
  - Dispositivos conectados: dos registros de 2 bits.

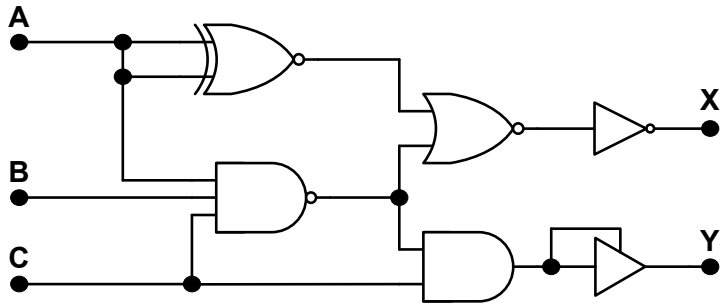
Sup. que $Q_{A1}= Q_{A0}= 1$ y queremos pasarlo al Reg.B





Ejercicio propuesto:

Dado el circuito de la figura, rellenar los valores de X e Y de la tabla.

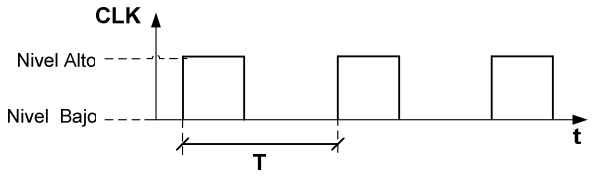


A	B	C	X	Y
0	1	0		
1	0	1		

Solución:

A	B	C	X	Y
0	1	0	1	Z
1	0	1	1	1

Concepto de Reloj: dispositivo que genera una señal periódica, cuyo periodo (T) está dividido en dos partes, una de nivel alto y otra de nivel bajo, que no tienen por qué tener la misma duración.



T → Periodo *(segundos)*

f → Frecuencia = 1/T *(hertzios=s^{-1})*

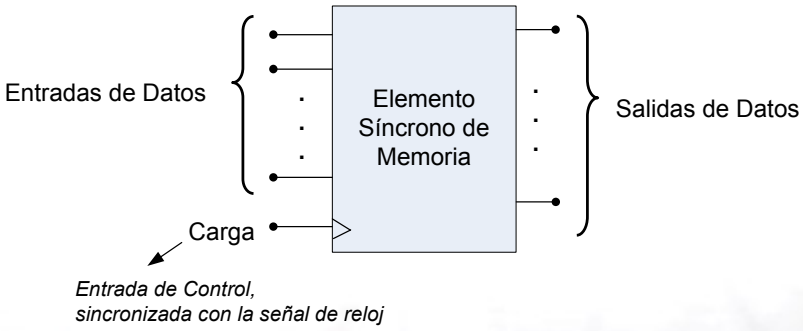
T (unidades más habituales)	*F (unidades más habituales)*
1 milisegundo (ms) = 10^{-3} segundos	1 kiloHertzio (KHz) = 10^{3} Hz
1 microsegundo (µs) = 10^{-6} segundos	1 megaHertzio (MHz) = 10^{6} Hz
1 nanosegundo (ns) = 10^{-9} segundos	1 gigaHertzio (GHz) = 10^{9} Hz

Ejercicio: Calcular el periodo de la señal de reloj que controla el funcionamiento de un Pentium a 500 MHz.

$$T = \frac{1}{f} = \frac{1}{500\,Mhz} = \frac{1}{500\,x10^{6}} = 0'2x10^{-8} = 2x10^{-9} = 2ns$$

Sincronización de dispositivos mediante señal de Reloj

- La señal de reloj se utiliza para sincronizar la evolución del estado de los elementos de memoria que forman parte de un dispositivo digital.
- Dichos elementos sólo evolucionan cuando se produzca un flanco activo de la señal de reloj.
- Los dispositivos digitales complejos (p.e. los microprocesadores) se diseñan siempre para ser gobernados mediante una señal de reloj, lo cual facilita tanto su diseño como su verificación.



- **Conceptos sobre memorias**
  - REGISTROS: Se emplean para almacenar pequeñas cantidades de información digital (como máximo, del orden de decenas de bytes).
  - MEMORIAS:
    - Utilizadas para almacenar grandes cantidades de información digital (orden de KB, MB, GB, TB,…).
    - Se utilizan para confeccionar el sistema de memoria (el cual se verá más adelante).
    - Interfaz del chip de memoria: Una memoria con 'a' líneas de dirección y 'n' líneas de datos se dice que es una memoria de organización **m**x**n**, donde m=2ª.

      *Por ejemplo, una memoria con 10 líneas de direcciones y 8 de datos es una memoria **1Kx8**. (1K=2^{10}).*

Organización lógica de una memoria:

- Una memoria se organiza en palabras de un determinado ancho:
  * El nº de palabras es una potencia de 2, y se denota por ***m***.
  * El ancho de una palabra es su nº de bits, y se denota por ***n***.
- La organización de una memoria se expresa de la forma **mxn**.
- Para hacer referencia a cada palabra de una memoria se utiliza el nº binario de 'a' bits, al que se denomina DIRECCIÓN. Se debe cumplir que m = 2ª.
- El CONTENIDO de una palabra o posición de memoria es la información que almacena.

¡Ojo! No confundir CONTENIDO con DIRECCIÓN de memoria.

Ejemplo: Organización de una memoria 8x4.

8 x 4 → 8 palabras, cada una de ellas de 4 bits.

	bit3	bit2	bit1	bit0	
0 0 0					**Palabra 0**
0 0 1					**Palabra 1**
0 1 0					**Palabra 2**
0 1 1					**Palabra 3**
1 0 0	**0**	**0**	**1**	**0**	**Palabra 4**
1 0 1					**Palabra 5**
1 1 0					**Palabra 6**
1 1 1					**Palabra 7**

$m = 8 = 2^3$ → a=3
n = 4 bits

Ejemplo: La dirección de memoria *100* contiene *0010*.

- **CLASIFICACIÓN DE LAS MEMORIAS:**

Diversos criterios:

- Posibilidad de Escritura: Todas las memorias pueden leerse, pero "no todas pueden escribirse" (**ROM**). Se dice que son memorias de SÓLO LECTURA las *escribibles* sólo una vez o las que requieren un equipo especializado para escritura (grabador).
- Volatibilidad: Actualmente, las memorias de los computadores están implementadas con dispositivos electrónicos que necesitan ser alimentados con una tensión eléctrica para funcionar. Las **memorias volátiles**, pierden la información al faltar la alimentación (todas las memorias de sólo lectura son no volátiles); las **memorias no volátiles**, mantienen la información aún no estando alimentadas.
- Necesidad de Refresco: Algunas memorias pierden la información almacenada si no se sobrescribe ésta con cierta frecuencia. Al proceso de sobre-escritura se le denomina **Refresco**. Normalmente, el refresco lo hace cierta lógica integrada dentro de la misma memoria. Las memorias de sólo lectura no necesitan refresco.
- Tipo de Acceso: Aleatorio (**RAM**) o secuencial.

- **CLASIFICACIÓN DE LAS MEMORIAS:**

1) Memoria ROM *(Read Only Memory)*

- Memoria de sólo lectura.
- Celda básica: DIODO.

2) Memoria RAM *(Random Access Memory)*

- Memoria de Lectura y Escritura.
- Tipos: SRAM (Static RAM) – Celda básica: BIESTABLE.
  DRAM (Dynamic RAM) – Celda básica: CONDENSADOR.

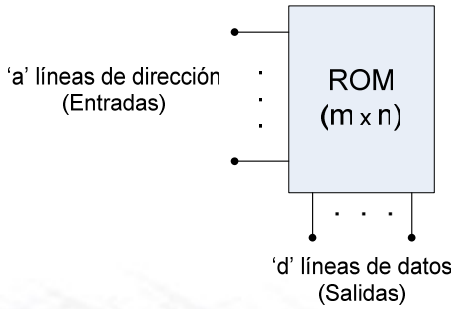
Concepto de Celda Básica: elemento que tiene la capacidad de almacenar 1 bit de información.

Funcionalidad: almacenar información digital de forma permanente y no modificable.

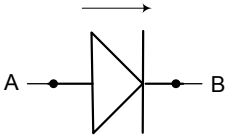
Esquema de Entradas y Salidas:

Dada una organización de mxn: $\begin{cases} m = 2^a \\ n = d \end{cases}$

m → nº de palabras.
a → nº de líneas de dirección.
n → ancho de la palabra (=nº de bits).
d → nº de líneas de datos.



Celda básica: DIODO



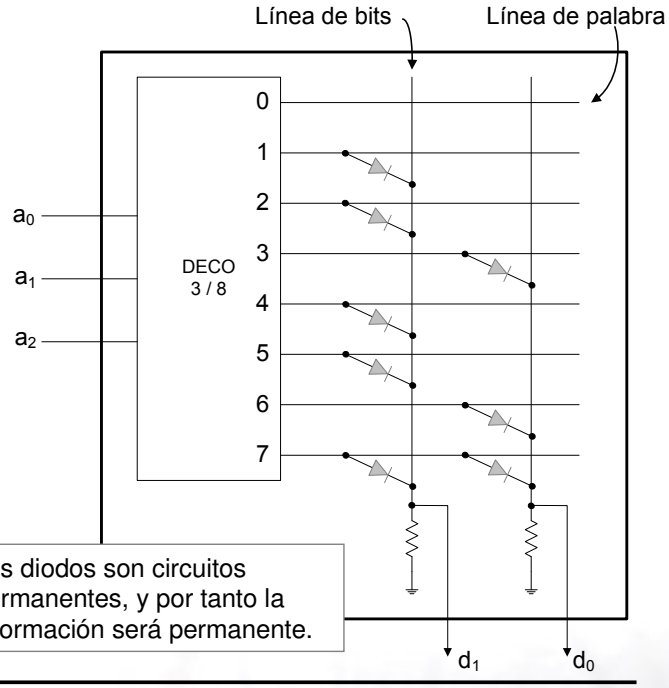
Si A=1 → Transmite el "1" lógico en la dirección de la flecha.
Si A=0 → El diodo queda en estado de Alta Impedancia.

Estructura Lógica

Direccs

$a_2\,a_1\,a_0$	d1	d0	
0 0 0	0	0	Palabra 0
0 0 1	1	0	Palabra 1
0 1 0	1	0	Palabra 2
0 1 1	0	1	Palabra 3
1 0 0	1	0	Palabra 4
1 0 1	1	0	Palabra 5
1 1 0	0	1	Palabra 6
1 1 1	1	1	Palabra 7

Estructura Física

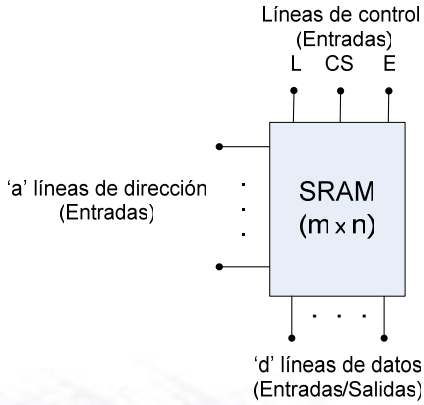


Los diodos son circuitos permanentes, y por tanto la información será permanente.

Funcionalidad: almacenar información digital de forma temporal y modificable.

Esquema de Entradas y Salidas:

Dada una organización de m x n: { m = 2ª ; n = d ; 3 líneas de control { L = Leer palabra del chip ; E = Escribir palabra ; CS = Chip Select }



Celda básica: BIESTABLE

Características:
- Costoso de implementar (un biestable ocupa mucho espacio en el área de silicio sobre el que se construye).
- Alto consumo de potencia.
- Respuesta rápida.

Funcionamiento del chip:

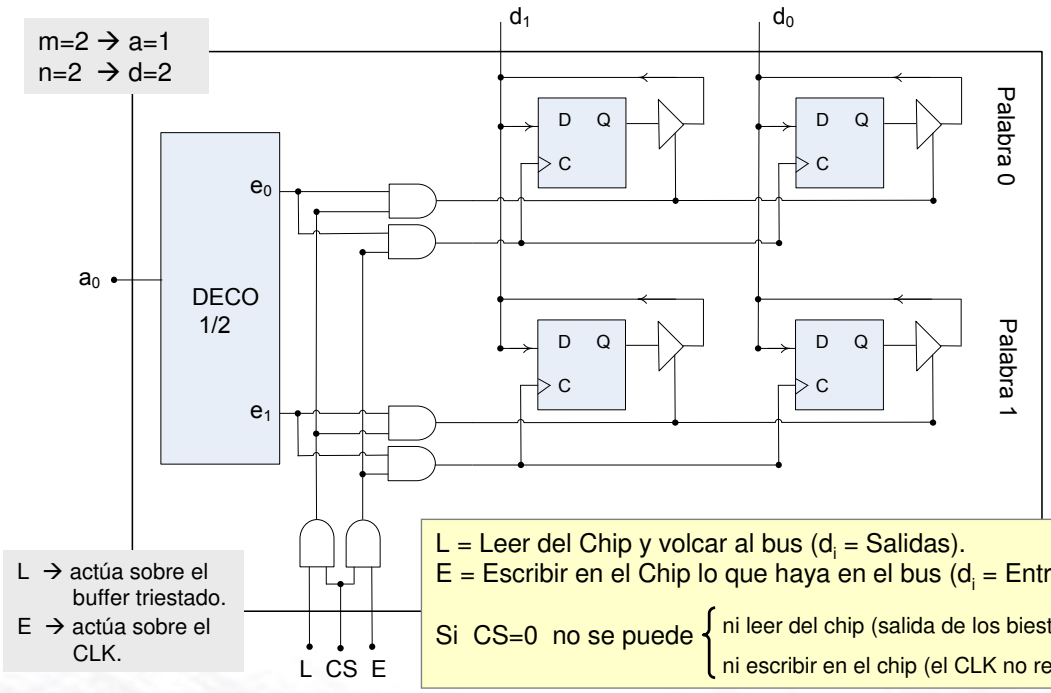
- Cuando se accede al chip, se accede a una sola palabra.
- La palabra a la que se accederá será la que se indique en las líneas de dirección del chip.
- La operación que se desea hacer sobre la palabra (Lectura o Escritura) se indica mediante las entradas de control (CS, L, E).

Si CS = 0 → chip deshabilitado (da igual el estado de L y E).

Si CS = 1 → chip habilitado { para lectura (si L=1). ; para escritura (si E=1). }

- Las líneas de datos podrán actuar como { Salidas (en operaciones de Lectura) o Entradas (en operaciones de Escritura) }

Estructura interna *(Ej. 2x2)*



L = Leer del Chip y volcar al bus (d_i = Salidas).
E = Escribir en el Chip lo que haya en el bus (d_i = Entradas).

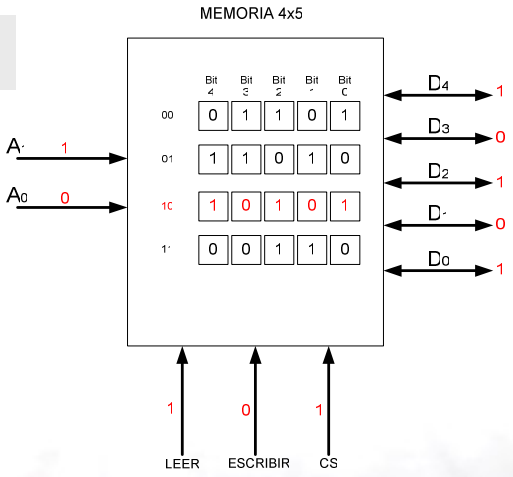
Si CS=0 no se puede { ni leer del chip (salida de los biestables cortada). / ni escribir en el chip (el CLK no recibe flancos).

LEER UNA PALABRA DEL CHIP

1) Colocar en las líneas de dirección del chip (A_i) la dirección de la palabra a la que se quiere acceder.
2) Activar CS y L *(CS=1, L=1).*
3) Los valores contenidos en los bits de la palabra indicada por la combinación A_i son volcados en las respectivas líneas D_i.

Mientras CS=1 y L=1 la información estará disponible en el Bus (de donde la leerá la CPU).

Ejemplo:



ESCRIBIR EN UNA PALABRA DEL CHIP

1) Colocar en las líneas de datos del chip (D_i) el dato que se quiera escribir.
2) Colocar en las líneas de dirección del chip (A_i) la dirección de la palabra en la que se quiere escribir.
3) Activar CS y E *(CS=1, E=1)*.
4) Los valores contenidos en las líneas D_i son volcados en los bits respectivos de la palabra indicada por la combinación A_i.

Ejemplo:

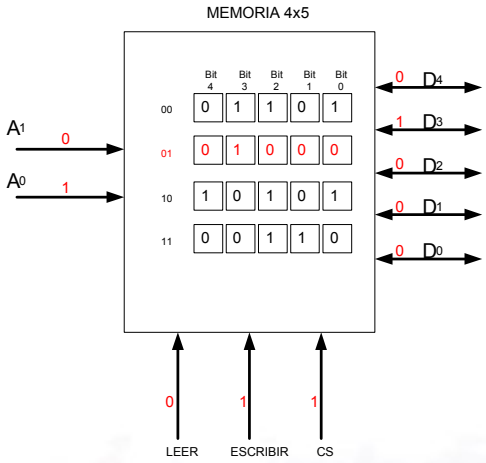
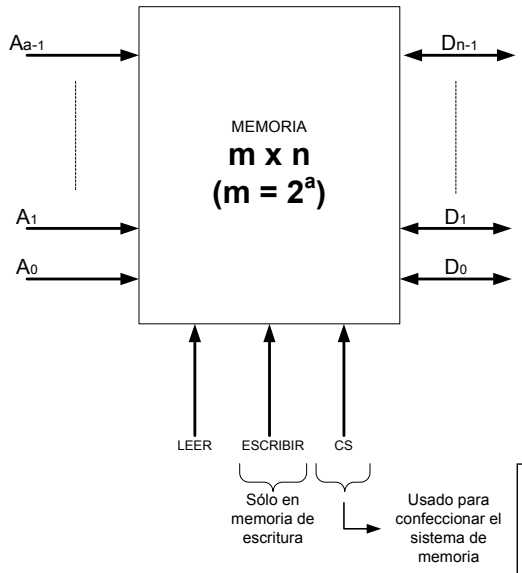


Gráfico Resumen del Interfaz del Chip de Memoria



CS: Cuando está a '0', todas las entradas y salidas del chip de memoria están desconectadas del entorno.

A_0 ... A_{a-1} : m líneas de dirección (ENTRADAS)
SELECCIÓN DE UNA PALABRA
Contenido del chip: 2^a = m palabras

D_0 ... D_{n-1} : n líneas de datos
LÍNEAS DE ENTRADA/SALIDA
n = tamaño de palabra

LEER: Línea de entrada. Controla lectura de la memoria (análoga a la línea de leer de los registros). Cuando vale '1', los n bits almacenados en la palabra ubicada en la posición $A_{a-1}\ A_{m-2}\ \ldots\ A_1\ A_0$ se lleva a las líneas $D_{n-1}\ D_{n-2}\ \ldots\ D_1\ D_0$

ESCRIBIR: Línea de entrada. Controla la escritura de la memoria (análoga a la línea de escribir de los registros). Un flanco ascendente (↑) hace que la secuencia de bits que se encuentra en la líneas $D_{n-1}\ D_{n-2}\ \ldots\ D_1\ D_0$ pase a estar almacenada en la palabra ubicada en las posiciones $A_{a-1}\ A_{m-2}\ \ldots\ A_1\ A_0$

- **Algunos de los tipos de memoria en el mercado:**
  - **ROM (Read Only Memory)**: memoria de sólo lectura.
    La información almacenada se fija durante el proceso de fabricación, no pudiendo ser modificada posteriormente.
  - **PROM (Programmable ROM)**: memoria de sólo lectura.
    La información almacenada se fija durante el proceso de escritura, el cual requiere un grabador. Una vez grabada la información, no puede cambiarse.
  - **EPROM (Erasable PROM)**: memoria de sólo lectura.
    La información almacenada se fija durante el proceso de escritura, el cual requiere un grabador. Una vez grabada la información, ésta puede cambiarse, sometiendo previamente la memoria a luz ultravioleta.
  - **EEPROM (Electrically EPROM)**: Análoga a la EPROM, pero no es necesario aplicar luz ultravioleta sobre la memoria para poder cambiar la información almacenada. Requiere un programador.

  - **FLASH**:
    * Evolución de las EEPROM, aumentando la velocidad y reduciendo el coste de las memorias EEPROM tradicionales.
    * La programación se puede hacer sin necesidad de un programador, dentro del mismo sistema en el que son empleadas.
    * Graban y leen en bloques fijos (512 B – 256 KiB).
    * Son memorias no volátiles, lo que las hace muy útiles en dispositivos electrónicos de consumo como cámaras fotográficas, vídeo consolas, teléfonos digitales, etc.
  - **SRAM**: RAM estática.
    * Memorias de L/E, volátiles y sin necesidad de refresco.
    * Emplean biestables para almacenar los bits.
  - **DRAM**: RAM dinámica.
    * Memorias de L/E, volátiles, basadas en condensadores y, por ello, necesitan refresco.

- Memorias más empleadas en computadores: **SRAM**, **DRAM** y **FLASH**.
- Comparativa según ciertas características de diseño:

Velocidad ↑	**Capacidad de almacenamiento ↑**	**Coste/Bit ↑**
SRAM	DRAM	SRAM
DRAM	FLASH	DRAM
FLASH	SRAM	FLASH

- *SRAM*: Máxima velocidad (→ se utiliza en Memoria caché).
- *DRAM*: Coste/bit más bajo (→ se utiliza en Memoria principal).
- *FLASH*: Memoria no volátil (→ se utiliza en la BIOS del computador).

- **Parámetros temporales**
  - Como todos los sistemas reales, **las memorias tardan un tiempo en responder**. Es decir, desde que se pone un valor en las entradas hasta que aparece el valor correcto en las salidas, en el caso de una lectura, o cambia el estado de una palabra de la memoria, en el caso de una escritura, transcurre un tiempo.
  - ***OBJETIVO***: minimizar ese tiempo.
  - Se suelen definir **dos parámetros temporales** asociados a las memorias:
    - ***Tiempo de Lectura***: Tiempo transcurrido desde que se pone una dirección en las líneas de dirección y se activa la línea LEER (↑) hasta que la memoria pone la palabra asociada a la dirección en las líneas de datos.
    - ***Tiempo de Escritura***: Tiempo transcurrido desde que se pone una dirección en las líneas de dirección y se activa la línea ESCRIBIR (↑) hasta que la memoria escribe el dato en la palabra asociada a la dirección.
  - Habitualmente, se utiliza como parámetro temporal el ***Tiempo de Acceso***, que es la media del *Tiempo de Lectura* y del *Tiempo de Escritura*.
  - El tiempo de acceso de las memorias actuales más habituales varía entre unos pocos nanosegundos y unos milisegundos.

Ejercicios propuestos:

1. **Se tiene una memoria ROM con organización 8x32. ¿Qué tipo de decodificador lleva?**

2. **¿Cuántos diodos hay en una memoria ROM que implementa la funcionalidad de un sumador elemental?**

3. **¿Cuántos biestables D hay en una memoria SRAM de 4Mx8? Contestar en potencias de 2.**