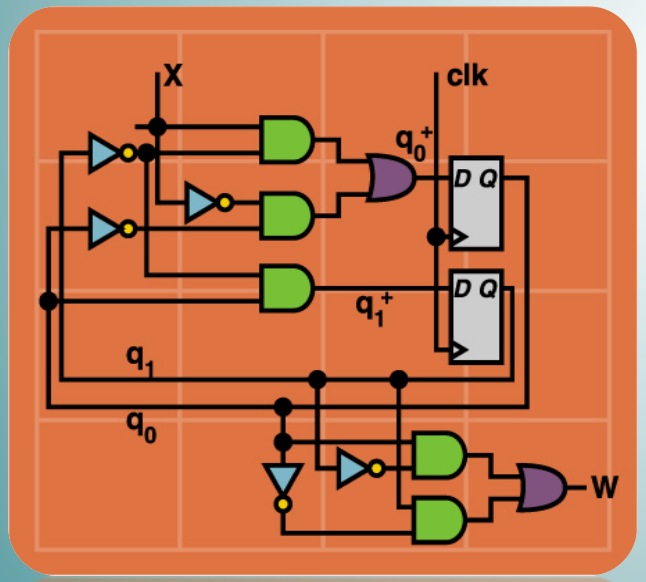
Sistemas Digitales

Tema 3. Circuitos Lógicos Combinacionales

«Digital Design and Computer Architecture» (Harris & Harris). Chapter 1 (1.5 - 1.7) / Chapter 2 (2.1 - 2.9)



Pablo Abad
Pablo Prieto Torralbo

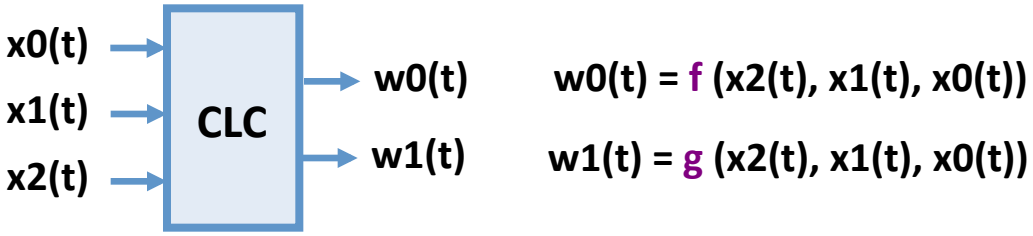
Departamento de Ingeniería Informática y Electrónica

Índice

- **Introducción:**
  - Definición de CLC, Modelo matemático.
  - Del transistor a la puerta lógica.
  - Puertas Lógicas.
  - CLCs jerárquicos.
- **Análisis Temporal:**
  - Tiempo de propagación.
  - Glitches.
- **Álgebra de conmutación:**
  - Axiomas y teoremas.
  - Expresiones/Circuitos equivalentes.
- **Análisis y Síntesis:**
  - Suma de Minterms.
  - Decodificador + OR.
  - ROM.

Introducción

- **Circuito Lógico Combinacional (CLC):**
  - Definición: circuito encargado de procesar (transformar) las señales binarias (información digital).
  - Se puede representar como una «caja negra» (abstracción) con los siguientes componentes:
    - Una ó más entradas (señales binarias).
    - Una ó más salidas (señales binarias).
    - Funcionalidad describiendo la relación entre entradas y salidas.
    - Timing: determina el retraso entre el cambio de una entrada y la respuesta de una salida.



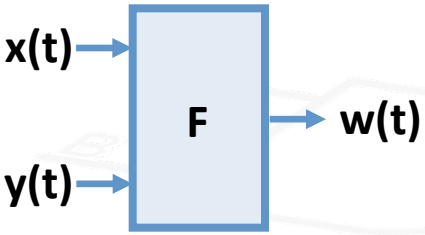
$w0(t) = f(x2(t), x1(t), x0(t))$

$w1(t) = g(x2(t), x1(t), x0(t))$

* **x0, x1, x2, w1 y w2 son señales eléctricas binarias únicamente con dos posibles valores: 0 y 1 voltio (1980: 5 v.; 2011: 1 v).**

Introducción

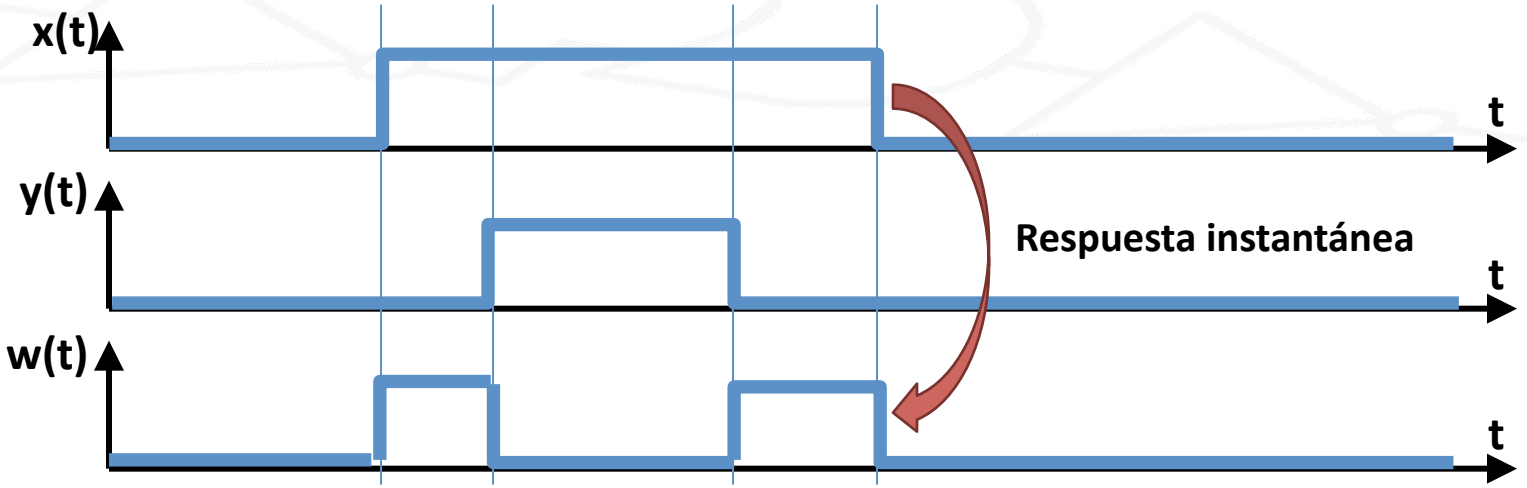
- **Circuito Lógico Combinacional (CLC):**
  - Un ejemplo concreto:



w(t) = F (x(t), y(t))

Formulación exhaustiva

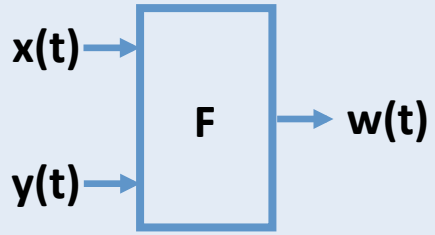
y(t)	X(t)	W(t)
0 v	0 v	0 v
0 v	1 v	1 v
1 v	0 v	0 v
1 v	1 v	0 v



Introducción

- **Circuito Lógico Combinacional (CLC):**
  - Modelado Matemático.

Señales eléctricas binarias
x(t), y(t), w(t)

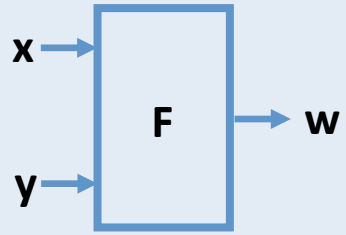


w(t) = F (x(t), y(t))

Formulación exhaustiva

y(t)	X(t)	W(t)
0 v	0 v	0 v
0 v	1 v	1 v
1 v	0 v	0 v
1 v	1 v	0 v

Variables Lógicas
x, y, w ∈ {0,1}

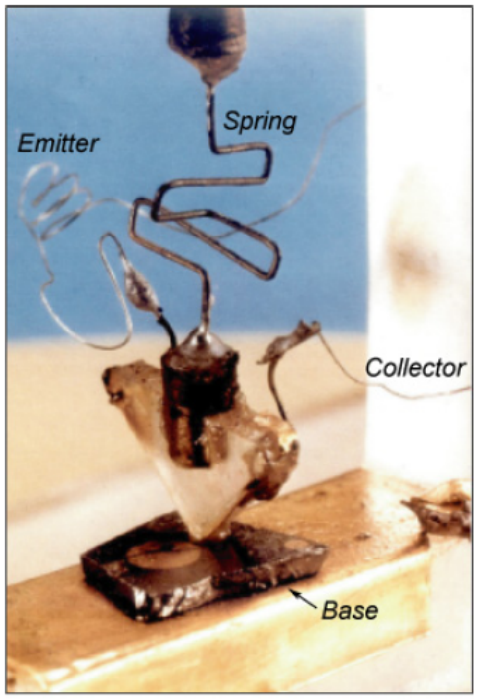


w = F (x, y) *func. lógica*

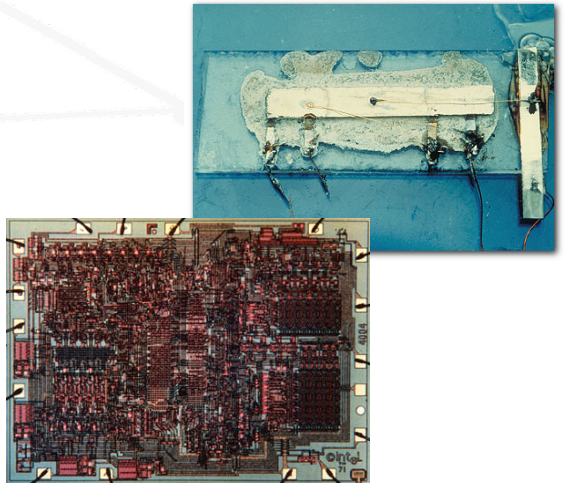
Tabla de Verdad

x	y	w
0	0	0
0	1	1
1	0	0
1	1	0

Introducción



- **Del Transistor a la Puerta Lógica:**
  - El transistor fue inventado en 1947 por tres físicos Americanos (John Bardeen, William Shockley, Walter Brattain) en los laboratorios de Bell Telephone. Ganaron el premio Nobel de física en 1956 por su hallazgo.
  - Esto hizo posible, en 1958, la fabricación del primer circuito integrado por Jack Kilby en Texas Instruments (1971, primer microprocesador comercial, Intel 4004).
  - Sin lugar a dudas, estos son algunos de los hechos más relevantes en electrónica en el siglo XX.

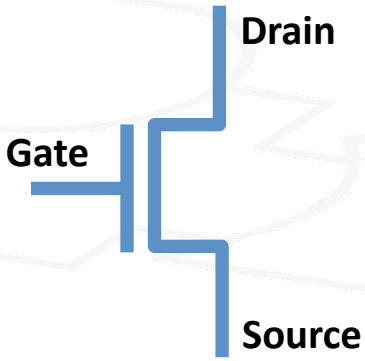


Introducción

- **Un poco de Física:**
  - Para entender cómo funciona un transistor, tendremos que «descender hasta el átomo» (casi).
  - Lo haremos de manera «informal» (lenguaje coloquial).
  - Los expertos (Fundamentos Físicos, Tema 5) arrojarán mucha más luz sobre todo esto.
  - Orden lógico (cronológico): Fundam. Físicos -> Sistemas Digitales.

Introducción

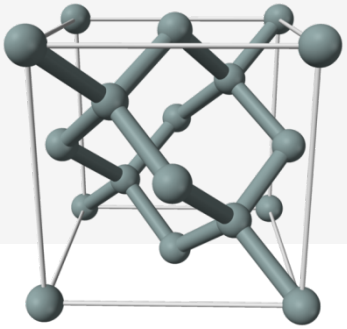
- **El Transistor:**
  - Dispositivo electrónico que conmuta entre aislante y conductor al aplicar un nivel de tensión en uno de sus terminales (puerta).
  - Esquema Básico:



Actúa como un circuito cerrado (conduce) entre Drenador y Fuente si se aplica una tensión positiva en la Puerta.

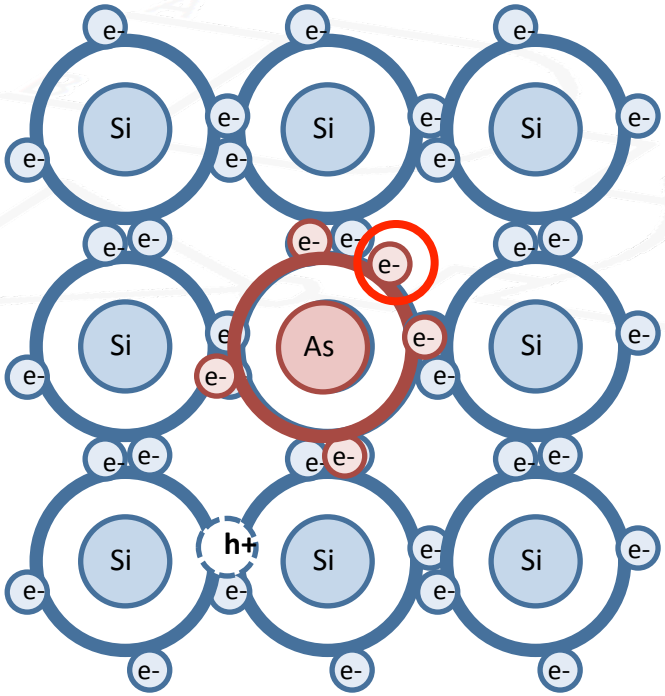
  - Dos tipos básicos (hay más):
    - Transistor Bipolar (BJT): Utilizado ámpliamente en electrónica analógica.
    - **Transistor de Efecto de Campo (FET, MOSFET, MOS): empleado en electrónica digital (ICs).**

Introducción



- **Semiconductores:**
  - Los transistores MOS se construyen de Silicio, un material Semiconductor.

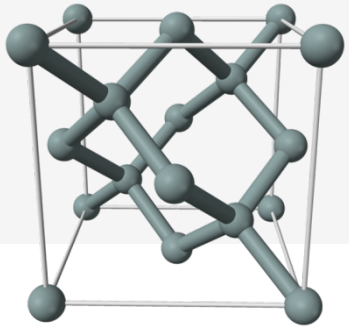
Cristal de Silicio



El silicio no es conductor, ya que todos sus electrones de último nivel forman enlaces (no hay e- libres).

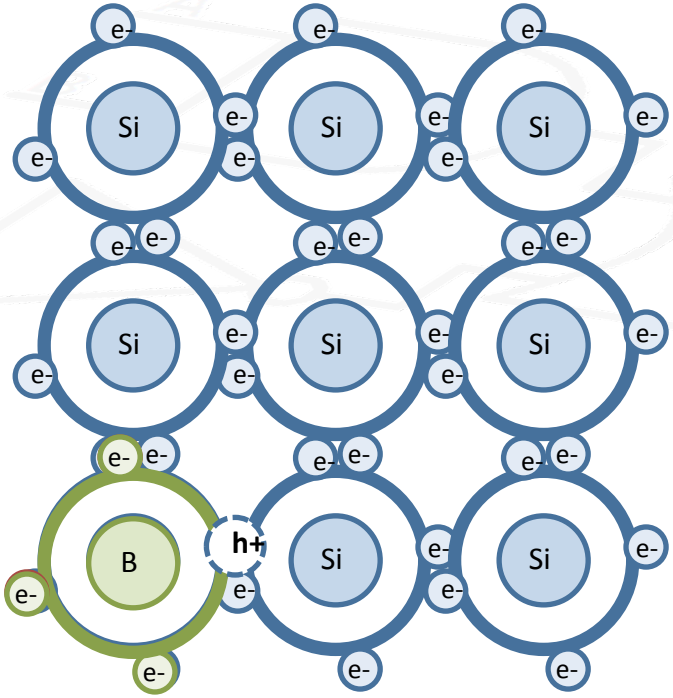
Dopado: Se añaden impurezas al silicio. Si añadimos As (grupo V), aparecen e- que no forman enlaces, y se mueven libremente. Este tipo de dopantes se denominan *tipo-n.*

Introducción



- **Semiconductores:**
  - Los transistores MOS se construyen de Silicio, un material Semiconductor.

Cristal de Silicio



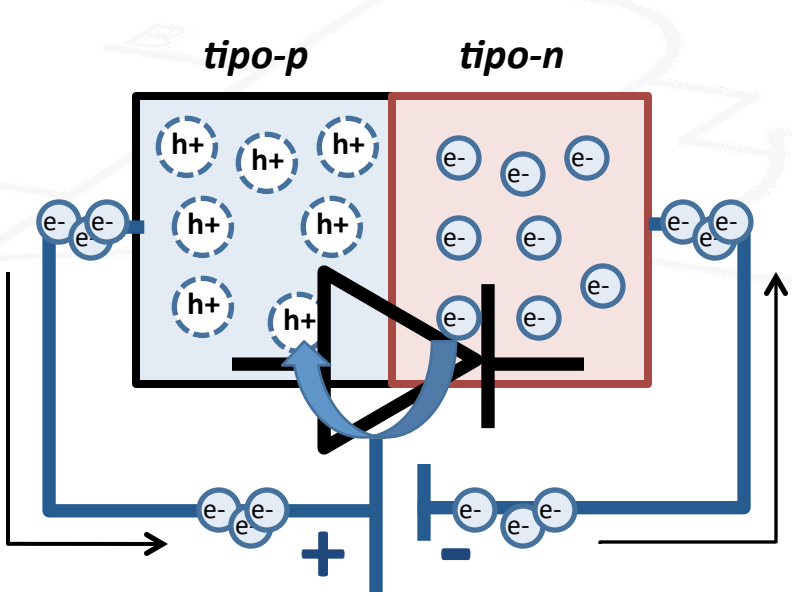
El silicio no es conductor, ya que todos sus electrones de último nivel forman enlaces (no hay e- libres).

Dopado: Se añaden impurezas al silicio. Si añadimos As (grupo V), aparecen e- que no forman enlaces, y se mueven libremente. Este tipo de dopantes se denominan *tipo-n.*

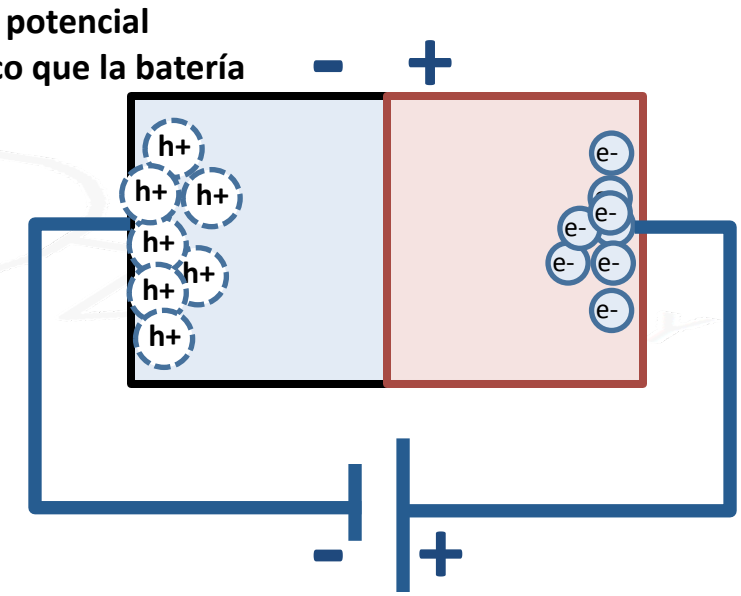
Existen también los dopantes tipo-p (B, grupo III), que generan huecos al tener menos e- en el último nivel.

Introducción

- **Unión P-N, El Diodo:**
  - La unión de un bloque de silicio *tipo-n* (dopado con As) y uno *tipo-p* se conoce como diodo. La región *tipo-p* se denomína ánodo y la *tipo-n* cátodo.
  - Su característica principal es que conducen la corriente en un solo sentido.



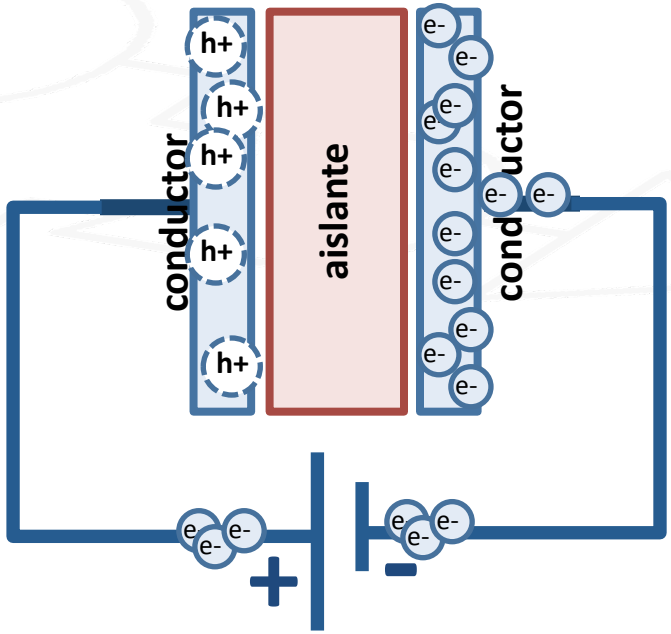
POLARIZACIÓN DIRECTA



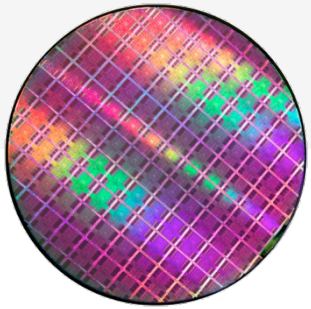
POLARIZACIÓN INVERSA

Introducción

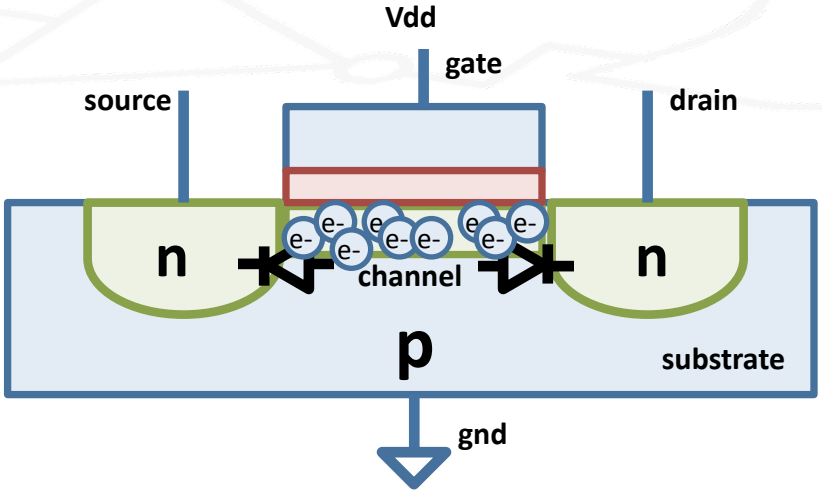
- **El Condensador:**
  - Componente eléctrico capaz de almacenar energía mediante un campo eléctrico.
  - Formado por un par de superficies conductoras separadas por un aislante.



Introducción



- **El Transistor MOS (Metal – Óxido – Semiconductor):**
  - Unión P-N-P (ó N-P-N) más un condensador.
  - Formado por una puerta de polisilicio (antes metal), una capa de óxido como aislante y un substrato de silicio dopado. En el substrato, dos regiones dopadas de forma complementaria se conectan a la fuente y al drenador.
  - Dependiendo del tipo de dopado del substrato, tenemos transistores tipo nMOS (substrato tipo-p) ó pMOS (substrato tipo-n).

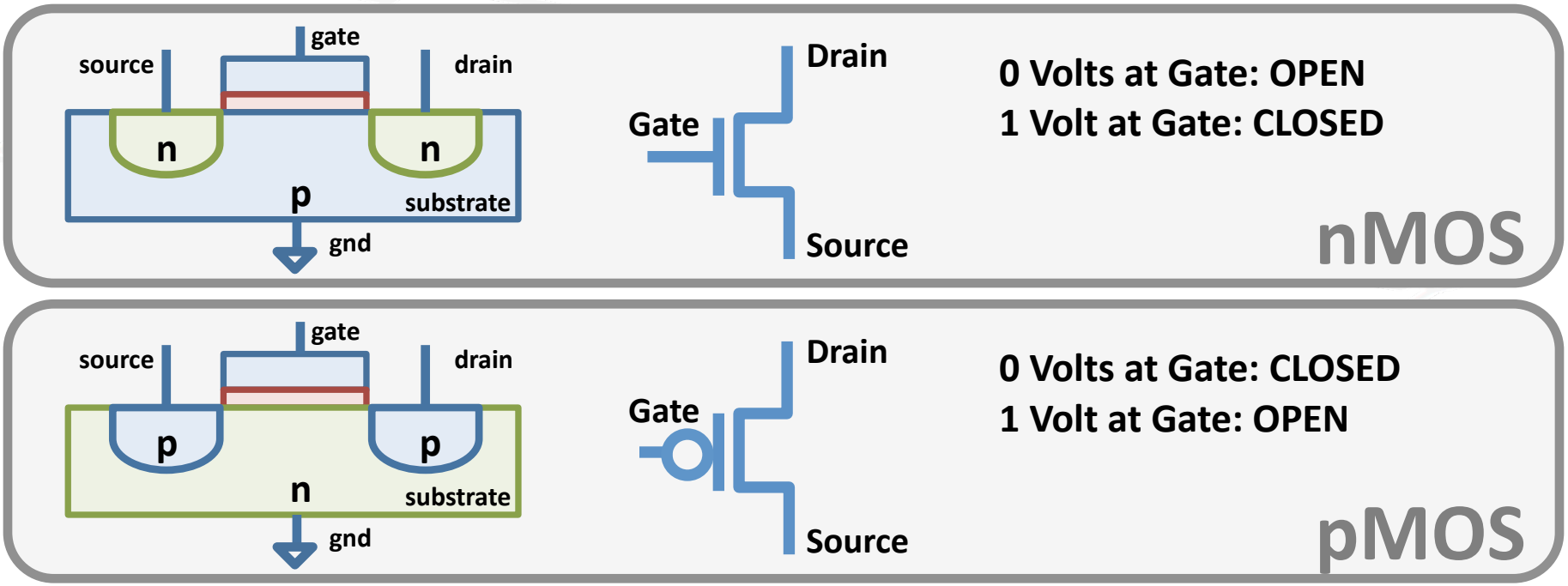


Si no se aplica tensión en la puerta, la unión n-p-n (doble diodo) no permitirá que haya corriente entre source-drain.

Aplicando tensión (Vdd) los e- son atraídos hacia uno de los bordes del condensador, creando un canal tipo n entre source y drain (conduce).

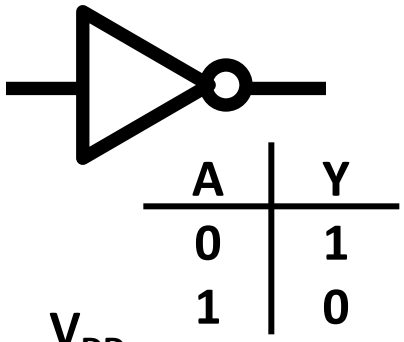
Introducción

- **CMOS: tipos complementarios (nMOS y pMOS):**
  - Los transistores tipo pMOS funcionan de forma complementaria a los nMOS.
  - pMOS conduce (entre source y drain) cuando no se aplica tensión en la puerta. Al aplicar tensión, el circuito se abre.

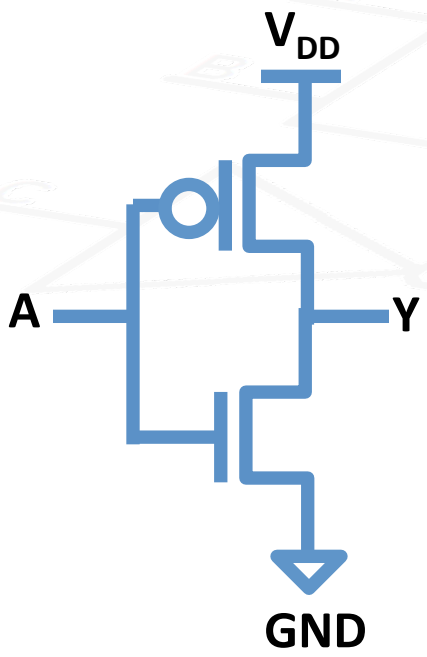


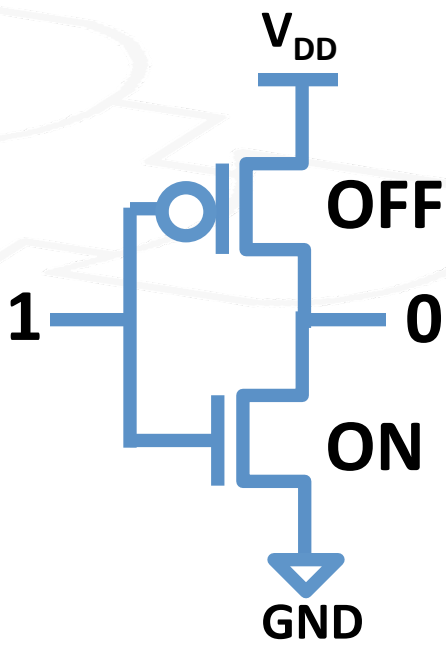
Introducción

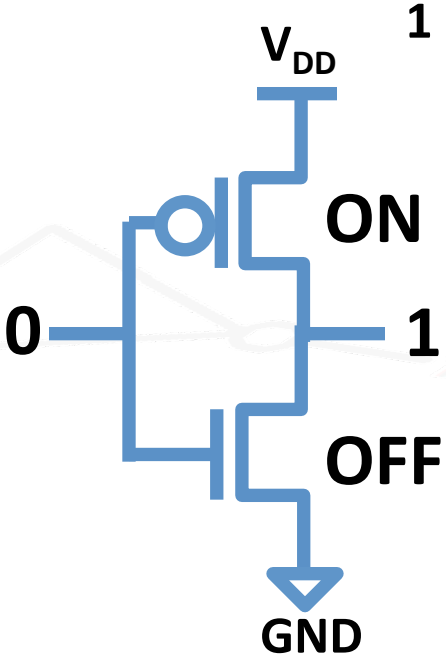
- **Nuestra primera puerta lógica: Inversor CMOS:**
  - Formada por un transistor tipo nMOS y otro tipo pMOS.
  - Invierte el valor de entrada en la salida.



A	Y
0	1
1	0

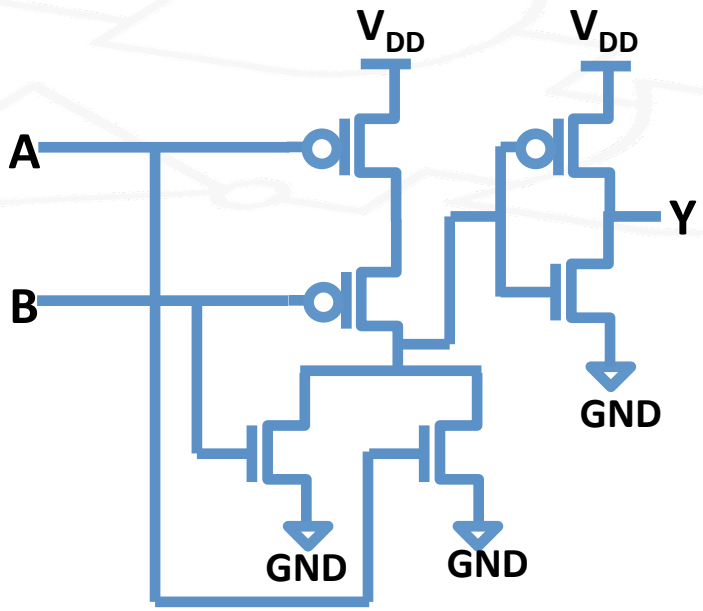






Introducción

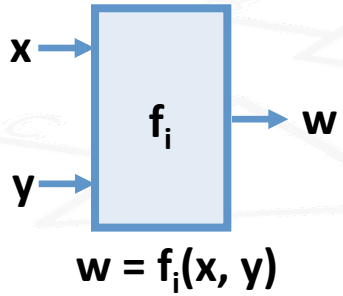
- **Del transistor a la puerta lógica:**
  - Todas las puertas lógicas con las que vamos a trabajar están construidas interconectando transistores tipo-n y tipo-p.
  - Los transistores también se utilizan para formar otros componentes Hardware, tales como las memorias RAM.



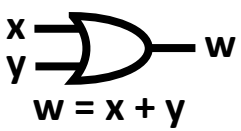
A	B	Y
0	0	
0	1	
1	0	
1	1	

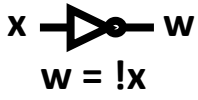
Introducción

- **Puertas Lógicas:**
  - Son los circuitos combinacionales más simples, con dos entradas y una salida (exceptuando la puerta NOT).
  - Con 2 entradas, encontramos 16 posibles funciones lógicas ().



X	Y	f_{15}	f_{14}	f_{13}	f_{12}	f_{11}	f_{10}	f_9	f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
			or						and	$x \cdot y$ nand	$x \oplus y$ xor			not		$x+y$ nor	

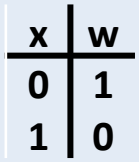
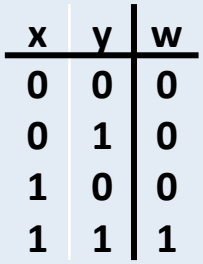
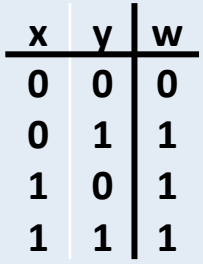




*Cualquier circuito combinacional se puede implementar interconectando adecuadamente puertas NOT, AND y OR (exclusivamente).

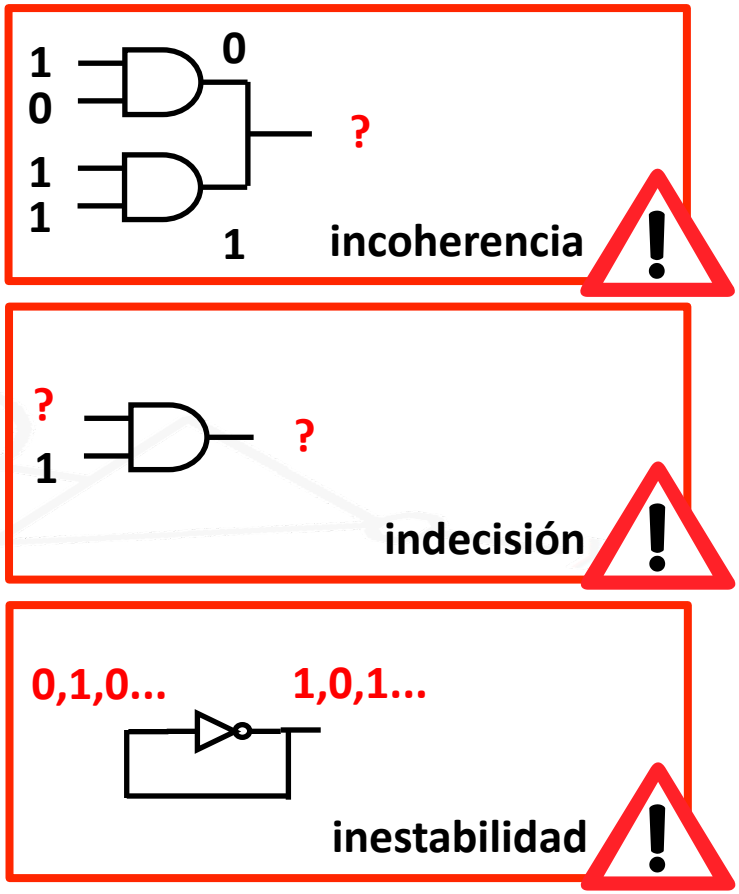
Introducción

- **Puertas Lógicas:**

Nombre	Símbolo	Expresión Lógica	Tabla de Verdad	Explicación
NOT		**w = !x**		La salida es la inversa de la entrada.
AND		**w = x · y**		La salida vale 1 cuando todas las entradas valen 1 (La salida vale 0 cuando alguna entrada vale 0).
OR		**w = x + y**		La salida vale 1 cuando alguna entrada vale 1 (La salida vale 0 cuando todas las entradas valen 0).

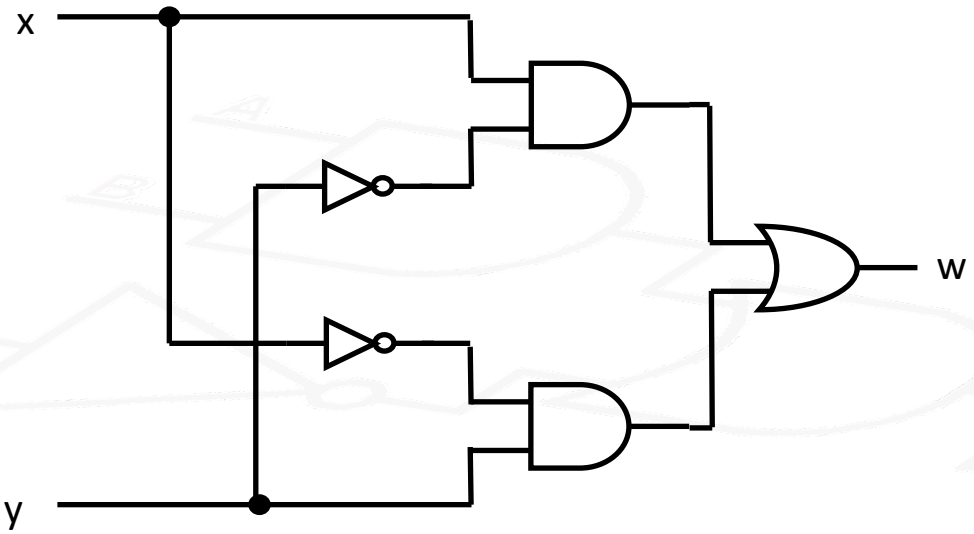
Introducción

- **Reglas de interconexión para puertas lógicas:**
  - No conectar directamente salidas entre si.
  - No dejar entradas sueltas (sin valor).
  - No ciclos.



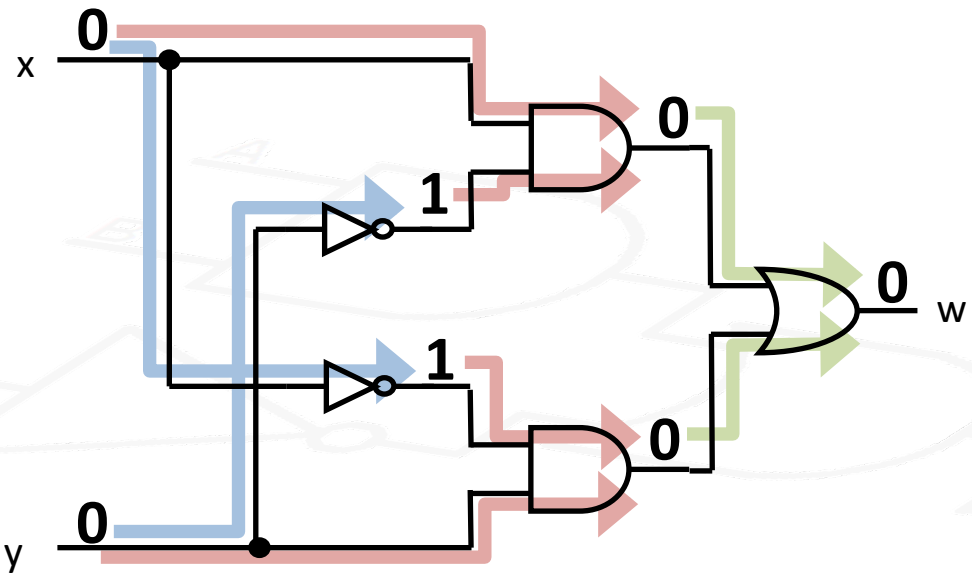
Introducción

- **Puertas + Interconexión: Nuestro Primer CLC:**



Introducción

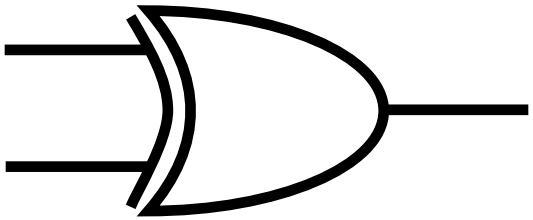
- **Puertas + Interconexión: Nuestro Primer CLC:**



Análisis Lógico: dado el esquema de un circuito lógico combinacional encontrar la tabla de verdad que especifica su funcionamiento.

x	y	w
0	0	0
0	1	
1	0	
1	1	

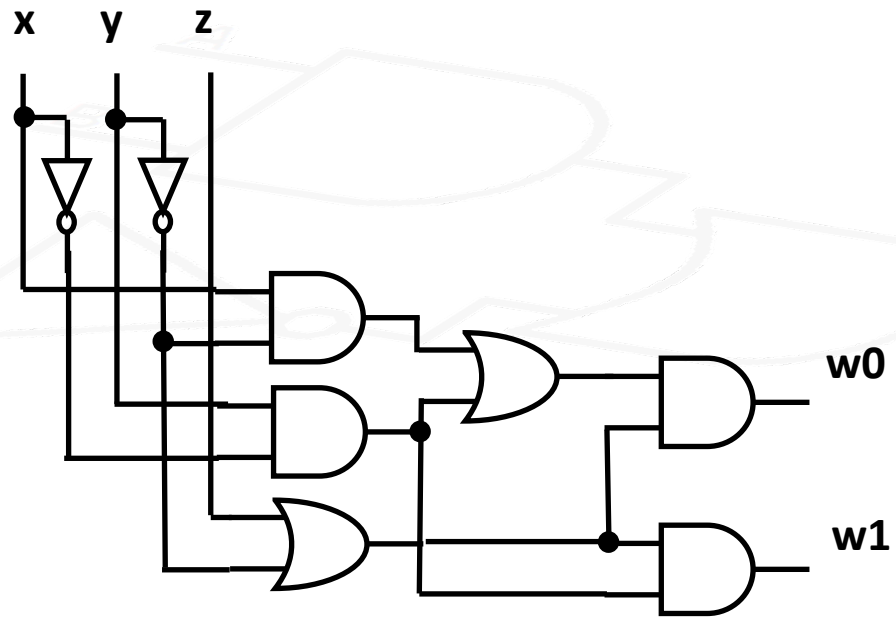
Completar



XOR: la salida vale 1 cuando hay un número impar de unos en las entradas.

Introducción

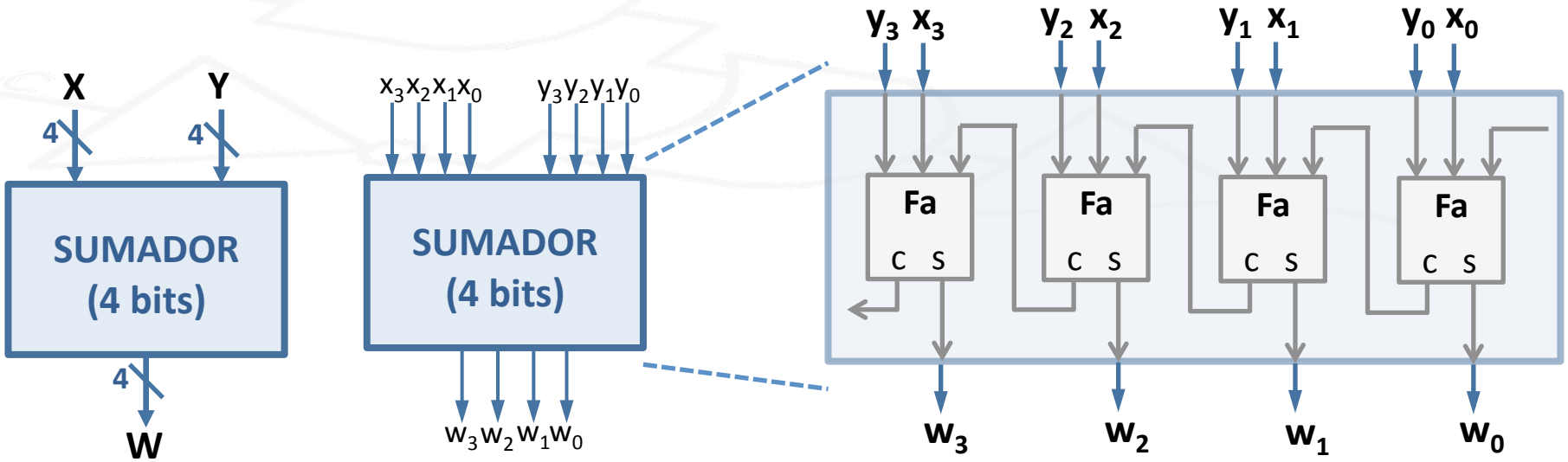
- **Ejercicio:**
  - Encontrar la tabla de verdad del siguiente CLC:



x	y	z	w0w1
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

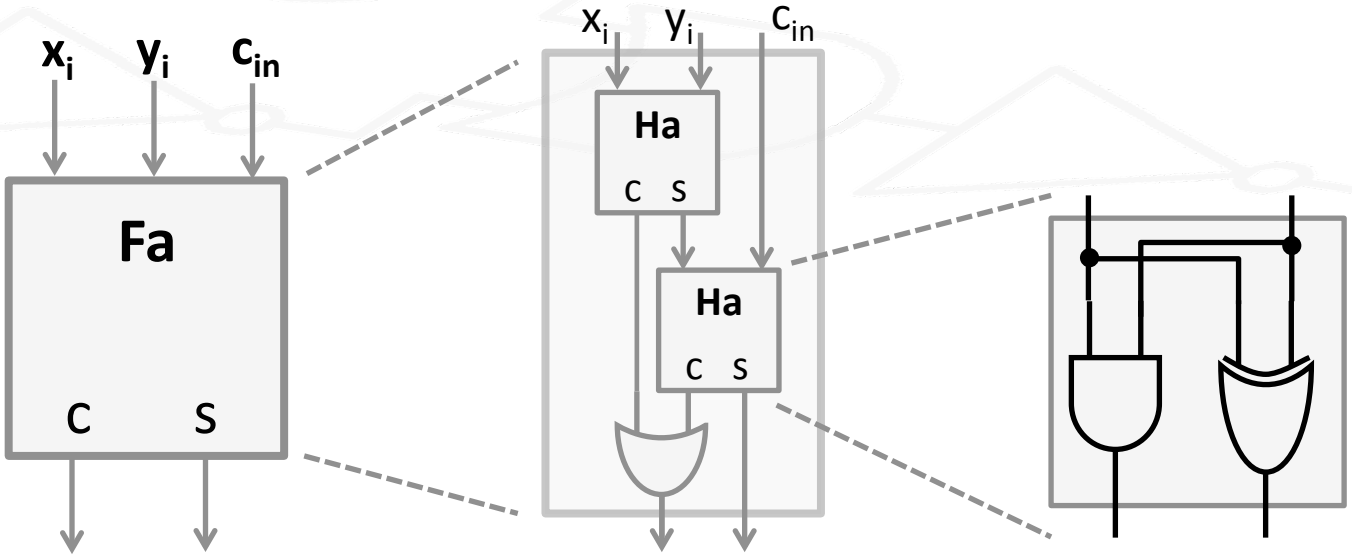
Introducción

- **Implementando CLCs grandes conectando CLCs más pequeños (un Sumador):**
  - Un sumador con propagación de acarreo que sume números de n bits se puede construir replicando n CLCs que realicen la suma de cada columna de bits (estos CLCs se denominan Full Adders ó Fa).



Introducción

- **Implementando CLCs grandes conectando CLCs más pequeños (un Sumador):**
  - A su vez, cada Full Adder se construye interconectando CLCs más simples llamados Half Adders.
  - Half Adder: CLC que codifica en binario en número de «unos» en sus entradas. Se construyen con 2 puertas lógicas (AND y XOR).

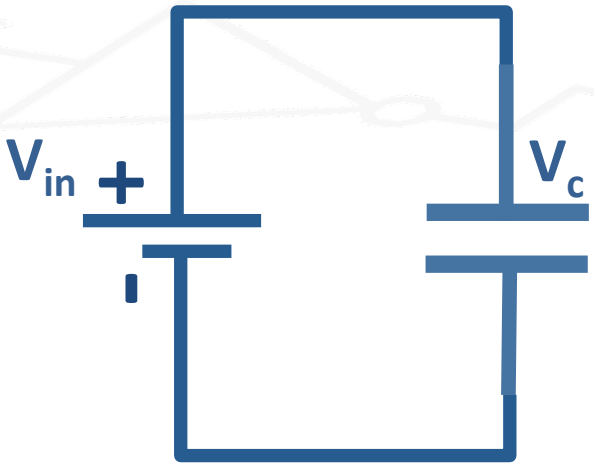


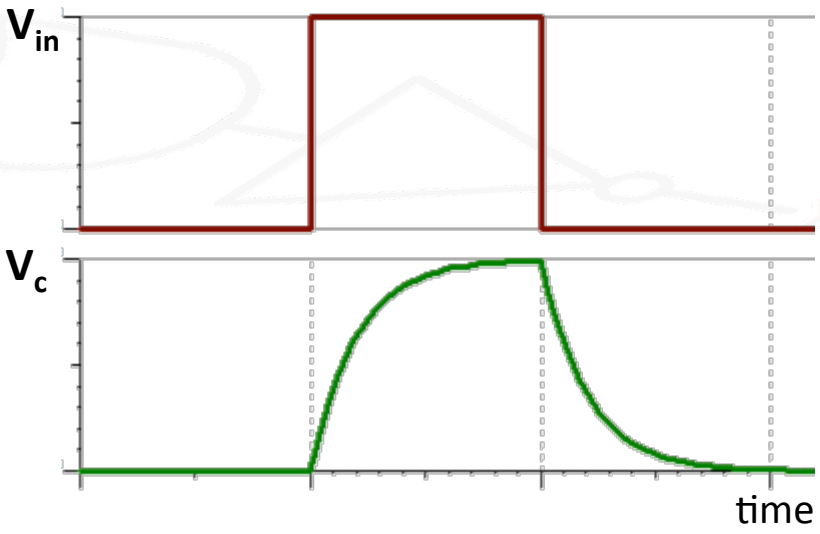
Índice

- **Introducción:**
  - Definición de CLC, Modelo matemático.
  - Del transistor a la puerta lógica.
  - Puertas Lógicas.
  - CLCs jerárquicos.
- **Análisis Temporal:**
  - Tiempo de propagación.
  - Glitches.
- **Álgebra de conmutación:**
  - Axiomas y teoremas.
  - Expresiones/Circuitos equivalentes.
- **Análisis y Síntesis:**
  - Suma de Minterms.
  - Decodificador + OR.
  - ROM.

Análisis Temporal

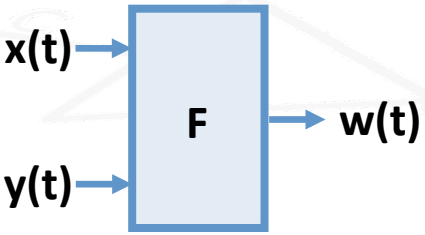
- **Completando el modelo de un CLC:**
  - En los modelos empleados hasta ahora no hemos hablado de tiempo, hemos considerado que la respuesta de un CLC es inmediata.
  - Los transistores (base de toda puerta lógica y por tanto de todo CLC) funcionan como interruptores, pero tardan un tiempo en conmutar.



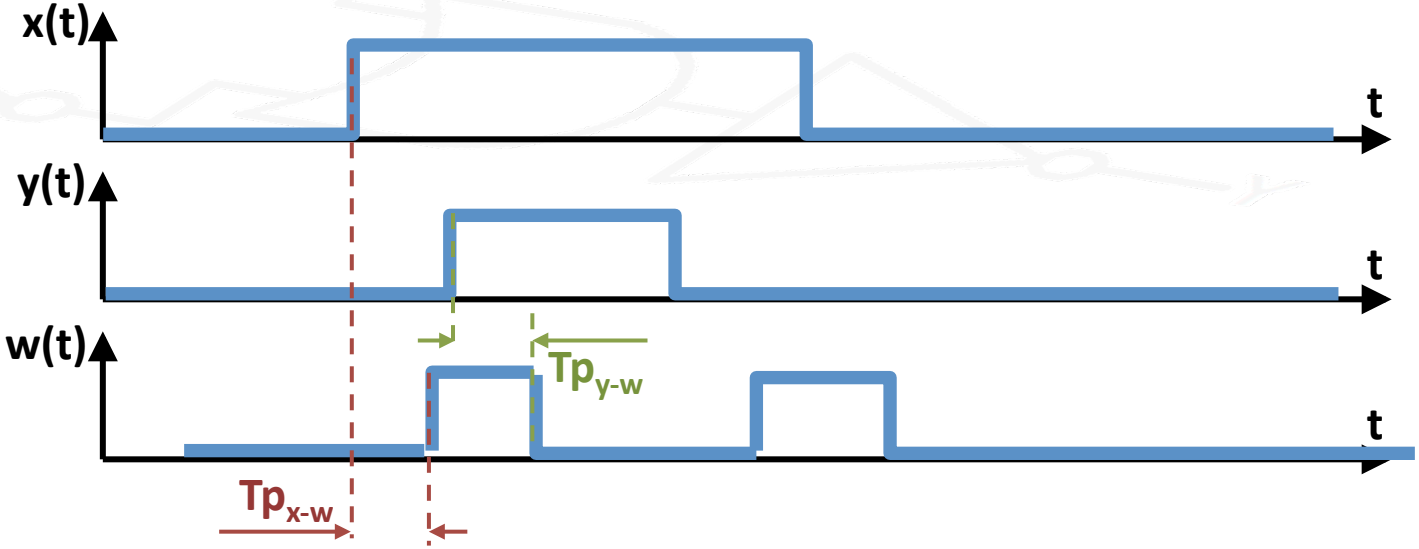


Análisis Temporal

- **Tiempo de Propagación:**
  - En un CLC real la respuesta no es inmediata.
  - Tiempo de propagación de la entrada e a la salida s (**Tp_{e-s}**): tiempo desde que se produce un cambio en la entrada e hasta que la salida s se estabiliza al valor que indica la tabla de verdad para la nueva combinación de los valores de entrada.



x	y	w
0	0	0
0	1	1
1	0	0
1	1	0



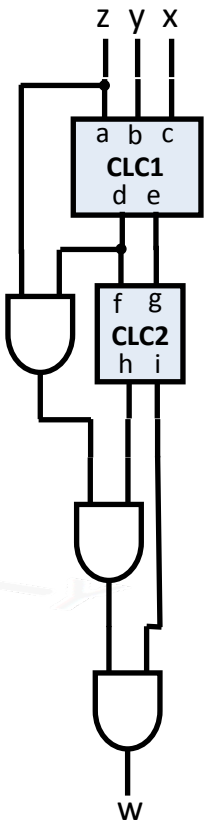
Análisis Temporal

- **Puertas Lógicas, Modelo Completo:**

Nombre	Símbolo	Exp Lógica	T. V.	
Not-1		w = !x	x | w 0 | 1 1 | 0	Tp_{NOT}= 10 u.t.
And-2		w = x · y	x y | w 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	Tp_{AND}=20 u.t.
Or-2		w = x + y	x y | w 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	Tp_{OR}=20 u.t.

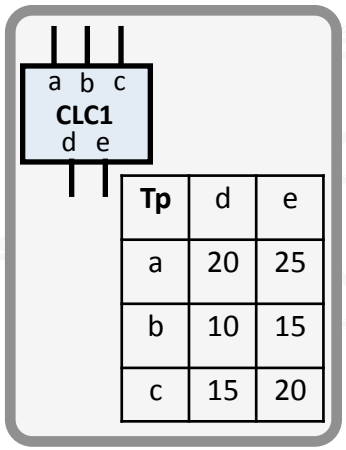
Análisis Temporal

- **Cálculo del Tiempo de Propagación de un CLC:**
  - **Camino** de la entrada *x* a la salida *w:* trayectoria (recorrido) de *x* a *w* pasando por CLCs interconectados. Pueden existir múltiples caminos entre una misma entrada y salida.
  - **Camino crítico** de la entrada *x* a la salida *w:* el camino más largo de *x* a *w*. La longitud se mide como la suma de los Tp de cada entrada a cada salida de los CLCs atravesados por el camino.
  - **Tiempo de propagación** de *x* a *w* (**Tp_{x-w}**): el tiempo del camino crítico de *x* a *w*.
  - **Tiempo de propagación de un CLC:** el mayor de sus caminos críticos (analizando todos los posibles pares entrada-salida).

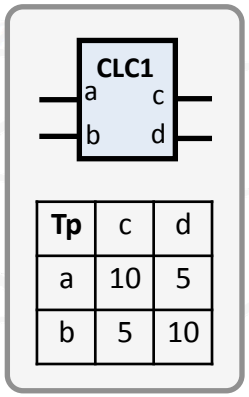


Análisis Temporal

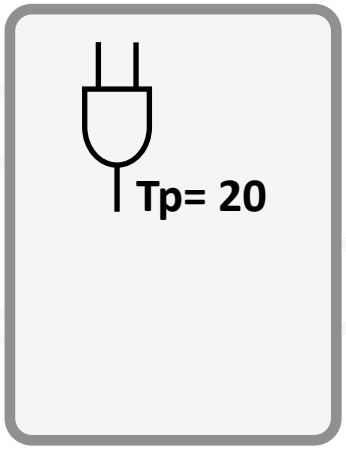
- **Ejemplo:**
  - Dado el CLC de la figura, y el Tp de los caminos internos de los CLCs que lo forman, calcular el camino crítico de dicho circuito:

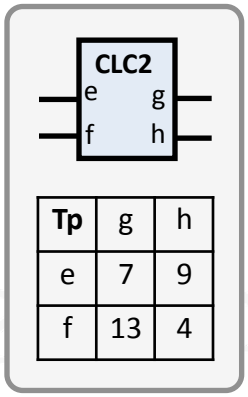


Tp	d	e
a	20	25
b	10	15
c	15	20

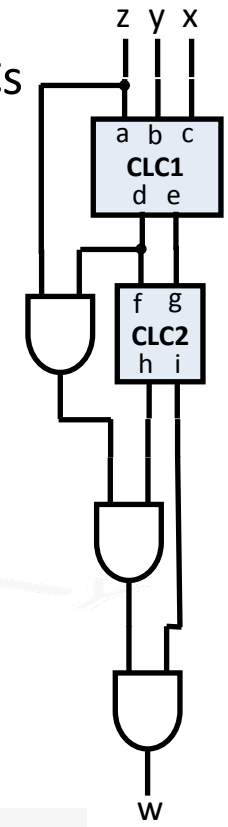


Tp	c	d
a	10	5
b	5	10





Tp	g	h
e	7	9
f	13	4

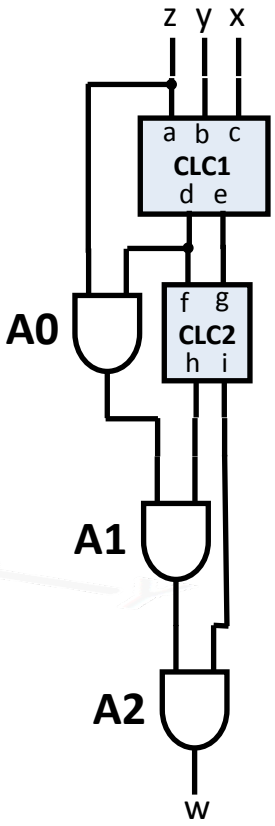


Paso 1: Determinar todos los pares entrada-salida.
Paso 2: Calcular el camino crítico de cada par.
Paso 3: El camino crítico del CLC es el mayor de los calculados en el paso 2.

P1: x-w, y-w, z-w

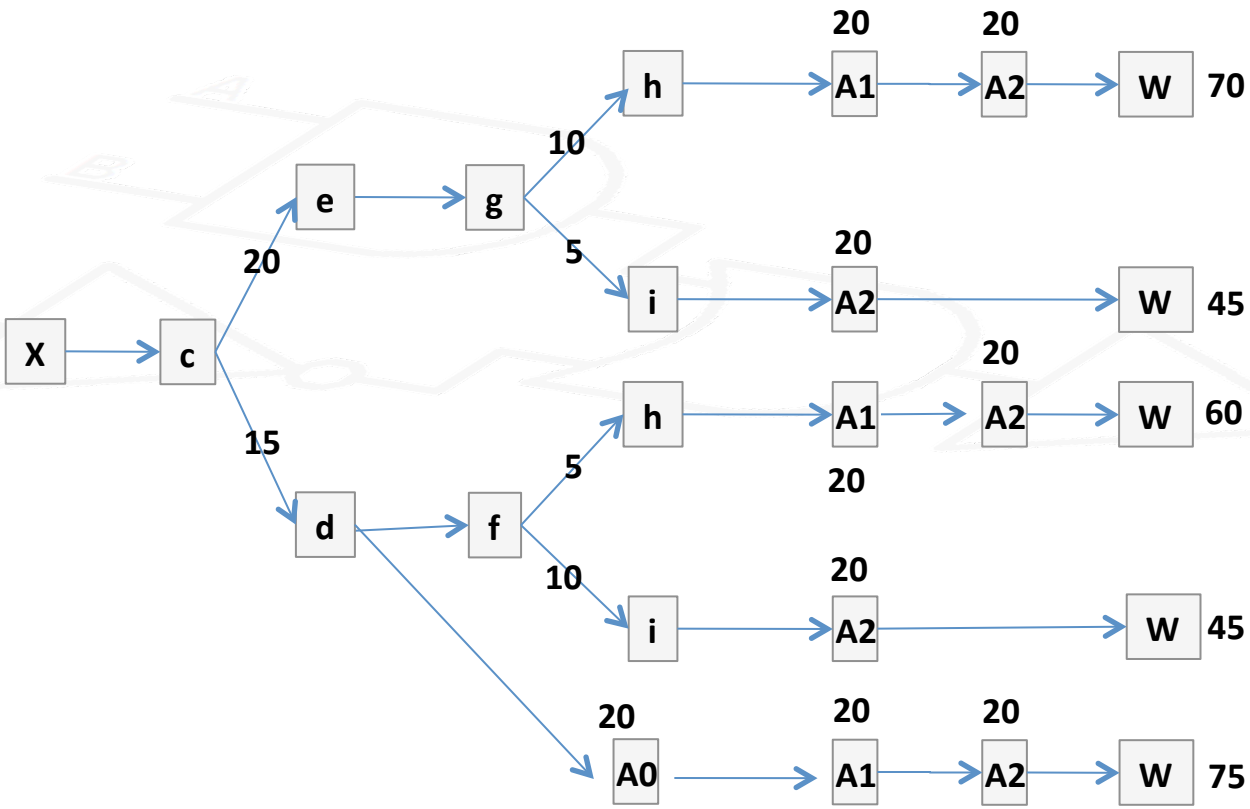
Análisis Temporal

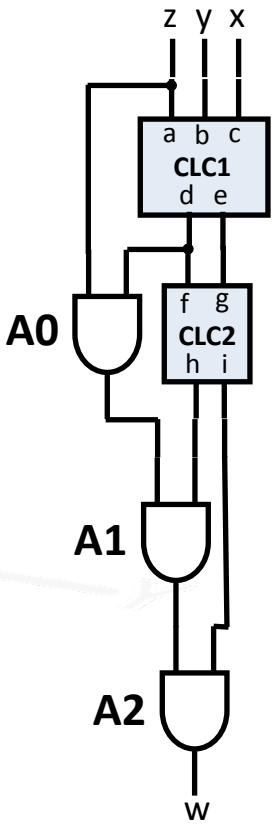
- **Ejemplo:**
  - Paso 2: lo haremos solamente para el par x-w:



Análisis Temporal

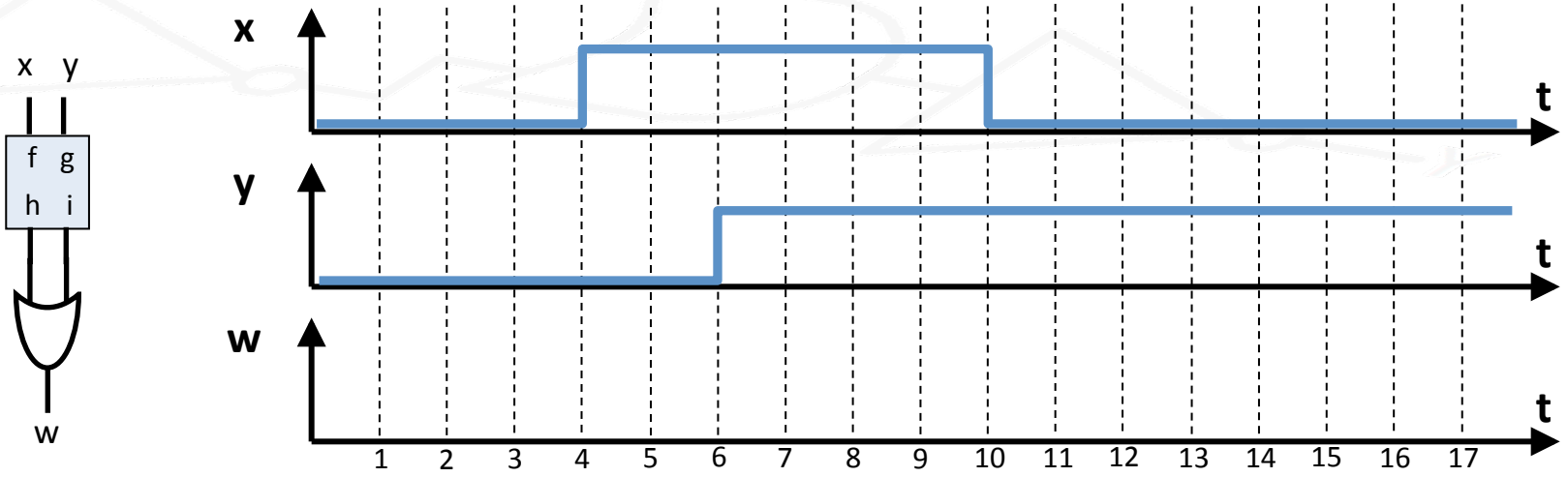
- **Ejemplo:**
  - Paso 2: lo haremos solamente para el par x-w:



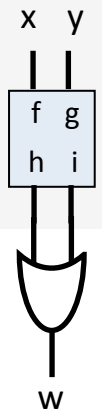


Análisis Temporal

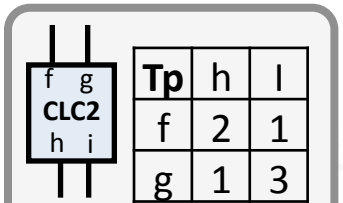
- **Cronogramas:**
  - Las señales de entada de un CLC serán señales digitales síncronas, que variarán a medida que avance el tiempo.
  - Un cronograma nos permite representar la forma que adquirirá la señal de salida de un CLC en función de: las señales de entrada, la tabla de verdad y el tiempo de propagación de cada uno de sus componentes.



Análisis Temporal



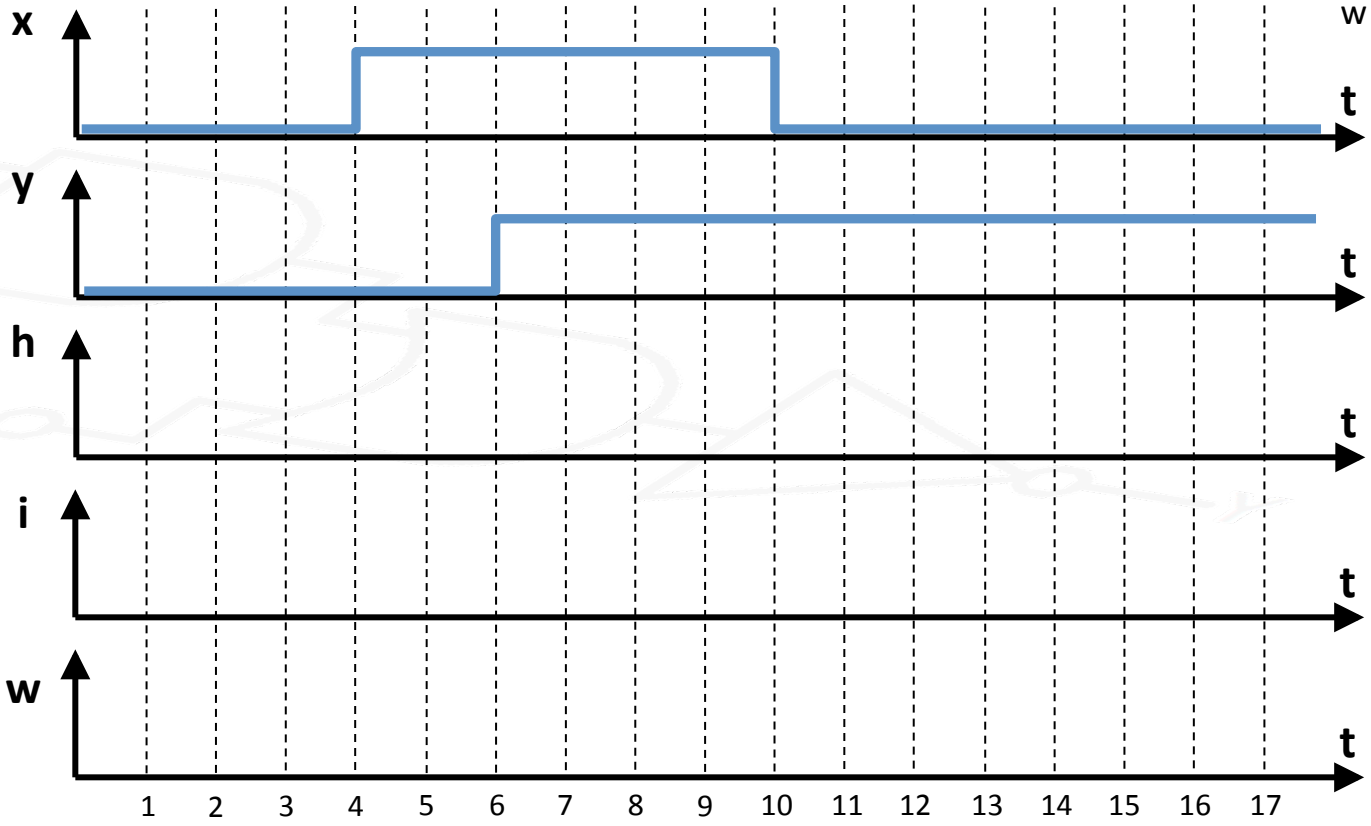
- **Cronogramas:**



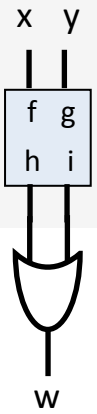
Tp	h	l
f	2	1
g	1	3

f	g	h	l
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

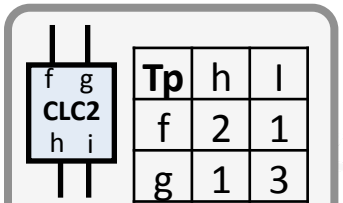
Tp = 2



Análisis Temporal



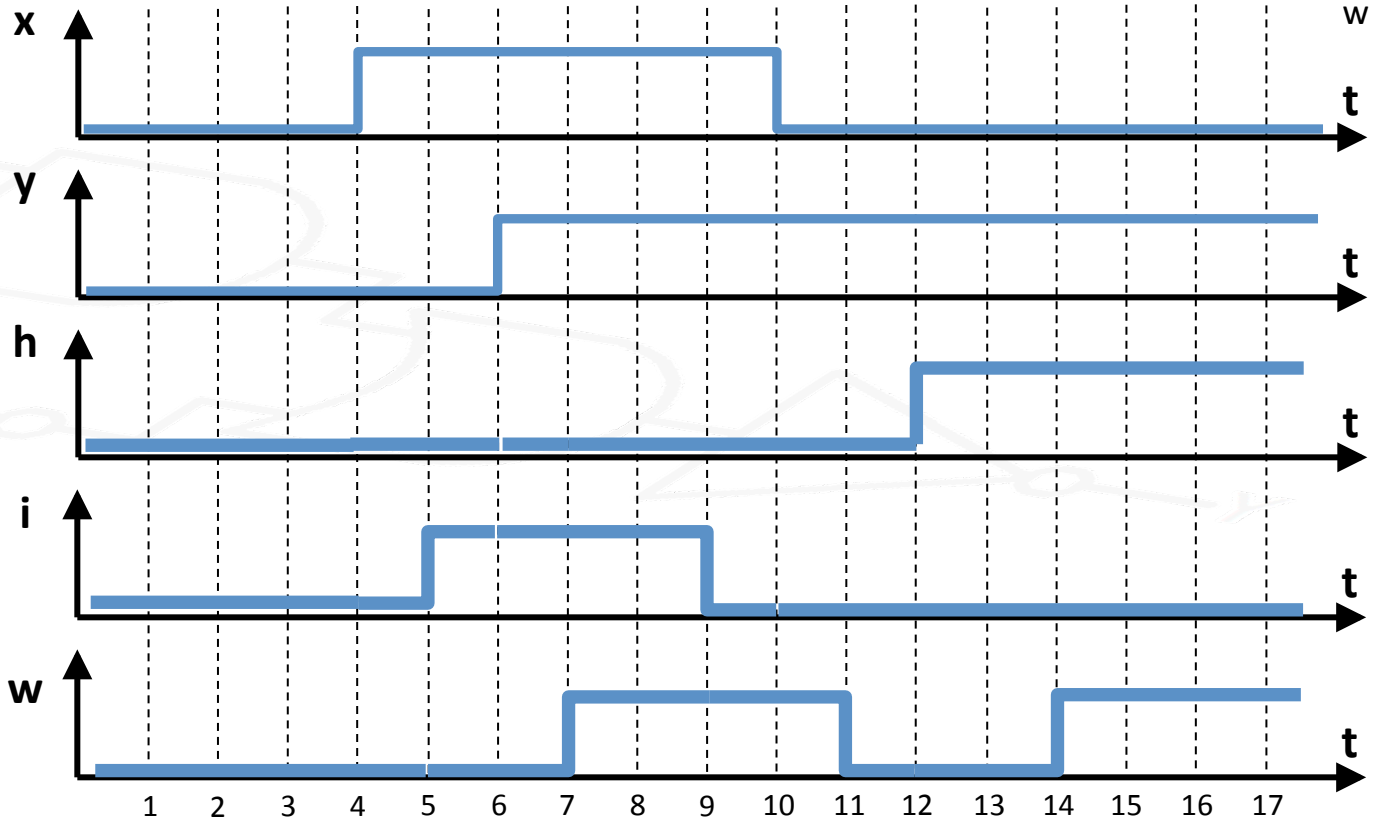
- **Cronogramas:**



Tp	h	l
f	2	1
g	1	3

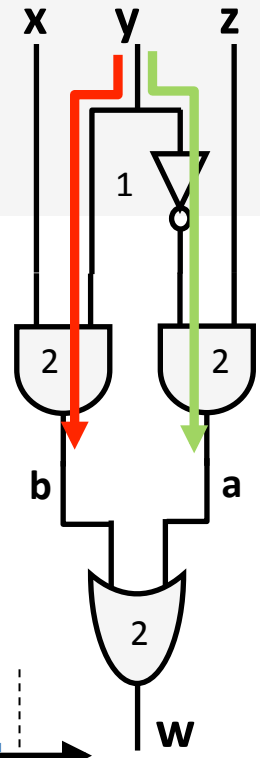
f	g	h	l
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

Tp = 2



Análisis Temporal

- **Cronogramas: una curiosidad, los Glitches:**
  - Si hay dos caminos don distinto retardo de una misma entrada a una misma salida, puede ocurrir que antes de que la salida se estabilice al valor que indica la T.V. se produzcan pulsos indeseados, conocidos como glitches.



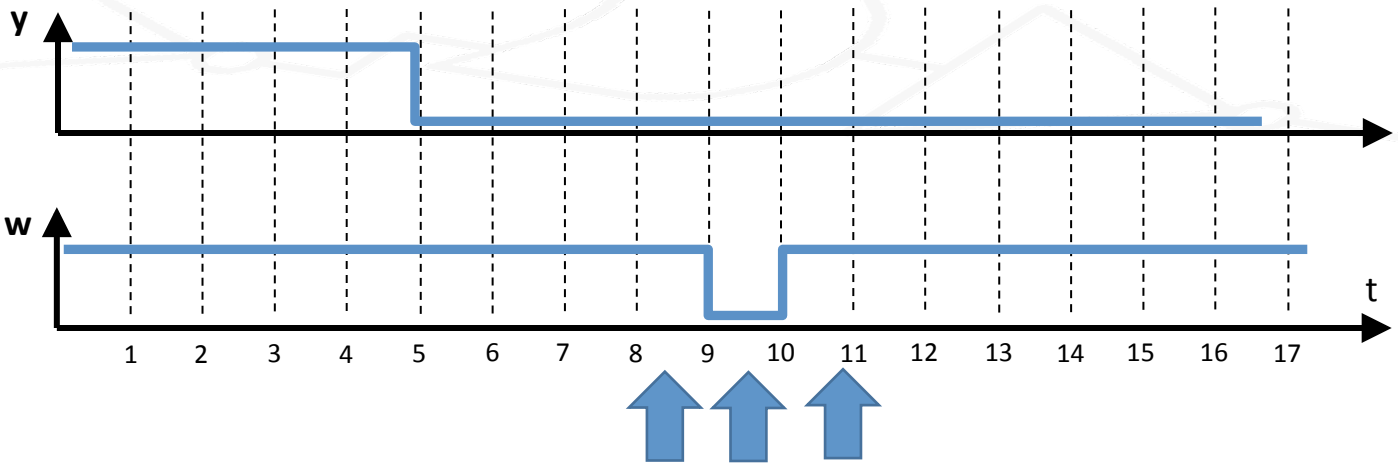
z	y	x	w
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

La entrada «y» cambia 1 → 0

La salida «w» *no debe* cambiar

Análisis Temporal

- **Tiempo de propagación y glitches, ¡un problema a resolver!:**
  - Si no tenemos en cuenta estos dos problemas, podemos considerar que la señal de salida de un circuito es correcta cuando todavía no lo es.
  - Consideraremos que la señal de salida es correcta pasado el tiempo de propagación del CLC (camino crítico).
  - ¿Cómo? Trabajaremos con **circuitos secuenciales síncronos** (próximo tema).



Índice

- **Introducción:**
  - Definición de CLC, Modelo matemático.
  - Del transistor a la puerta lógica.
  - Puertas Lógicas.
  - CLCs jerárquicos.
- **Análisis Temporal:**
  - Tiempo de propagación.
  - Glitches.
- **Álgebra de conmutación:**
  - Axiomas y teoremas.
  - Expresiones/Circuitos equivalentes.
- **Análisis y Síntesis:**
  - Suma de Minterms.
  - Decodificador + OR.
  - ROM.

Álgebra de Conmutación

- Si queremos implementar cierta funcionalidad (qué hace) descrita por una tabla de verdad, llegar a sintetizar (generar) el circuito con puertas Not, And y Or sin herramientas puede ser muy complejo.

- **Álgebra de Boole (Álgebra de Conmutación):** una herramienta para el análisis y síntesis de circuitos:
  - Definida por primera vez en 1854 por George Boole. En 1930, Claude Shannon observó que las reglas de este álgebra eran aplicables al diseño de circuitos de conmutación.

Álgebra de Conmutación

- Un álgebra está definida por:
  - Un conjunto de elementos *K.*
  - Un conjunto de operaciones Φ que actúan sobre los miembros de *K* y que cumplen una serie de propiedades.

- **Álgebra de Boole:**
  - El conjunto ***K*** lo forman solo dos elementos **{0,1}**.
  - Tres **operaciones** (lógicas) {+,*,!} definidas sobre *K*:
    - Una operación unaria (f(x)): Función de Negación ó **NOT** (!).
    - Dos operaciones binarias (f(x,y)): suma (+, **OR**) y producto (*, **AND**).
  - Estas operaciones cumplen las propiedades conocidas como los postulados de *Huntington*.

Álgebra de Conmutación

- Postulados (axiomas) de Huntington:
  - **Conjunto Cerrado:** $x \cdot y \in K \quad , \quad x+y \in K \quad , \quad !x \in K$
  - **Ley Conmutativa:** $x+y = y+x \quad , \quad x \cdot y = y \cdot x$
  - **Ley asociativa:** $(x+y)+z = x+(y+z) \quad , \quad (x \cdot y) \cdot z = x \cdot (y \cdot z)$
  - **Ley distributiva:** $(x+y) \cdot z = x \cdot z + y \cdot z \quad , \quad x+(y \cdot z) = (x+y) \cdot (x+z)$
  - **Identidad:** $x+0=x \quad , \quad x \cdot 1 = x$
  - **Complemento:** $x+(!x)=1 \quad , \quad x \cdot (!x) = 0$

Álgebra de Conmutación

- Teoremas de 1 variable (demostrados a partir de los axiomas):
  - **Elemento Nulo:** $x+1=1$, $x\cdot 0=0$
  - **Idempotencia:** $x+x=x$, $x\cdot x=x$
  - **Involución:** $!(!x) = x$
- Teoremas de 2 variables:
  - **Cobertura:** $x+(x\cdot y) = x$ [demo: $x\cdot 1+x\cdot y = x\cdot(1+y) = x$]
  - **Combinación:** $(x\cdot y)+(x\cdot !y) = x$ [demo: $(x\cdot y)+(x\cdot !y)=x\cdot(y+!y)=x$]
  - **Morgan:** $!(x+y+z) = !x\cdot !y\cdot !z$, $!(x\cdot y\cdot z) = !x+!y+!z$

Álgebra de Conmutación

- **Demostración de Teoremas:**
  - Además de a través de los axiomas, los teoremas se pueden demostrar mediante las tablas de verdad de las operaciones NOT, AND y OR.
  - Ejemplo: Teorema de combinación (x·y)+(x·!y) = x.

NOT

x	w
0	1
1	0

AND

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

x	y	x·y	!y	x·!y	(x·y)+(x·!y)
0	0	0	1	0	0
0	1	0	0	0	0
1	0	0	1	1	1
1	1	1	0	0	1

Álgebra de Conmutación

- **Ejercicio:**
  - Demostrar el siguiente teorema, obteniendo la tabla de verdad de cada lado de la igualdad:

$$(x+y)\cdot(x+z) = x+(y\cdot z)$$

NOT

x	w
0	1
1	0

AND

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Álgebra de Conmutación

- **Ejercicio:**
  - Demostrar el siguiente teorema, obteniendo la tabla de verdad de cada lado de la igualdad:

$$(x+y)\cdot(x+z) = x+(y\cdot z)$$

NOT

x	w
0	1
1	0

AND

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

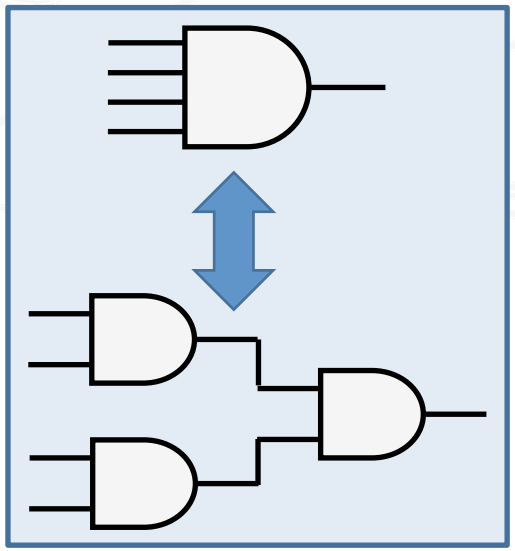
OR

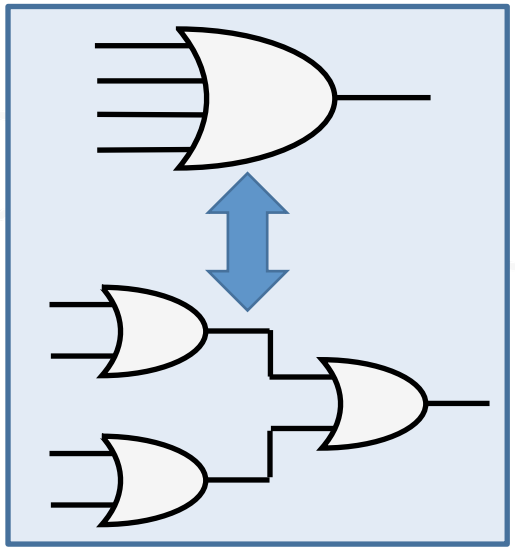
x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

x	y	z	A= x+y	B= x +z	A·B	C=x	D= y·z	C+D
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	0	1
1	0	1	1	1	1	1	0	1
1	1	0	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1

Álgebra de Conmutación

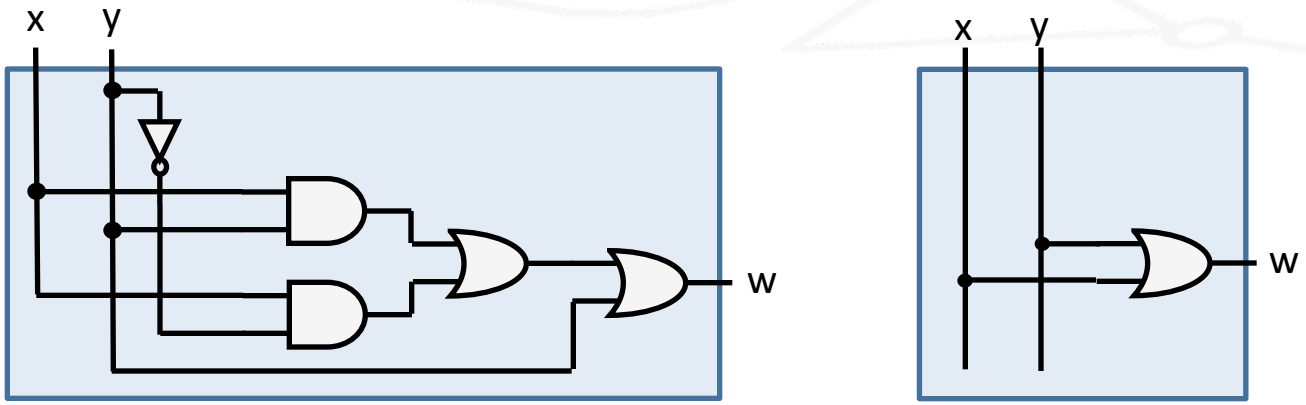
- **Puertas AND y OR de n entradas:**
  - De los teoremas:
    - $x \cdot (y \cdot z) = (x \cdot y) \cdot z = x \cdot y \cdot z = x \cdot z \cdot y = y \cdot z \cdot x = y \cdot z \cdot x = z \cdot x \cdot y = z \cdot y \cdot x$.
    - $x + (y + z) = (x + y) + z = x + y + z = x + z + y = \ldots = z + y + x$.





Álgebra de Conmutación

- **Expresiones/Circuitos equivalentes:**
  - Reglas de precedencia en expresiones lógicas: () → ! → · → +
    - (x·y)+(x·z) = x·y+x·z.
  - **Expresiones lógicas equivalentes**: expresiones que comparten la misma tabla de verdad (las igualdades de los teoremas):
    - Ejemplo: (x·y)+(x·!y)+y es equivalente a x+y.
  - Una expresión lógica especifica un circuito lógico, y viceversa, por tanto…
  - **Circuitos lógicos equivalentes:**



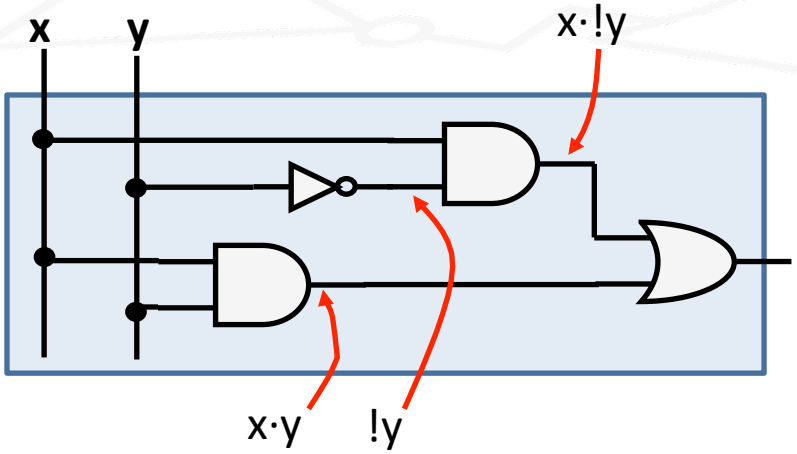
Índice

- **Introducción:**
  - Definición de CLC, Modelo matemático.
  - Del transistor a la puerta lógica.
  - Puertas Lógicas.
  - CLCs jerárquicos.
- **Análisis Temporal:**
  - Tiempo de propagación.
  - Glitches.
- **Álgebra de conmutación:**
  - Axiomas y teoremas.
  - Expresiones/Circuitos equivalentes.
- **Análisis y Síntesis:**
  - Suma de Minterms.
  - Decodificador + OR.
  - ROM.

Análisis y Síntesis

- **Análisis de CLCs:** a partir del esquema de un CLC, obtener su tabla de verdad, con el fin de caracterizar el circuito.
- Se suelen emplear los siguientes pasos:
  - 1. Del CLC a su expresión lógica directa.
  - 2. De la expresión lógica a la tabla de verdad.

Paso 1:



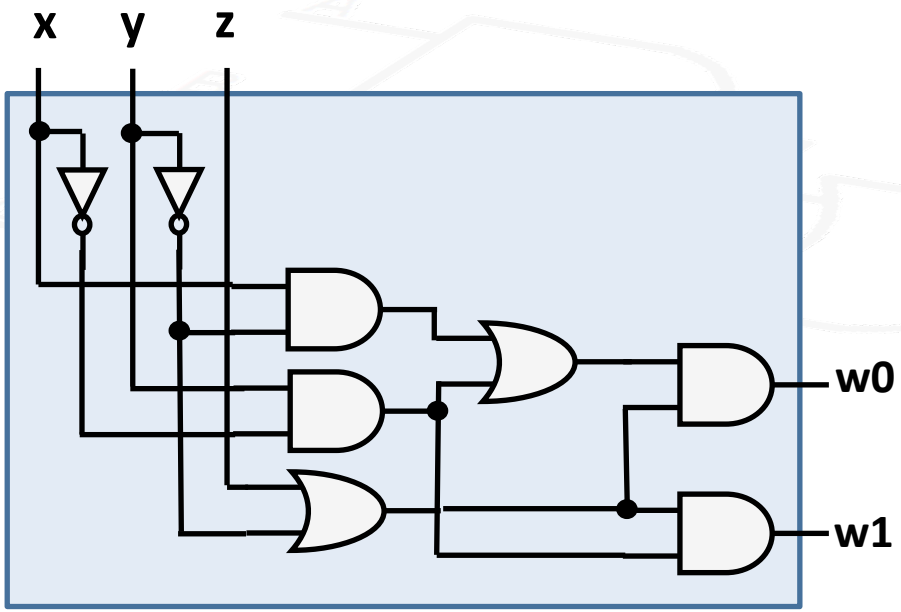
(x·y)+(x·!y)

Paso 2:

x	y	x·y	x·!y	(x·y)+(x·!y)
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

Análisis y Síntesis

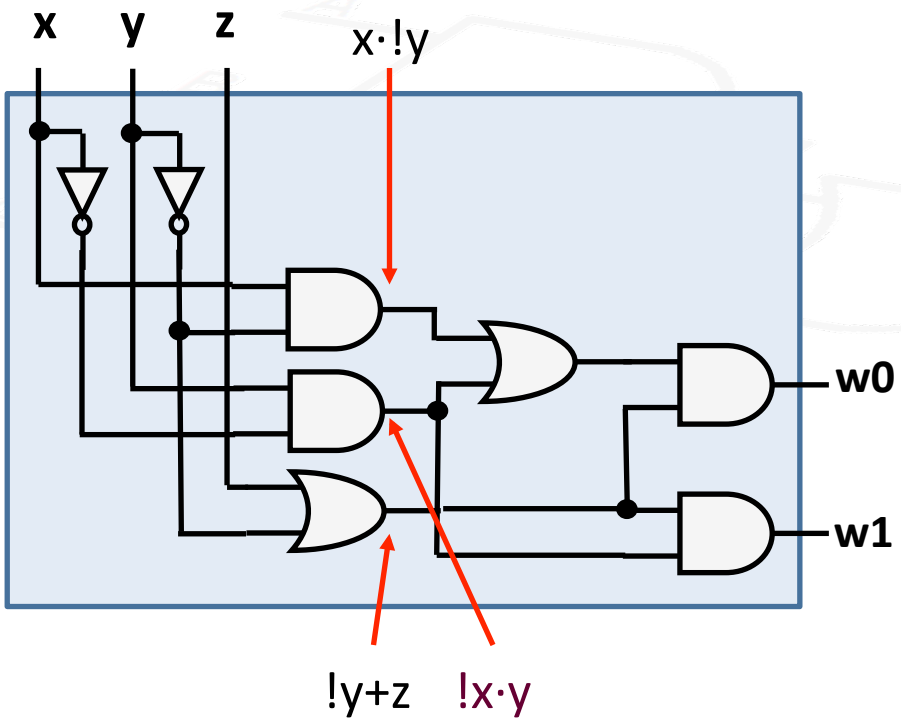
- **¿Qué hacer cuando la complejidad del circuito crece?:**
  - **Posibilidad 1:** hacer la tabla de verdad de cada subexpresión (a la salida de cada puerta).



x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Análisis y Síntesis

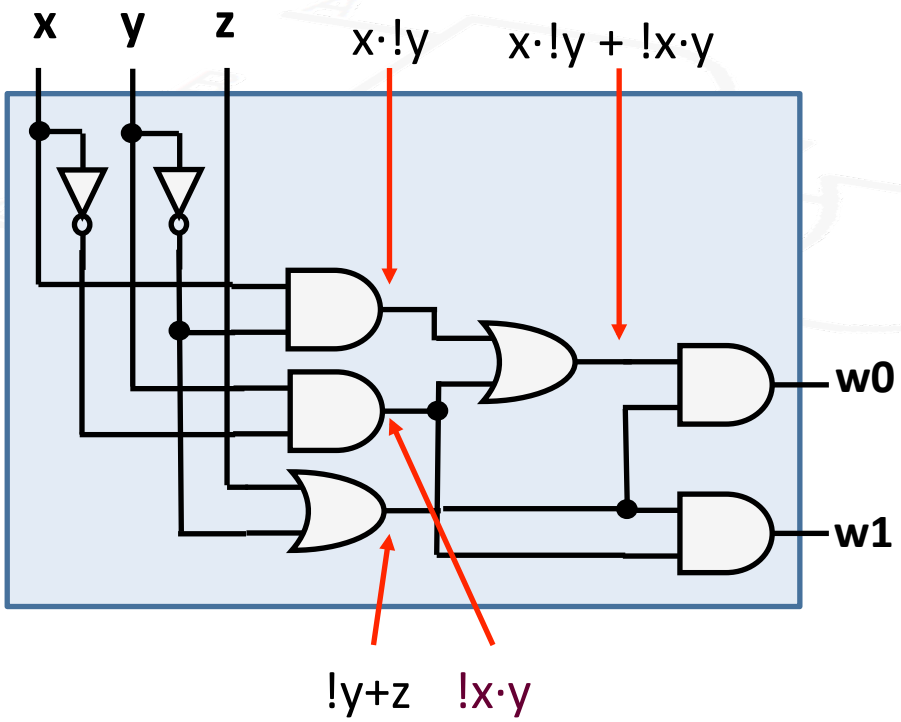
- **¿Qué hacer cuando la complejidad del circuito crece?:**
  - **Posibilidad 1:** hacer la tabla de verdad de cada subexpresión (a la salida de cada puerta).



x	y	z	x·!y	!x·y	!y+z
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	0	0	1

Análisis y Síntesis

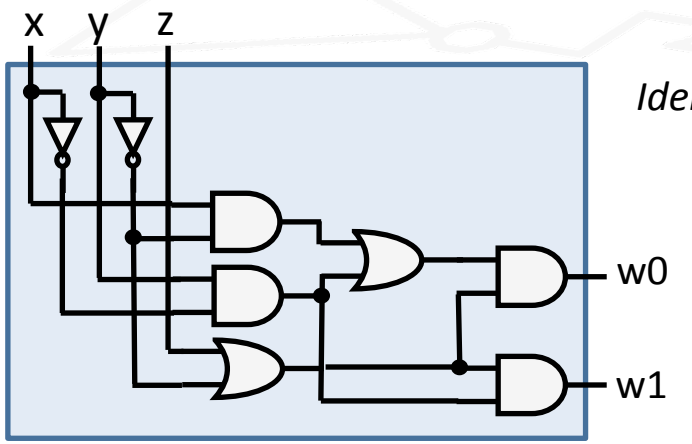
- **¿Qué hacer cuando la complejidad del circuito crece?:**
  - **Posibilidad 1:** hacer la tabla de verdad de cada subexpresión (a la salida de cada puerta).



x	y	z	x·!y	!x·y	!y+z	x·!y+!x·y	w1	w0
0	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0
0	1	1	0	1	1	1	1	1
1	0	0	1	0	1	1	0	1
1	0	1	1	0	1	1	0	1
1	1	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0

Análisis y Síntesis

- **¿Qué hacer cuando la complejidad del circuito crece?:**
  - **Posibilidad 2:** Manipular las expresiones para reducirlas a suma de productos y luego hacer la tabla de verdad.



w0= (x·!y + !x·y)·(!y+z)

Distributiva ||

x·!y·(!y+z)+!x·y·(!y+z)

Distributiva ||

x·!y·!y + x·!y·z + !x·y·!y +!x·y·z

Idempotencia || *Complemento*

x·!y + x·!y·z + 0 +!x·y·z

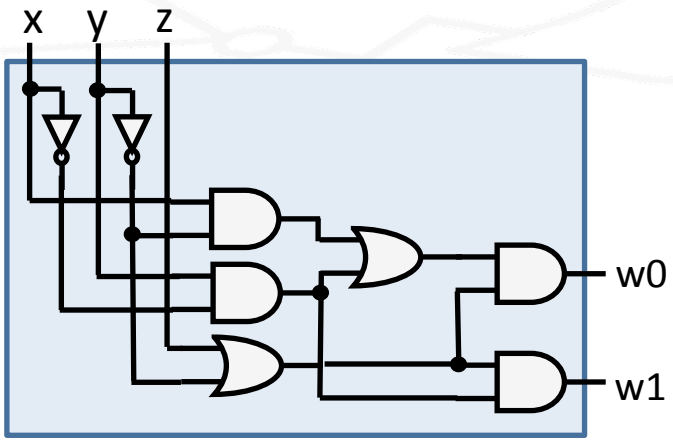
Cobertura ||

x·!y +!x·y·z

x	y	z	w0	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	!x·y·z
1	0	0	1	x·!y
1	0	1	1	
1	1	0	0	
1	1	1	0	

Análisis y Síntesis

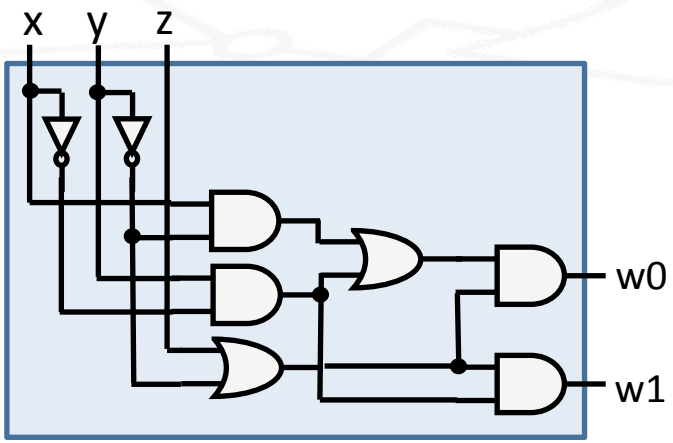
- **Ejercicio:**
  - Repetir el proceso de manipulación de expresiones realizado en la transparencia anterior, esta vez para la salida **w1**.
  - A la vista de las expresiones obtenidas, ¿podrías dibujar un circuito equivalente al de la figura que utilice menos puertas lógicas para su implementación?

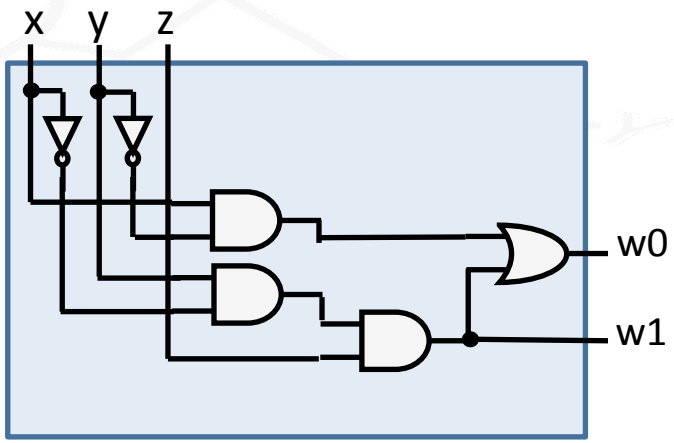


Análisis y Síntesis

- **Ejercicio:**
  - Repetir el proceso de manipulación de expresiones realizado en la transparencia anterior, esta vez para la salida **w1**.
  - A la vista de las expresiones obtenidas, ¿podrías dibujar un circuito equivalente al de la figura que utilice menos puertas lógicas para su implementación?

w1= !x·y·(!y+z) = !x·y·z





Análisis y **Síntesis**

- **Síntesis de CLCs:** a partir de una tabla de verdad, obtener el CLC que implementa dicha funcionalidad.

- Vamos a aprender tres técnicas sencillas:
  - Suma de Minterms.
  - Decodificador + OR.
  - ROM.

- Se trata de técnicas que ofrecen una solución válida, pero no tiene por qué ser la óptima (mínimo número de puertas lógicas).

Análisis y **Síntesis**

- **Síntesis en suma de Minterms:**
  - **Función Minterm:** se trata de una función cuyo valor es 1 para una única combinación de sus variables de entrada. Para el resto de combinaciones el valor de la función es 0.
  - $m_i(x_2,x_1,x_0)$: función minterm i de las variables x_2, x_1, x_0. La salida vale 1 para la fila i de la tabla de verdad.

x	y	z	m_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

¿Qué expresión lógica corresponde con este minterm? *(¿Qué puerta lógica utilizarías para implementarlo?).*

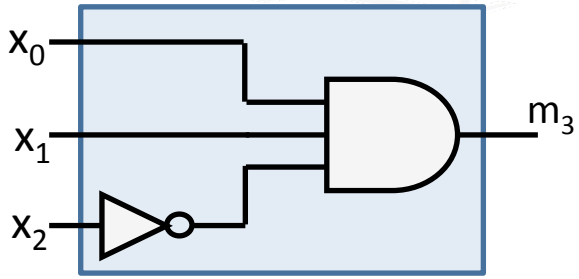
Análisis y **Síntesis**

- **Síntesis en suma de Minterms:**
  - **Función Minterm:** se trata de una función cuyo valor es 1 para una única combinación de sus variables de entrada. Para el resto de combinaciones el valor de la función es 0.
  - $m_i(x_2,x_1,x_0)$: función minterm i de las variables x_2, x_1, x_0. La salida vale 1 para la fila i de la tabla de verdad.

x	y	z	m_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

¿Qué expresión lógica corresponde con este minterm? *(¿Qué puerta lógica utilizarías para implementarlo?).*

$$m_3(x_2,x_1,x_0) = !x_2 \cdot x_1 \cdot x_0$$



Análisis y **Síntesis**

- **Síntesis en suma de Minterms:**
  - Cada fila de la tabla de verdad tiene su propio minterm. La suma de dos funciones minterm proporciona como resultado una tabla de verdad en la que son distintas de cero las filas correspondientes a ambos minterms.
  - Por tanto, **cualquier combinación de 1s y 0s como salida en una tabla de verdad puede ser implementada mediante suma de minterms**.

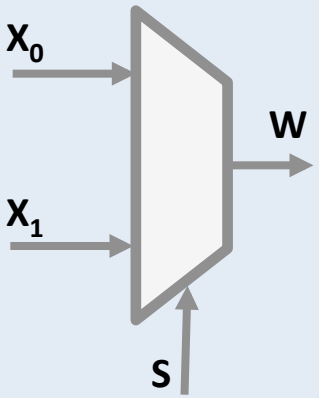
x	y	z	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

x	y	z	m_2	m_5	m_2+m_5
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	**1**	0	**1**
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	**1**	**1**
1	1	0	0	0	0
1	1	1	0	0	0

Análisis y **Síntesis**

- **Ejemplo: Síntesis en suma de Minterms de un Multiplexor.**

Esquema:



Funcionalidad:
if S=0 then $W=X_0$,
else $W=X_1$

Paso 1:
Tabla de Verdad

s	x_1	x_0	w
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Paso 2:
Minterms

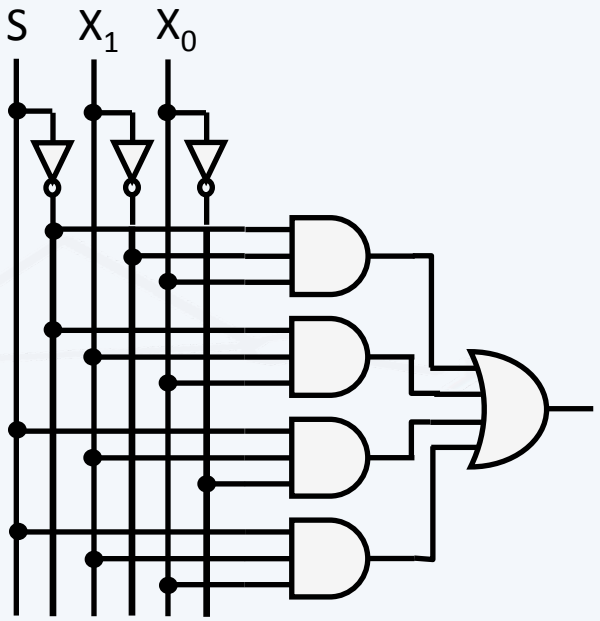
$m_1 = !S \cdot !X_1 \cdot X_0$

$m_3 = !S \cdot X_1 \cdot X_0$

$m_6 = S \cdot X_1 \cdot !X_0$

$m_7 = S \cdot X_1 \cdot X_0$

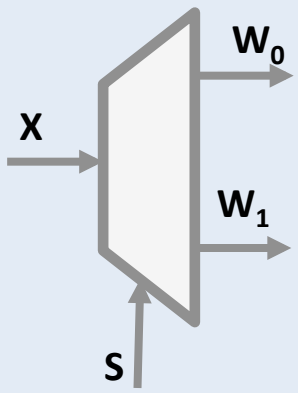
Paso 3:
Síntesis



Análisis y **Síntesis**

- **Ejercicio 1:** sintetizar en suma de minterms un **Demultiplexor**.

Esquema:



Funcionalidad:
if S=0 then W_0=X,
else W_1=X

Paso 1:
Tabla de Verdad

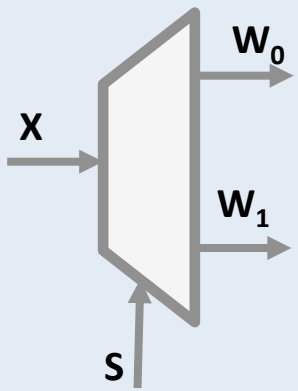
Paso 2:
Minterms

Paso 3:
Síntesis

Análisis y **Síntesis**

- **Ejercicio 1:** sintetizar en suma de minterms un **Demultiplexor**.

Esquema:



Funcionalidad:
if S=0 then W_0=X,
else W_1=X

Paso 1:
Tabla de Verdad

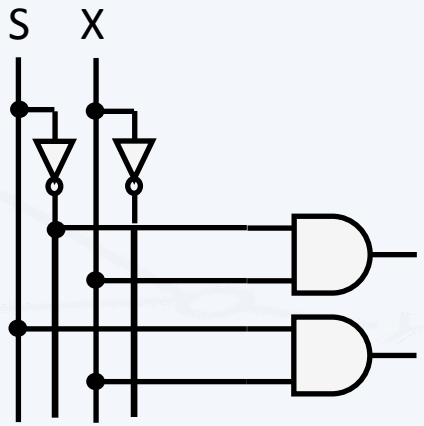
X	S	W_0	W_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Paso 2:
Minterms

$m_2(W_0)=!S\cdot X$

$m_3(W_1)=S\cdot X$

Paso 3:
Síntesis



Análisis y **Síntesis**

- **Ejercicio 2:** sintetizar en suma de minterms la siguiente T.V.

Paso 1:
Tabla de Verdad

x	y	z	w_1	w_0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0

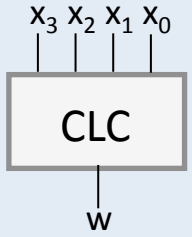
Paso 2:
Minterms

Paso 3:
Síntesis

Análisis y **Síntesis**

- **Ejercicio 3:** dibujar el esquema lógico (suma de minterms) del siguiente CLC.

Esquema:



Funcionalidad:
Este CLC detecta, activando la salida w, cuando la entrada de 4 bits se encuentra codificando en binario alguno de los números naturales 3, 4 ó 13.

Paso 1:
Tabla de Verdad

Paso 2:
Minterms

Paso 3:
Síntesis

Análisis y **Síntesis**

- **Ejercicio 4:** implementa, en suma de minterms, un circuito conversor de números en Signo-Magnitud a Complemento a 2 representados con 3 bits.

- **Ejercicio 5:** la puerta lógica NAND es un tipo de puerta universal, lo que quiere decir que únicamente con este tipo de puerta se puede implementar cualquier función booleana. Demuestra dicha afirmación implementando las funciones NOT, AND y OR con este tipo de puerta. (utiliza los teoremas del álgebra de Boole como ayuda).

Análisis y **Síntesis**

- **Síntesis con <u>Decodificador y OR</u>:**
  - Un decodificador de n entradas y 2^n salidas (Dec-n-2^n) implementa cada uno de los 2^n minterms de n entradas.

Esquema:

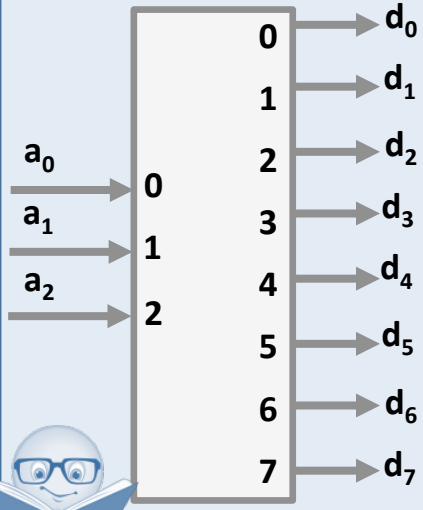
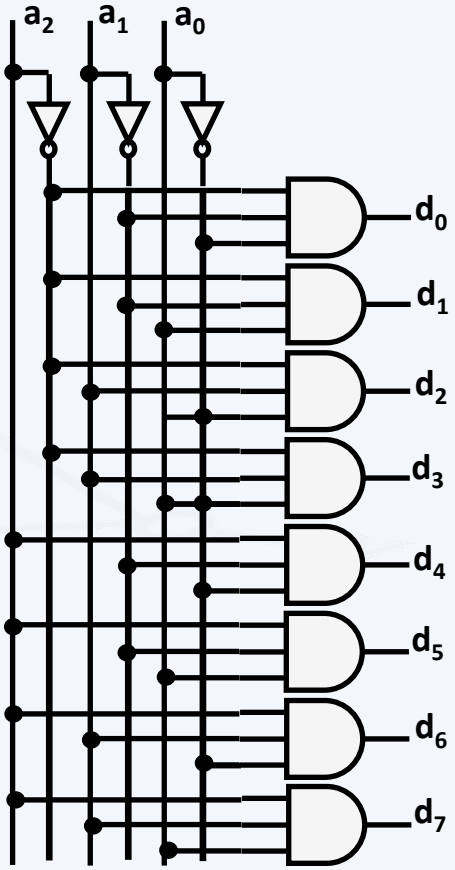


Tabla de Verdad

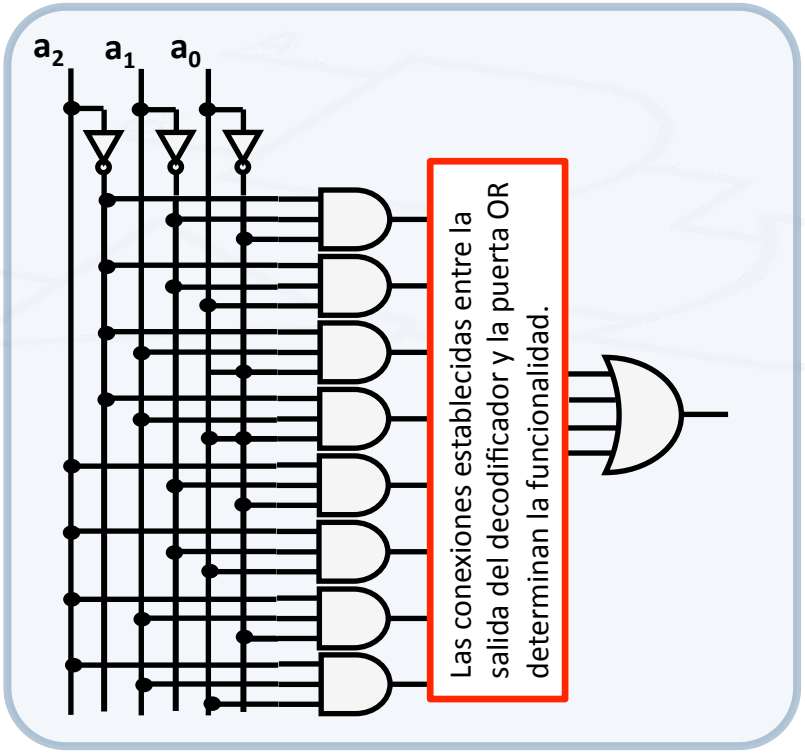
a_2	a_1	a_0	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

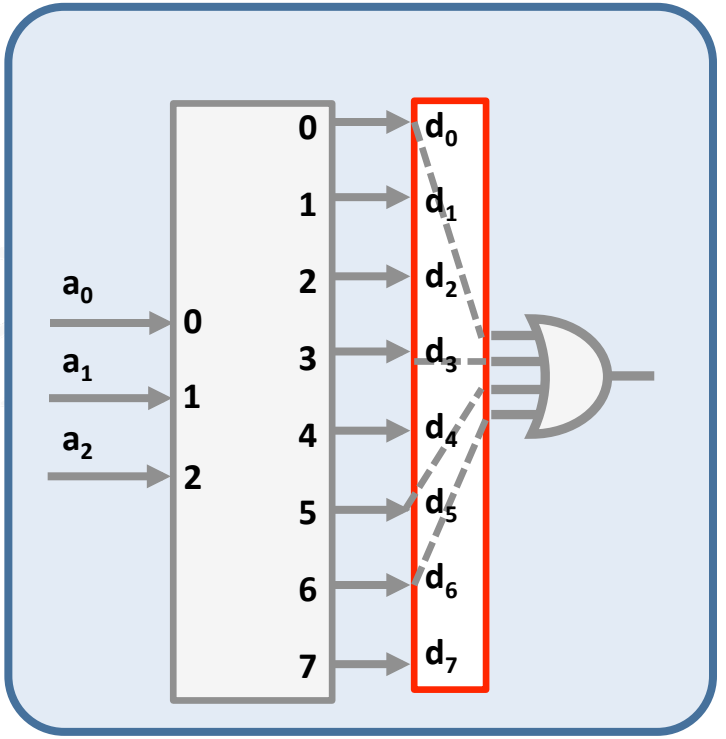
Implementación



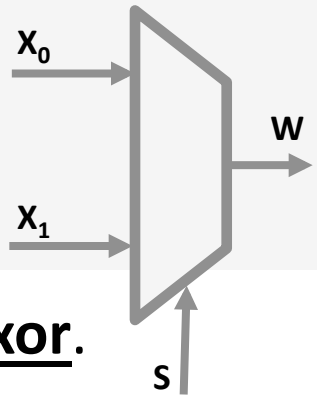
Análisis y **Síntesis**

- **Síntesis con <u>Decodificador y OR</u>:**
  - Utilizando una OR para conectar las salidas del decodificador con la salida del circuito, podremos implementar cualquier CLC en suma de minterms.





Análisis y **Síntesis**



- **Ejemplo: Síntesis en suma de Minterms de un Multiplexor.**

Tabla de Verdad

s	x_1	x_0	w
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Minterms

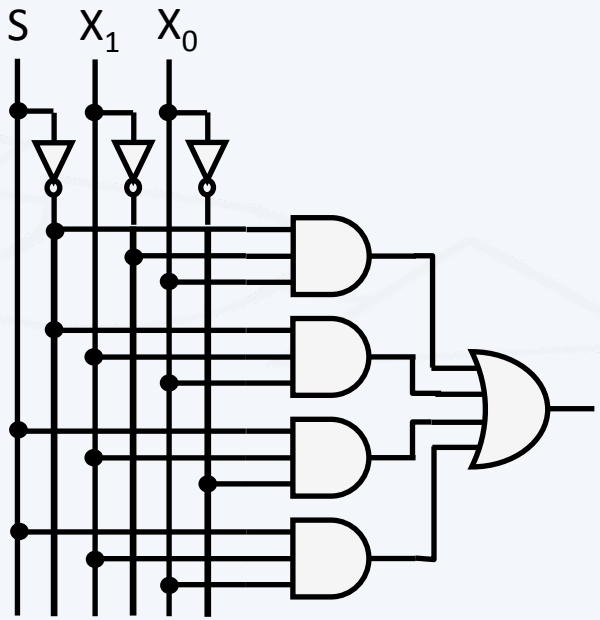
$m_1 = !S \cdot !X_1 \cdot X_0$

$m_3 = !S \cdot X_1 \cdot X_0$

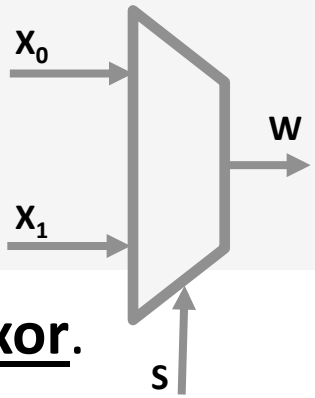
$m_6 = S \cdot X_1 \cdot !X_0$

$m_7 = S \cdot X_1 \cdot X_0$

Síntesis (suma de minterms)



Análisis y **Síntesis**



- **Ejemplo: Síntesis en suma de Minterms de un <u>Multiplexor</u>.**

Tabla de Verdad

S	X_1	X_0	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Minterms

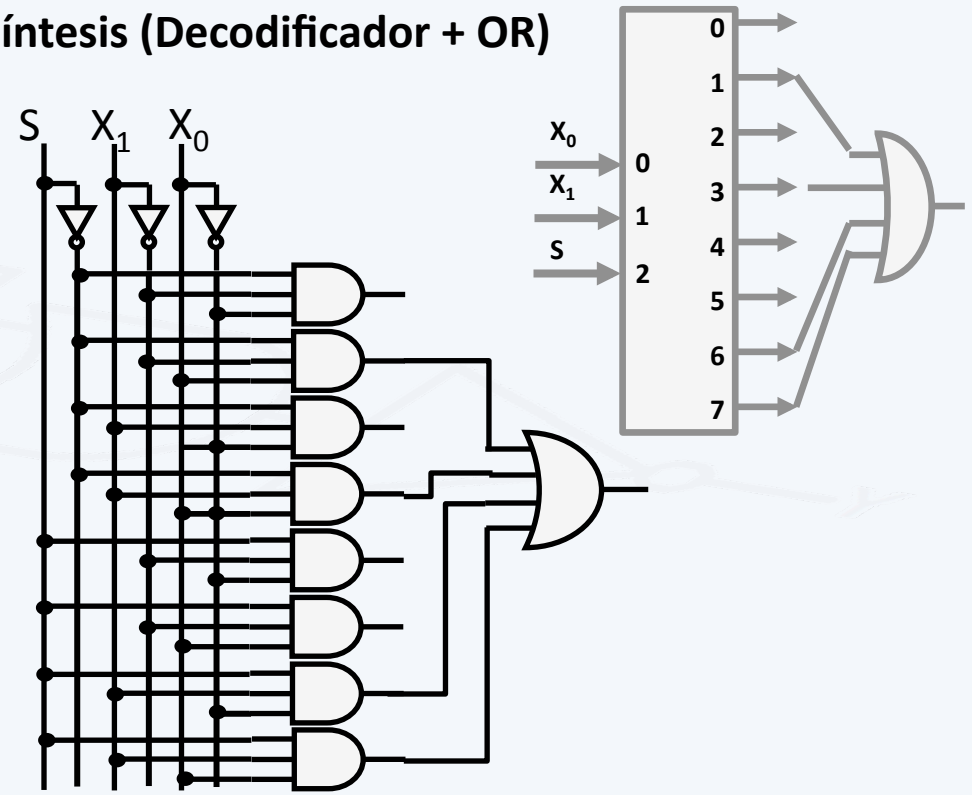
$m_1 = !S \cdot !X_1 \cdot X_0$

$m_3 = !S \cdot X_1 \cdot X_0$

$m_6 = S \cdot X_1 \cdot !X_0$

$m_7 = S \cdot X_1 \cdot X_0$

Síntesis (Decodificador + OR)

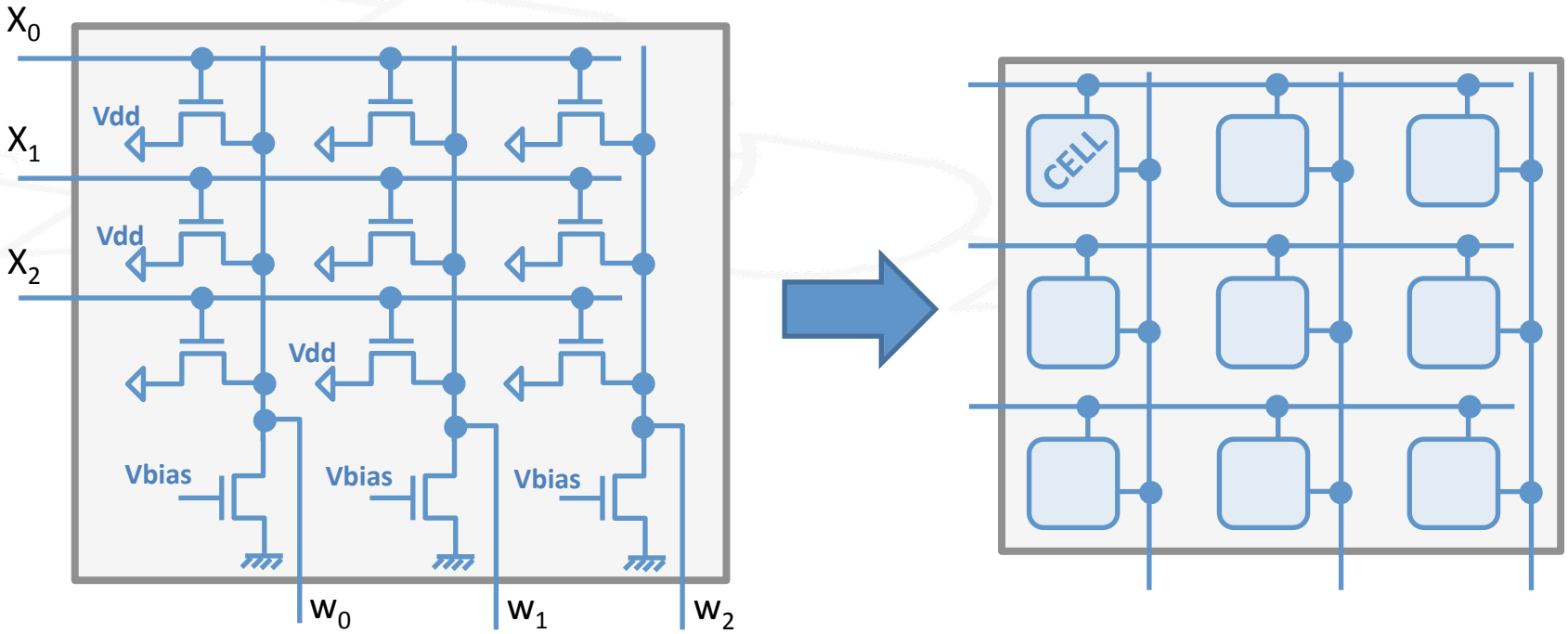


Análisis y **Síntesis**

- **Síntesis con <u>ROM</u>:**
  - ROM (Read Only Memory): Memoria de solo lectura. Circuito en el que se programa (almacena) una configuración determinada de manera externa, para generar una funcionalidad específica.
  - Principal característica (Frente a RAM): Almacenamiento «No-Volatil».
  - Tipos de ROM:
    - **MROM** (mask-programmed), **PROM** (programmable): solamente pueden ser programadas una vez, ni re-programadas ni borradas.
    - **EPROM** (Erasable-programmable): puede ser borrada y reprogramada, pero de manera «externa» (luz ultravioleta).
    - **EEPROM** (electrically Erasable Programmable): puede ser borrada y reprogramada sin dejar de formar parte del sistema digital en el que se emplea (Flash).

Análisis y **Síntesis**

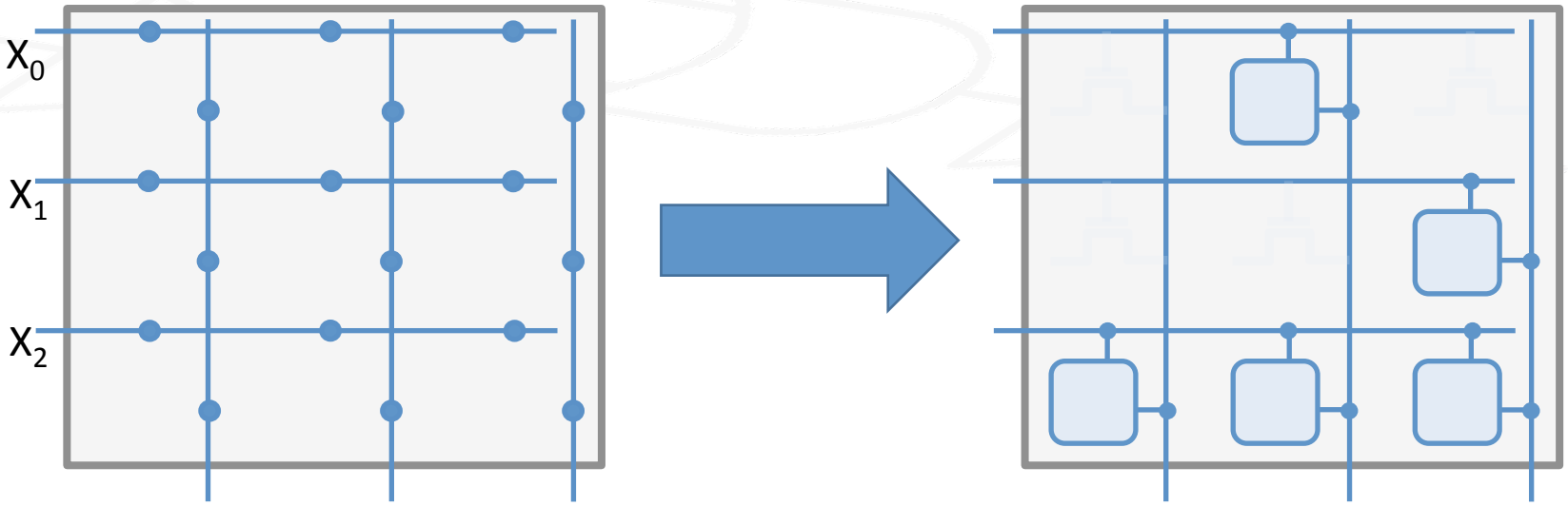
- **Matriz de Conexiones:**
  - Elemento clave de la ROM. Toda entrada y salida se pueden conectar a través de dicha matriz. Los cables de entrada y salida se unen a través de un elemento denominado «celda», que consiste en un transistor CMOS.



Análisis y **Síntesis**

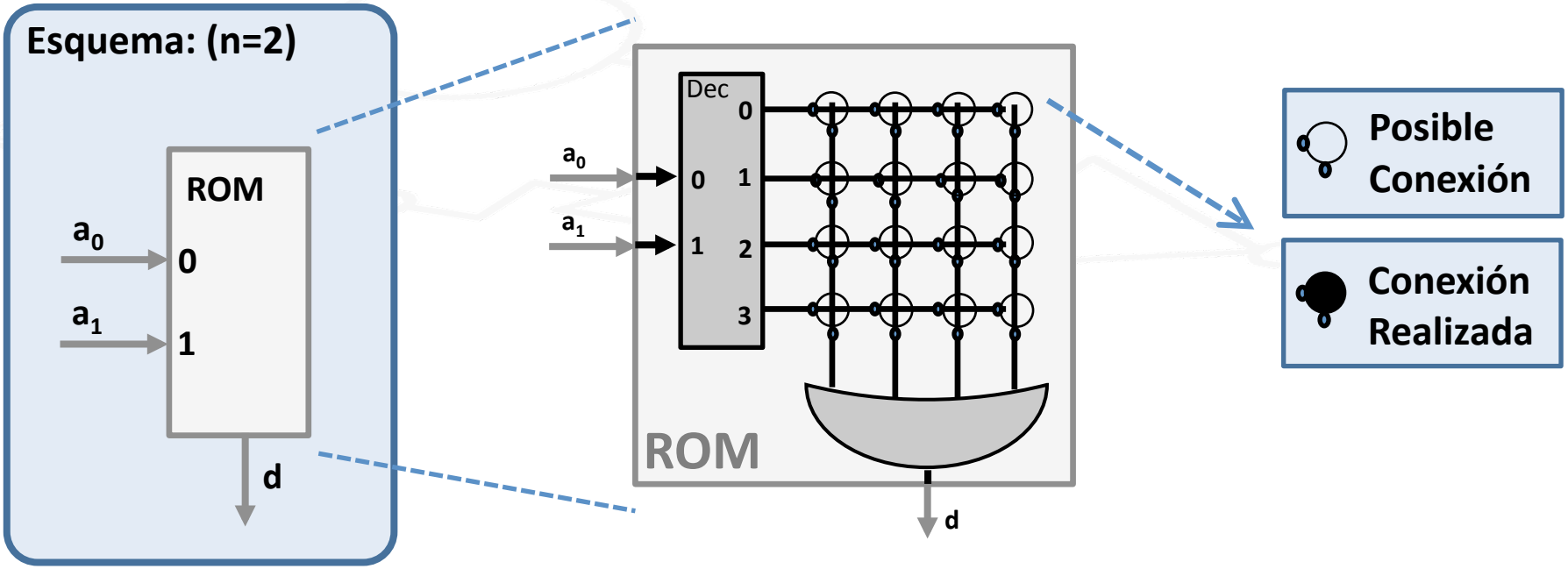
- **Matriz de Conexiones:**
  - Es el elemento programable de la ROM. Se puede configurar la matriz para que posea celdas solo en aquellas intersecciones que nos interesen. El resto de intersecciones se pueden dejar «abiertas» (rompemos la comunicación entre una entrada y una salida determinadas).

Matriz sin programar **Matriz Programada**

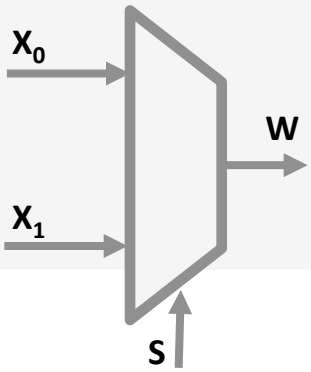


Análisis y **Síntesis**

- **ROM:** CLC compuesto por un decodificador de n entradas, una matriz de tamaño [2^nx2^n] y una puerta OR de 2^n entradas:
  - Con esta configuración podemos implementar cualquier función lógica de n entradas en suma de minterms.



Análisis y **Síntesis**



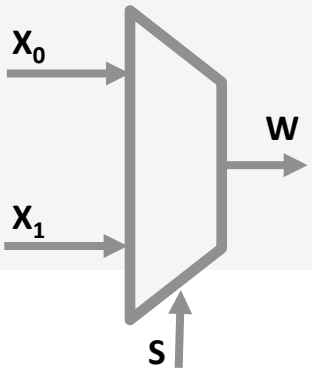
- **Ejemplo:** implementación de un Multiplexor con ROM.

Tabla de Verdad

S	X_1	X_0	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Esquema: (n=??)

Análisis y **Síntesis**

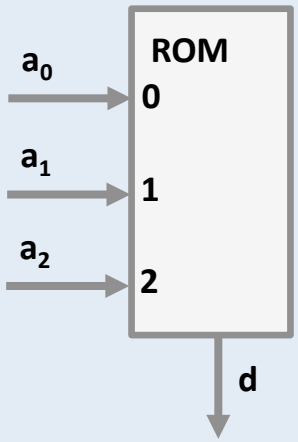


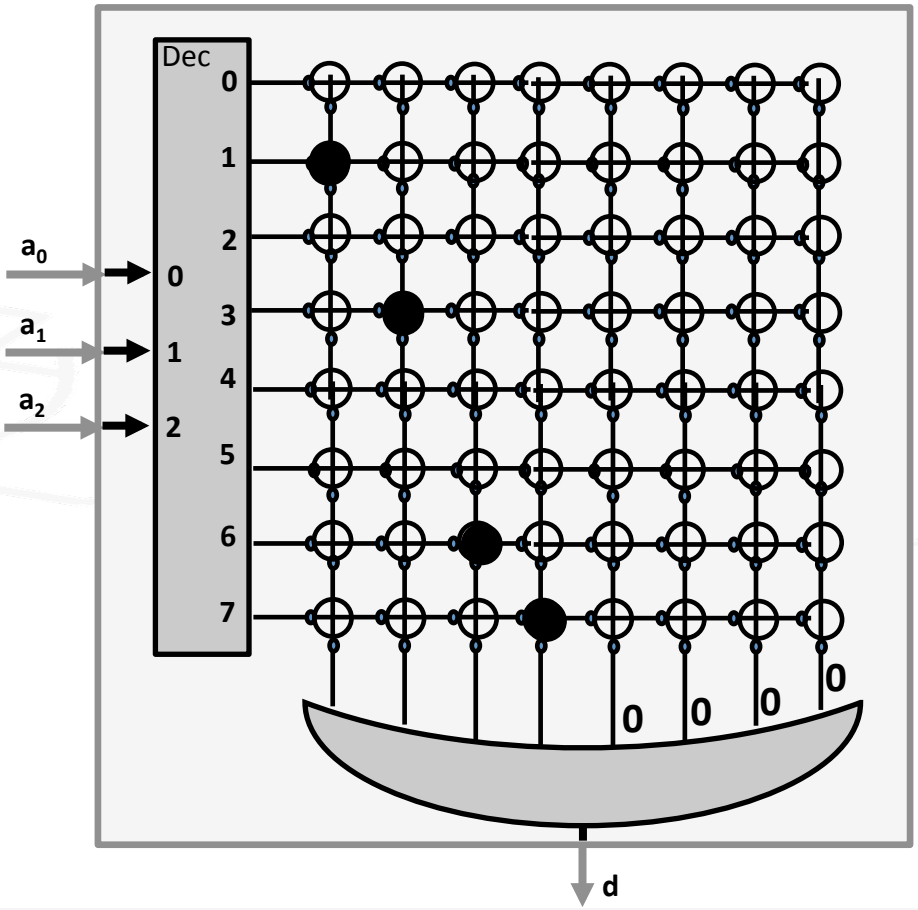
- **Ejemplo:** implementación de un Multiplexor con ROM.

Tabla de Verdad

S	X_1	X_0	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Esquema: (n=??)

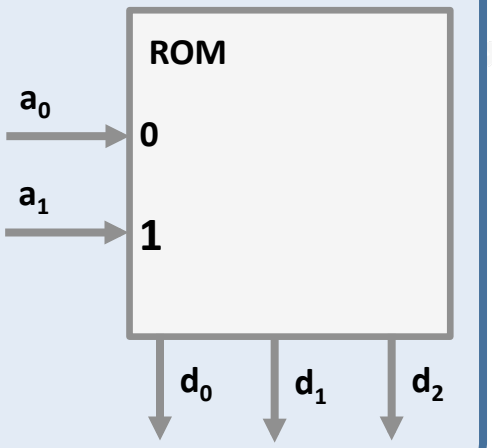




Análisis y **Síntesis**

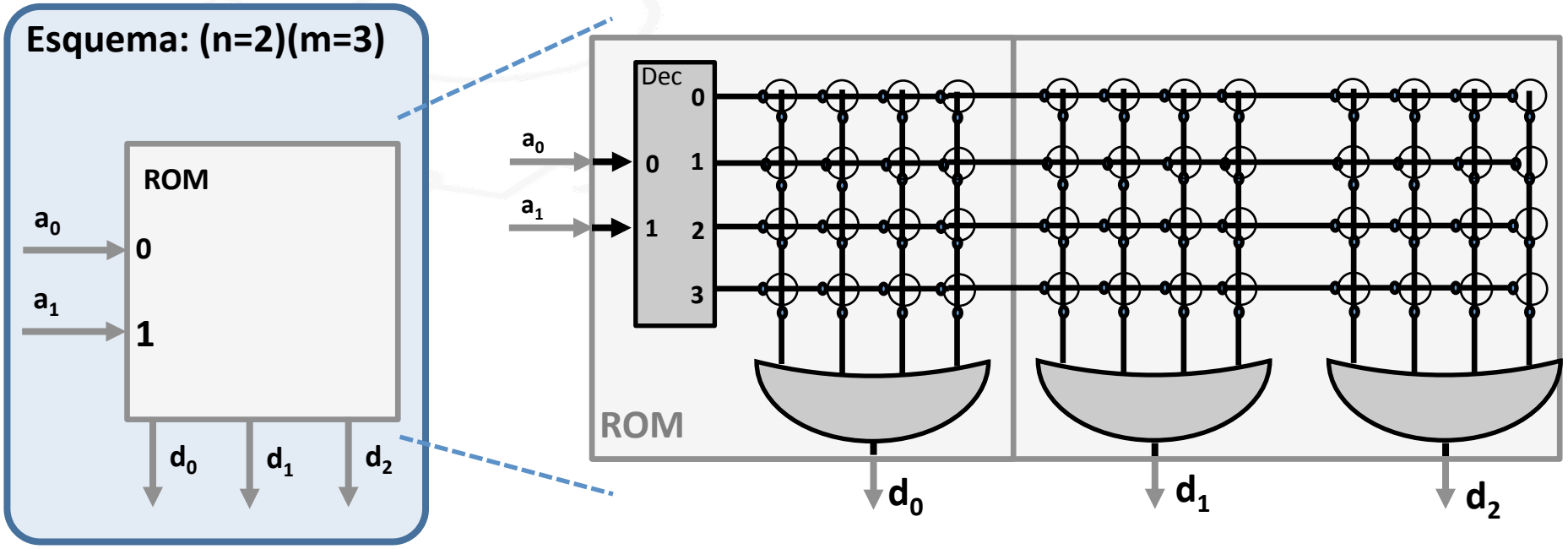
- **Añadiendo Salidas:** ¿qué pasa si el CLC que queremos implementar tiene más de una señal de salida? (m):
  - Con esta configuración podemos implementar cualquier función lógica de n entradas en suma de minterms.

Esquema: (n=2)(m=3)



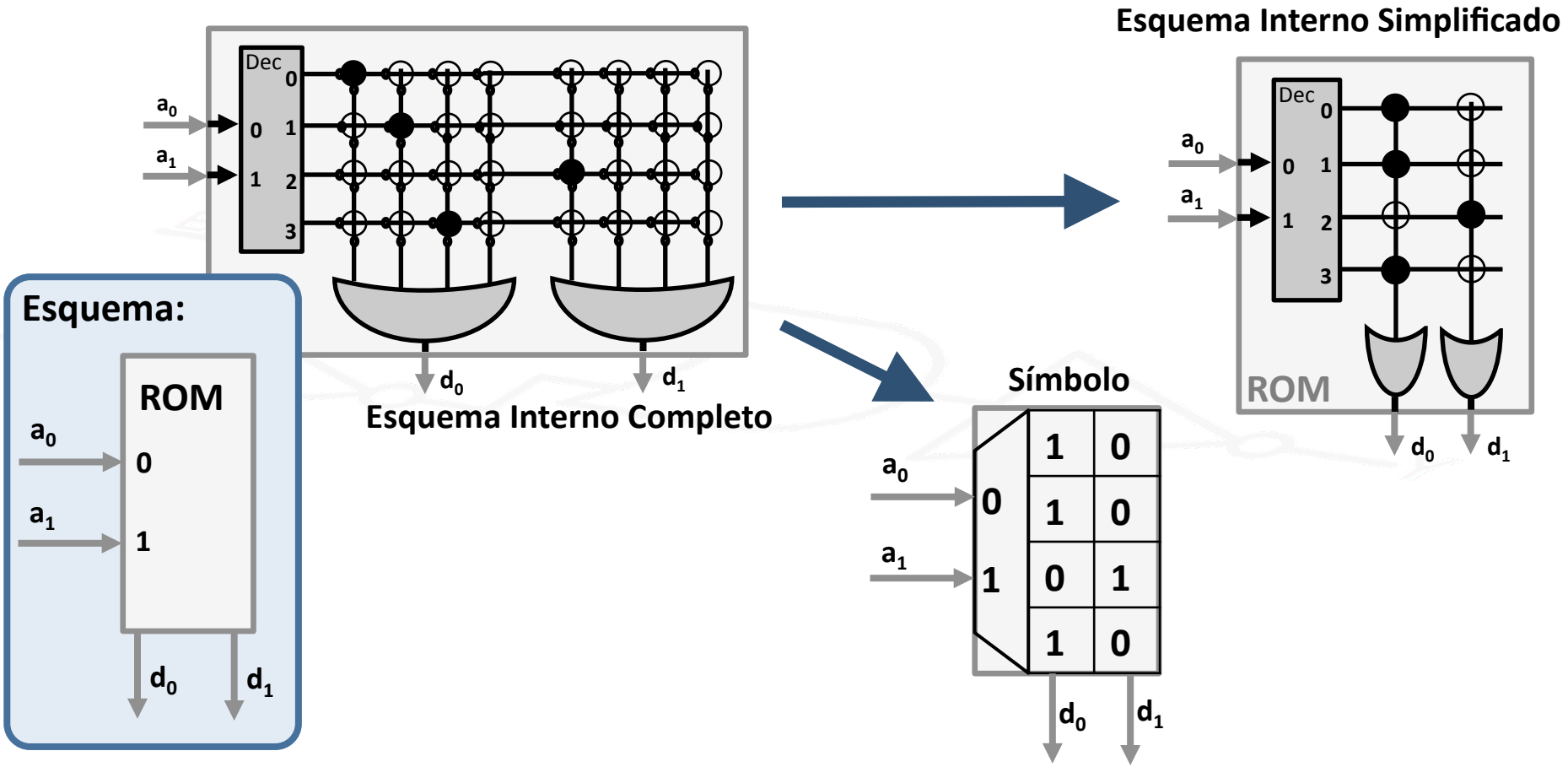
Análisis y **Síntesis**

- **Añadiendo Salidas:** ¿qué pasa si el CLC que queremos implementar tiene más de una señal de salida? (m):
  - Con esta configuración podemos implementar cualquier función lógica de n entradas en suma de minterms.



Análisis y **Síntesis**

- **Simplificando la representación gráfica de una ROM:**



Análisis y **Síntesis**

- **Ejercicio:** sintetizar, empleando una ROM, el siguiente CLC:

Tabla de Verdad

Pi	Xk	Zf	An	Po	Mt
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	0	0