Introducción al Algebra de Boole
Principios de Lógica Digital

Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Departamento Sistemas
Arquitectura de Computadores
2011
Elvira Quiroga

Esta presentación es de carácter conceptual y tiene la finalidad de acompañar el desarrollo del tema en el aula.

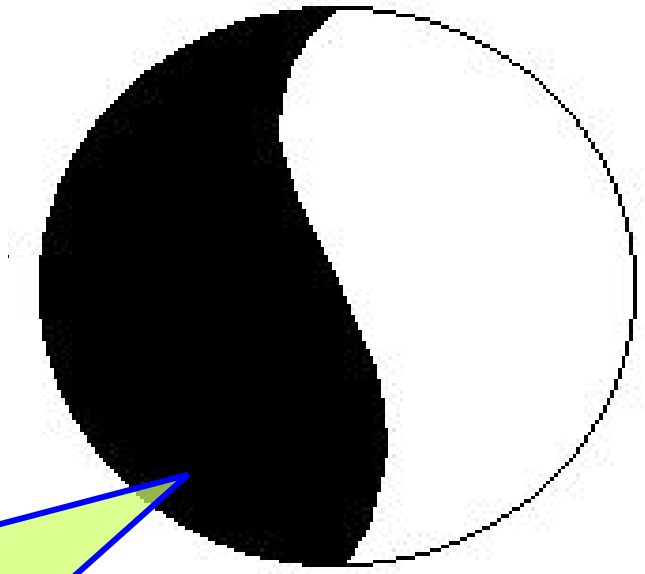
***NO** sustituye la bibliografía indicada por la Cátedra.*

Introducción

- En esta Unidad se analizan los principios básicos de la lógica digital que se aplican al diseño de una computadora digital.
- En 1884 **George Boole** publicó su trabajo sobre un Álgebra para representar la Lógica. Boole estaba interesado en capturar la matemática del pensamiento y desarrolló una representación para las declaraciones como "la puerta está abierta" o "la puerta está no abierta".
- El Álgebra de Boole, en su forma actual fue desarrollada por Shannon.

Algebra de Boole

- En el Algebra de Boole las variables son binarias y sólo pueden tomar dos valores que son complementarios entre si.
- Estos valores se designan como:
- **1** SI / ALTO / **V**ERDADERO / ON
- **0** NO / BAJO/ **F**ALSO / OFF

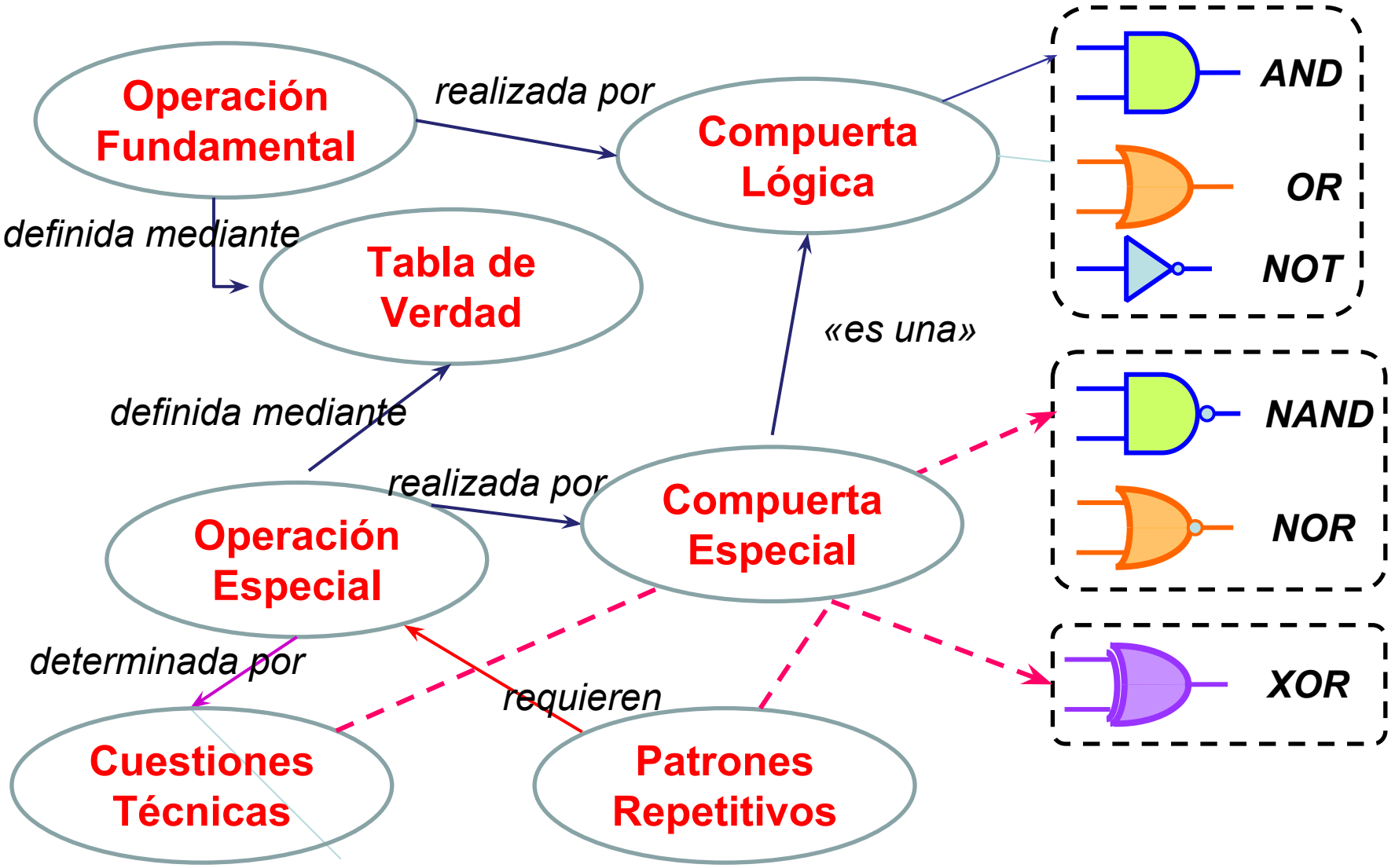


Recordar que en este símbolo si la porción clara representa un estado o valor **A**, la porción oscura representa el estado complementario, "**No A**".

Operadores Fundamentales

- El algebra de Boole reconoce dos operadores fundamentales:
  - SUMA LOGICA (+; OR):
  - PRODUCTO LOGICO (·; AND).
- Algunos autores también consideran al COMPLEMENTO (NO) entre las operaciones fundamentales.
- Estos operadores y cualquier función booleana quedan definidos mediante sus **Tablas de Verdad**.

Esquema Conceptual

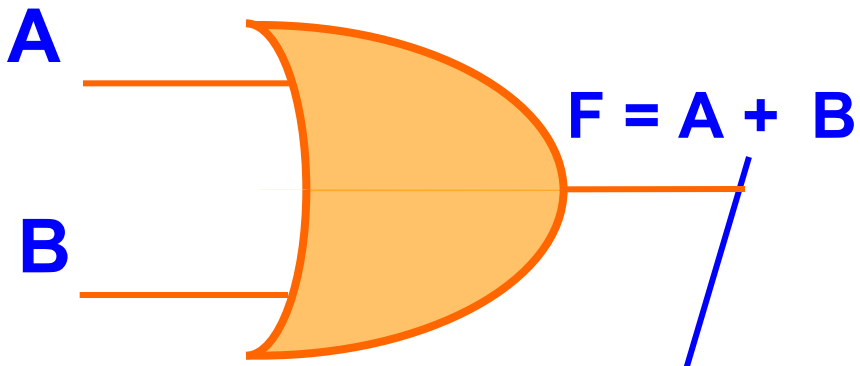


Compuertas Lógicas

- Los dispositivos físicos que implementan una función booleana simple –es decir un operador booleano, se denominan compuertas lógicas o simplemente **compuertas.**
- En lenguaje técnico también se las denomina *«gates»*

Compuertas Lógicas: OR

- ***Realiza la <u>Suma Lógica</u>***
- ***La función lógica OR es Falsa sólo cuando todas las variables de entrada están en "0"***

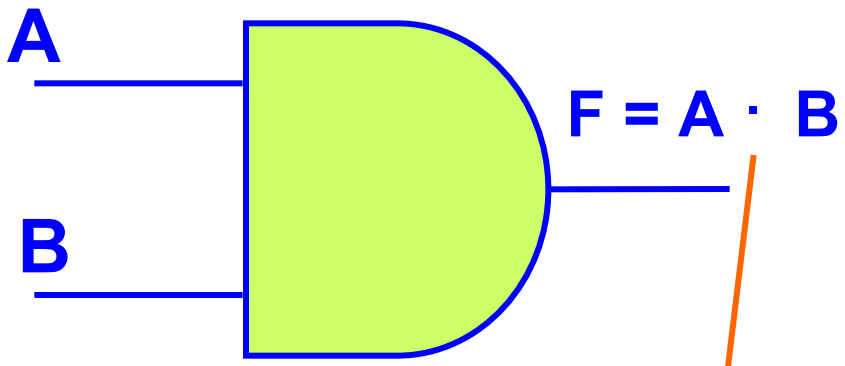


A	B	F
0	**0**	**0**
0	1	1
1	0	1
1	1	1

Se lee A "o" B (OR)

Compuertas Lógicas: AND

- ***Realiza el Producto Lógico***
- ***La función lógica AND es verdadera sólo cuando todas las variables de entrada están en "1"***



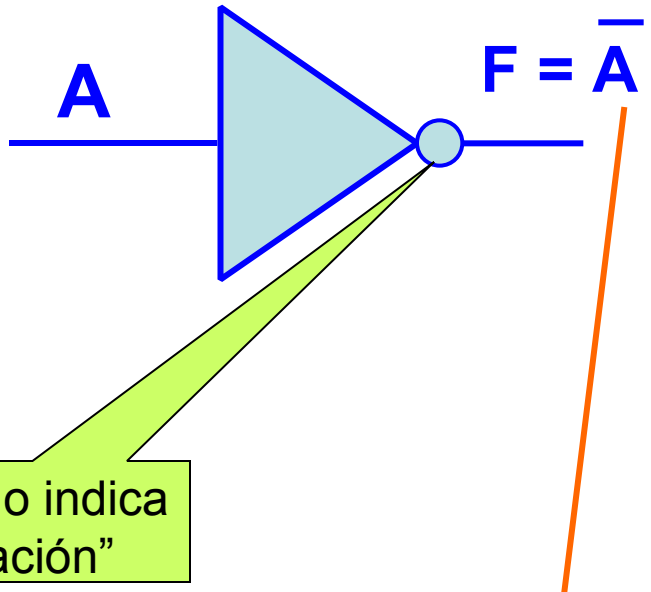
A	B	F
0	0	0
0	1	0
1	0	0
1	**1**	**1**

Se lee A "y" B (AND)

Compuertas Lógicas: NOT

- ***Realiza la Complementaicón***
- ***Este operador "invierte" el valor lógico de la entrada.***

A	F
0	1
1	0

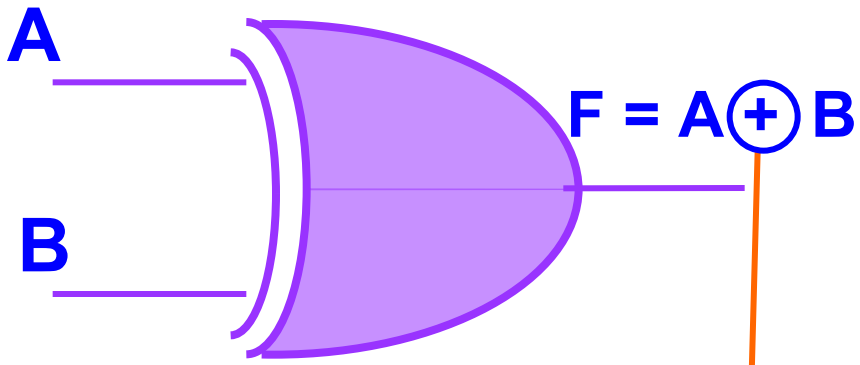


Se lee "no A" (NOT)

Compuertas Lógicas: OR-EX

- ***Esta función lógica especial, <u>OR-Exclusiva</u> es verdadera sólo cuando es IMPAR la cantidad de variables de entrada que están en "1"***

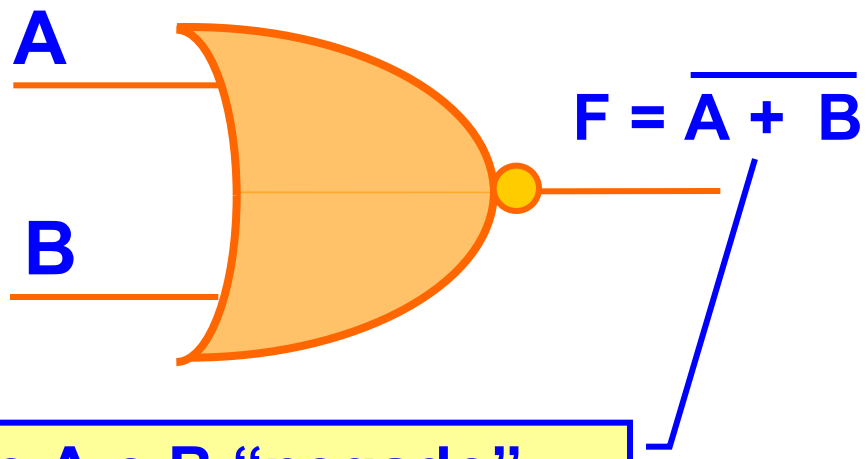
A	B	F
0	0	0
0	**1**	**1**
1	**0**	**1**
1	1	0



Se lee A "exclusive-or" B (OR-EX)

Compuertas Negadas: NOR

- ***La función lógica NOR es Verdadera sólo cuando todas las variables de entrada están en "0"***

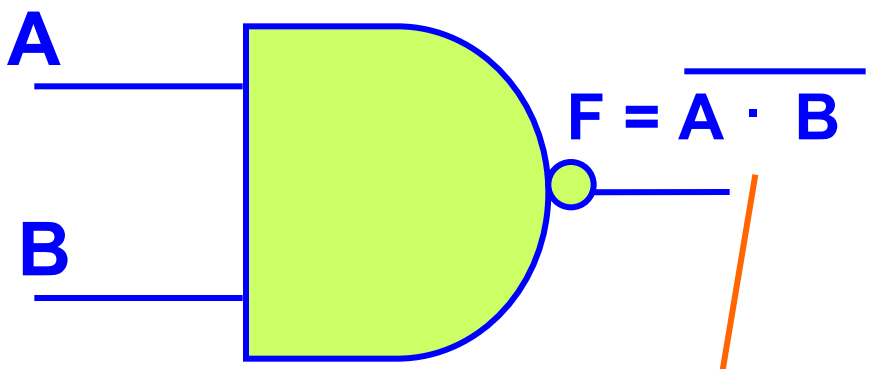


A	B	F
0	**0**	**1**
0	1	0
1	0	0
1	1	0

Compuertas Negadas: NAND

- ***La función lógica NAND es Falsa sólo cuando todas las variables de entrada están en "1"***

A	B	F
0	0	1
0	1	1
1	0	1
1	**1**	**0**



Se lee A y B "negado"(NAND)

Propiedades del Algebra de Boole

- El Algebra de Boole tiene un conjunto de propiedades que se definen en el siguiente conjunto de reglas:
- **Postulados**
  - Conmutativa
  - Distributiva
  - Regla de Identidad
  - Regla del Complemento
- **Teoremas**
  - Asociativa
  - Idempotencia
  - Teorema de De Morgan
  - Regla de Involución
  - Reglas del cero y del uno

Nos proporcionan los formalismos que "soportan" las manipulaciones de las funciones destinadas a optimizarlas –reducir la cantidad de componentes y de conectores

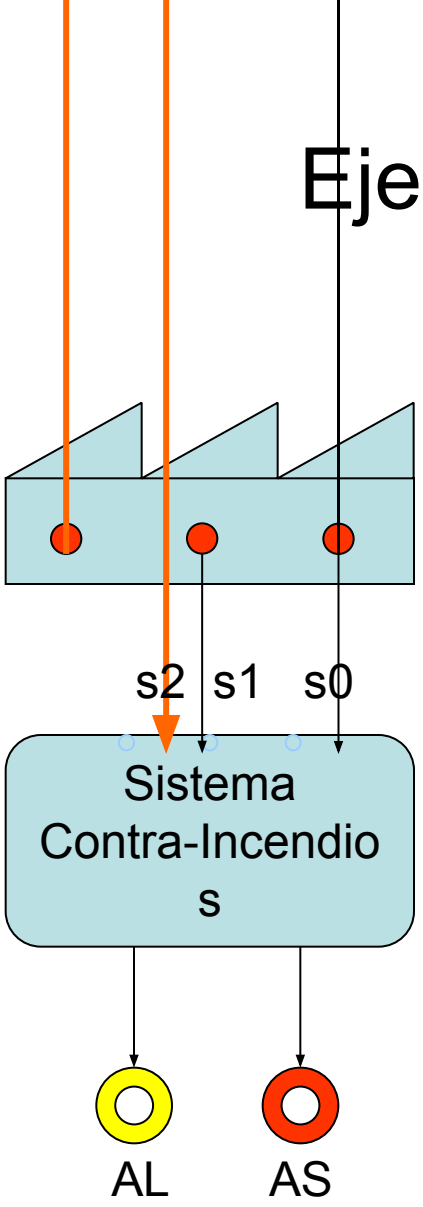
Funciones booleanas

- Un conjunto de variables booleanas vinculadas entre sí mediante los operadores de suma lógica, producto lógico y complementación constituye una ***función booleana***.
  - La Tabla de Verdad es una de las formas de expresar una función booleana.
  - También se usan expresiones literales (***polinómica y factorial***) y expresiones simbólicas.
  - La ***forma canónica*** de un función booleana es una expresión cuyos términos contienen la totalidad de las variables del problema.

Funciones Booleanas: Ejemplo

- El sistema de seguridad contra incendios de un depósito funciona en base a tres sensores S0, S1 y S2. Cuando dos de estos sensores están activados (en “1”) se enciende una alarma luminosa [AL]. Además, si S2 se activa, también se enciende la alarma sonora [AS].

Ejemplo: Tabla de Verdad



s2	s1	s0	AL	AS
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Ejemplo: Formas Literales

- *Expresión Literal Completa –Suma de Productos*

AL = $\overline{S2}$.S1.S0 + S2.$\overline{S1}$.S0 + S2.S1.$\overline{S0}$ + S2.S1.S0

AS = S2.$\overline{S1}$.$\overline{S0}$ + S2.$\overline{S1}$.S0 + S2.S1.$\overline{S0}$ + S2.S1.S0

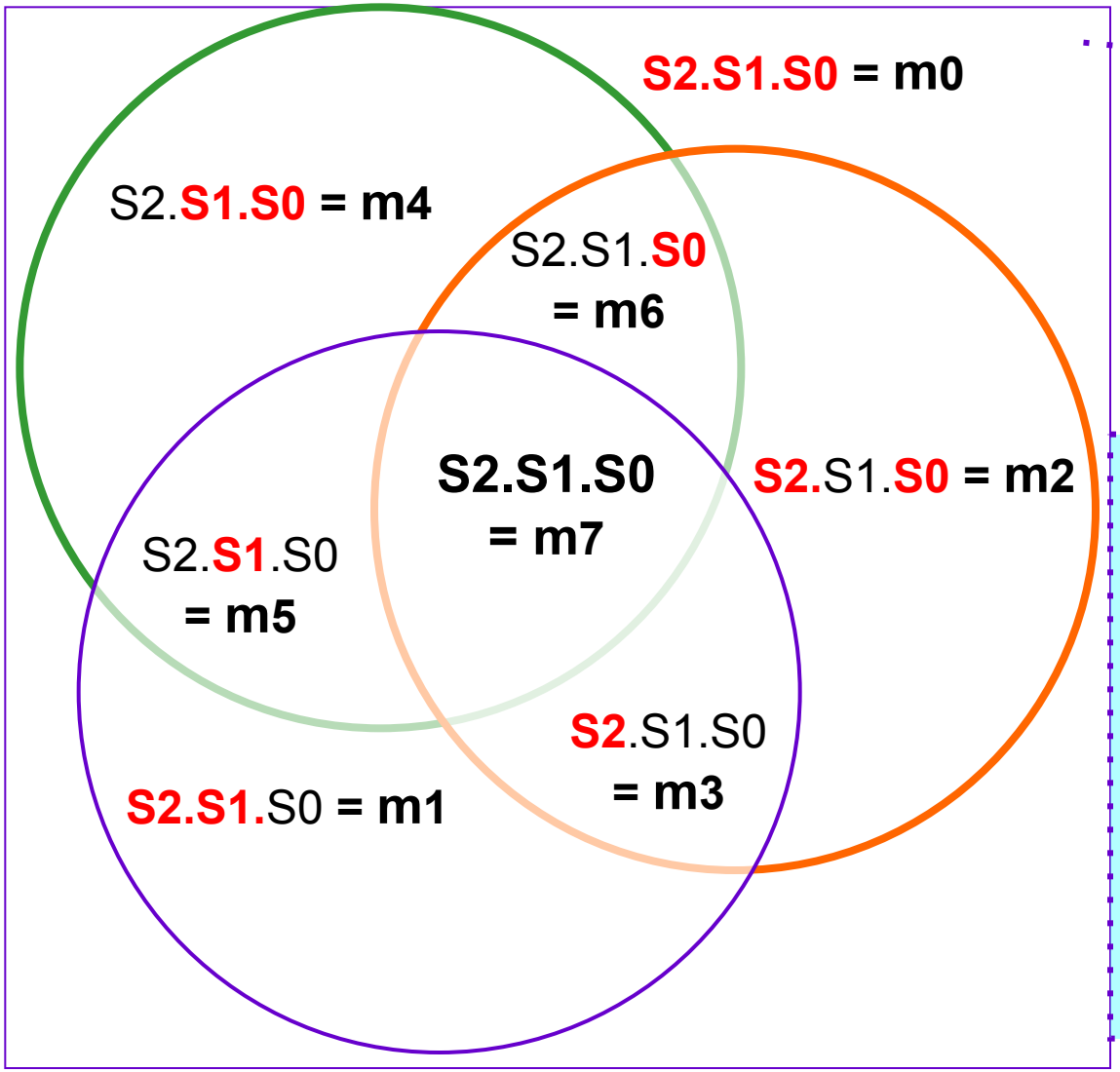
- *Expresión Literal en Minitérminos*

AL = $m_3 + m_5 + m_6 + m_7$

AS = $m_4 + m_5 + m_6 + m_7$

Dado que todos los términos de esta forma POLINÓMICA contienen todas las variables, esta expresión se denomina CANÓNICA

Minitérminos

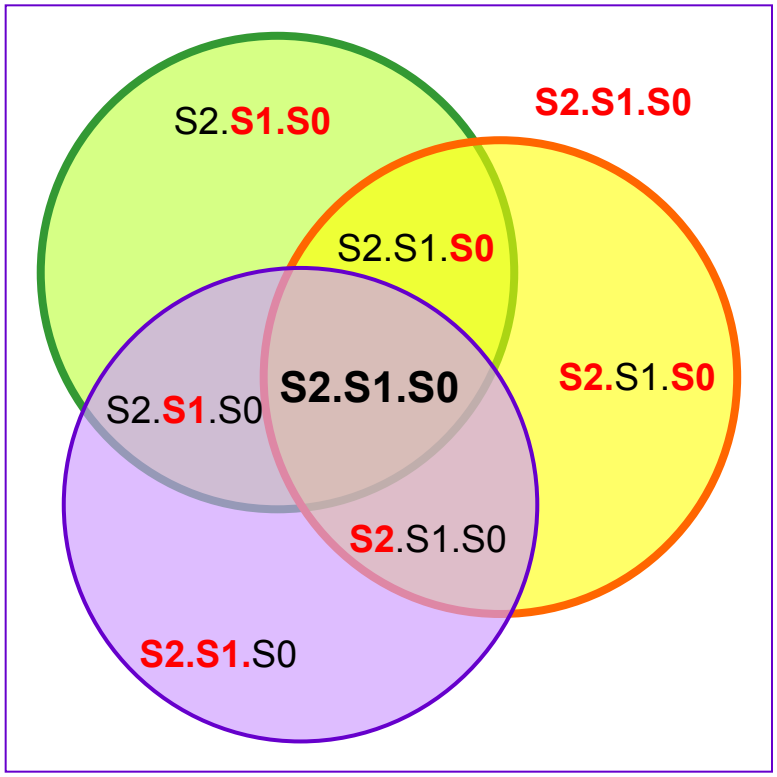


La expresión ***"minitérmino"*** alude a que estos términos designan las áreas mínimas de un diagrama de Euler-Venn.

Ejemplo: Minitérminos

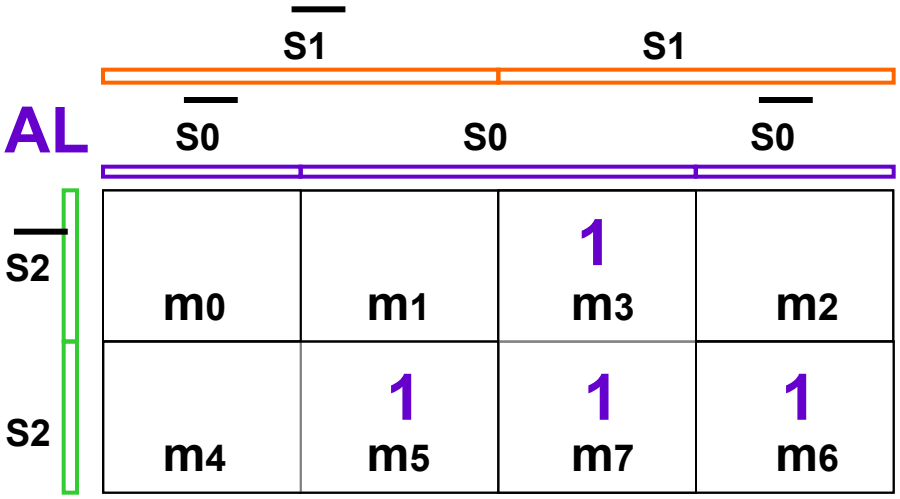
#	s2	s1	s0	Leído como
0	0	0	0	$\mathbf{S_2 S_1 S_0}$
1	0	0	1	$\mathbf{S_2 S_1} S_0$
2	0	1	0	$\mathbf{S_2} S_1 \mathbf{S_0}$
3	0	1	1	$\mathbf{S_2} S_1 S_0$
4	1	0	0	$S_2 \mathbf{S_1 S_0}$
5	1	0	1	$S_2 \mathbf{S_1} S_0$
6	1	1	0	$S_2 S_1 \mathbf{S_0}$
7	1	1	1	$S_2 S_1 S_0$

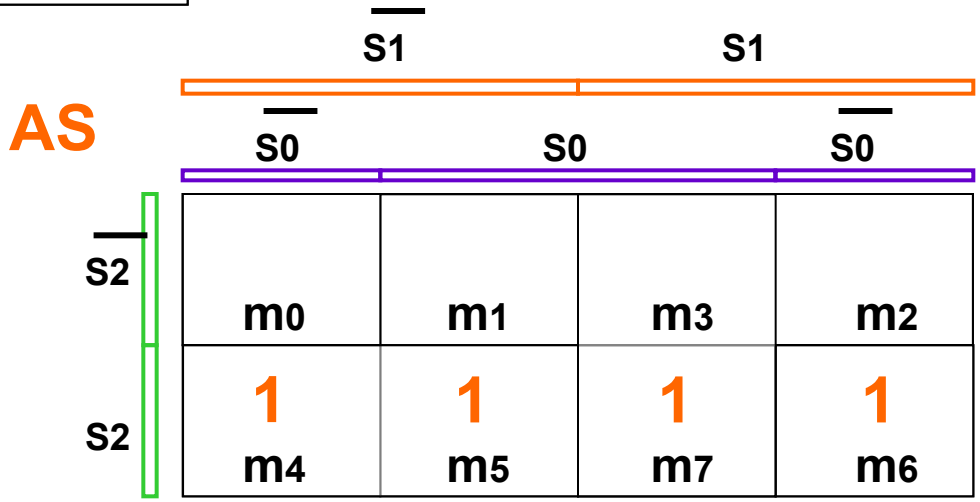
Minitérminos y Mapa de Karnaugh



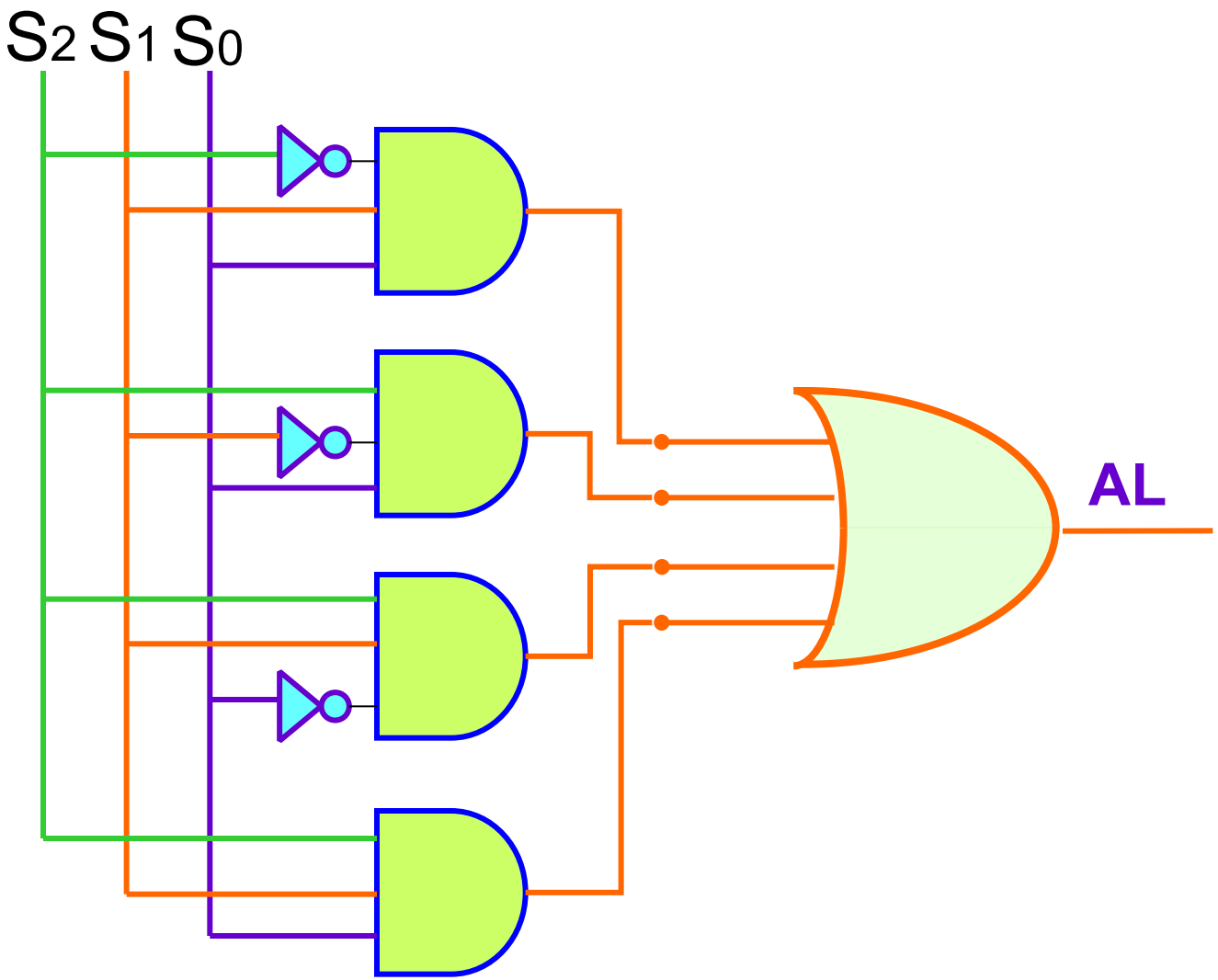
	$\overline{S1}$	S1	S1	$\overline{S1}$
	$\overline{S0}$	S0	S0	$\overline{S0}$
$\overline{S2}$	m0	m1	m3	m2
S2	m4	m5	m7	m6

Ejemplo: Mapas de Karnaugh

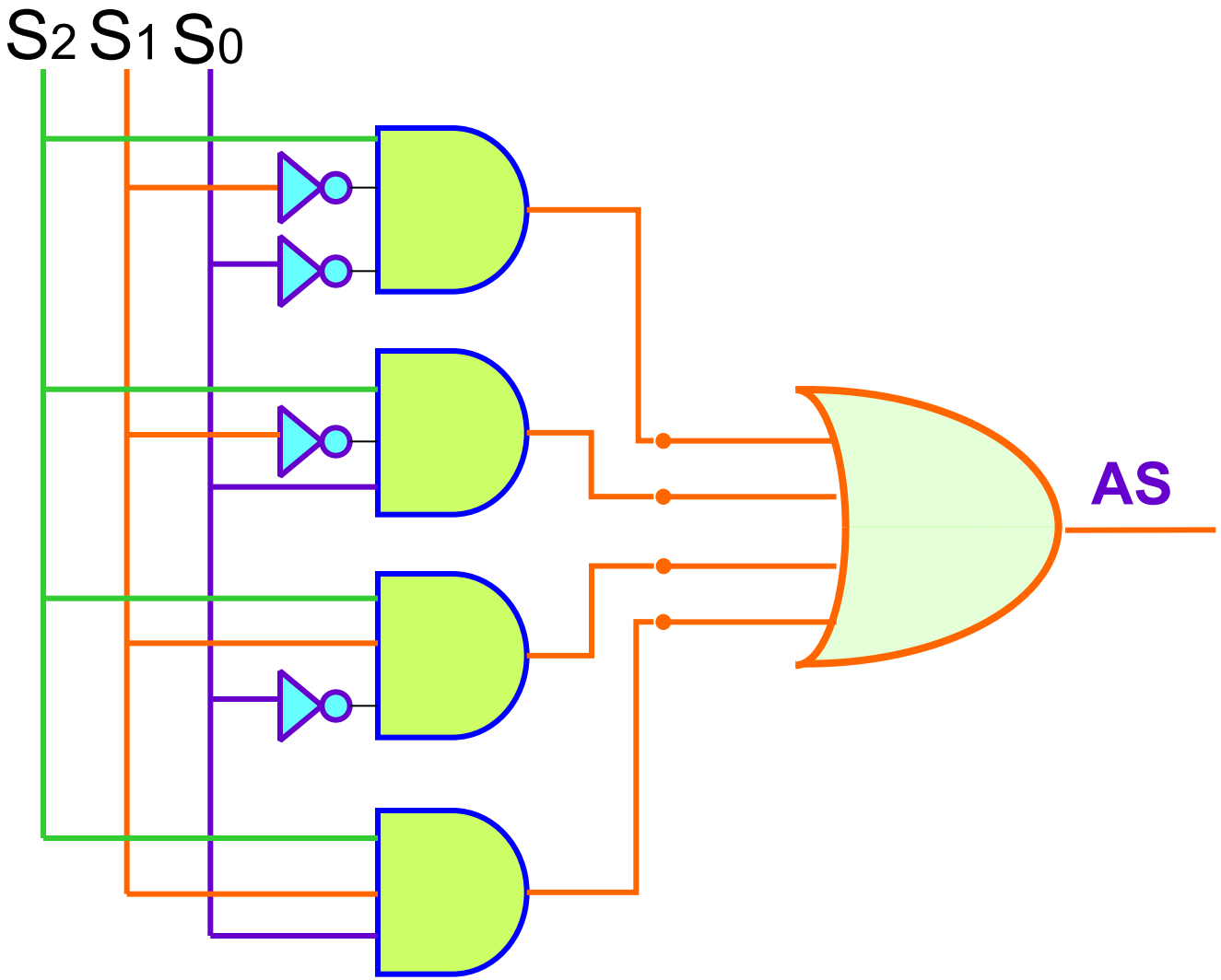




Ejemplo: Circuitos



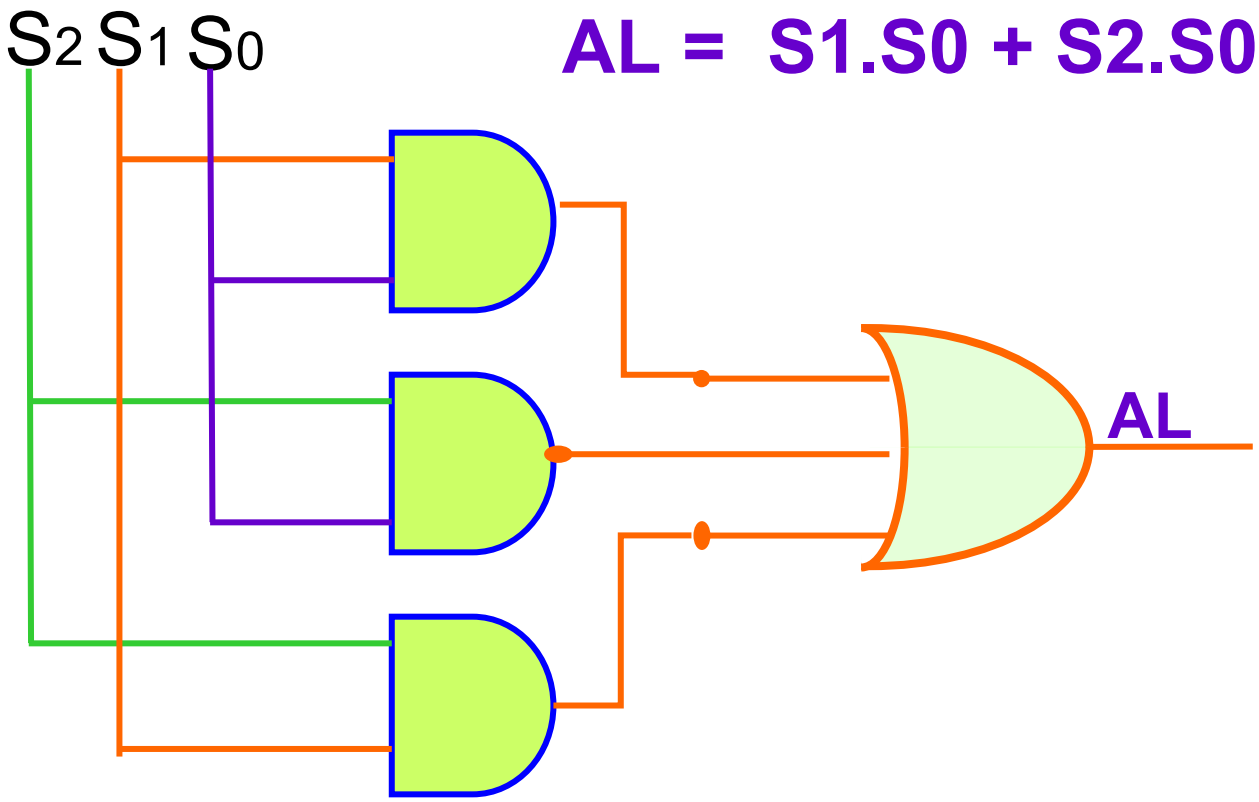
Ejemplo: Circuitos



Ejemplo: Circuitos Minimización

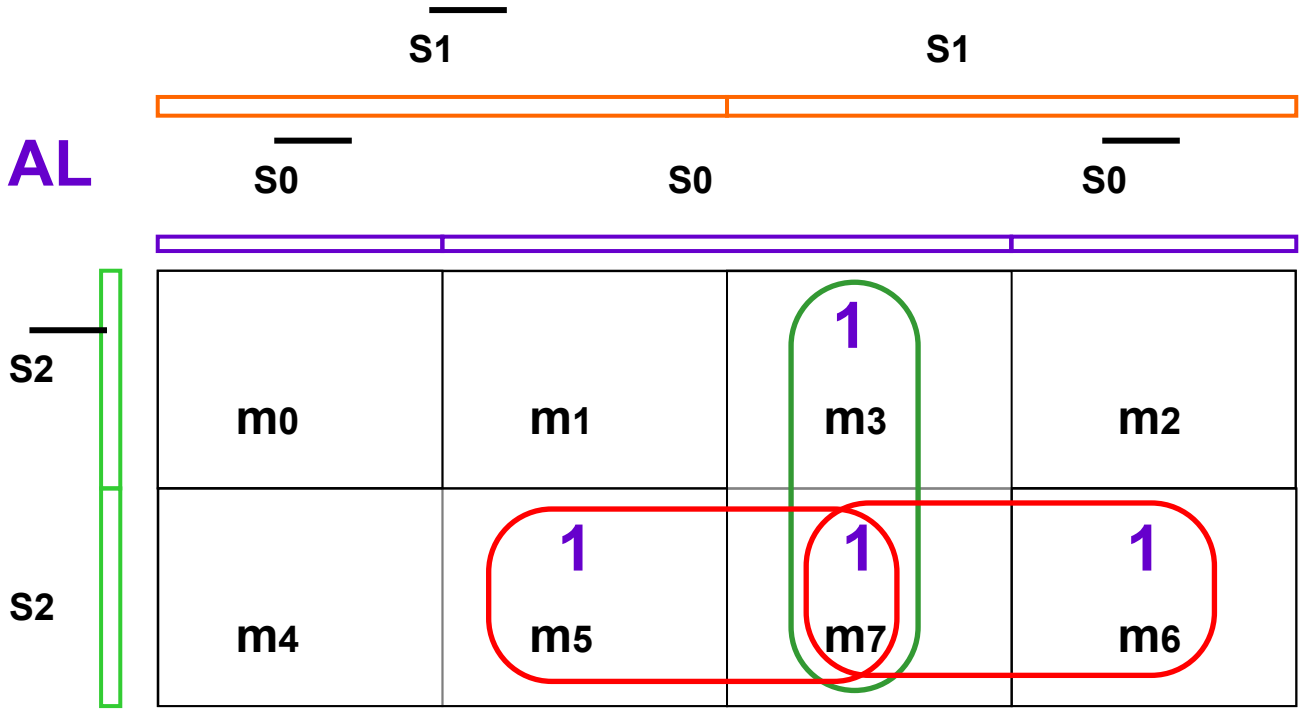
AL = **S2**.S1.S0 + S2.**S1**.S0 + S2.S1.**S0** + S2.S1.S0

AL = (**S2** + S2).S1.S0 + (**S1** + S1).S2.S0 + (**S0** + S0).S2.S1

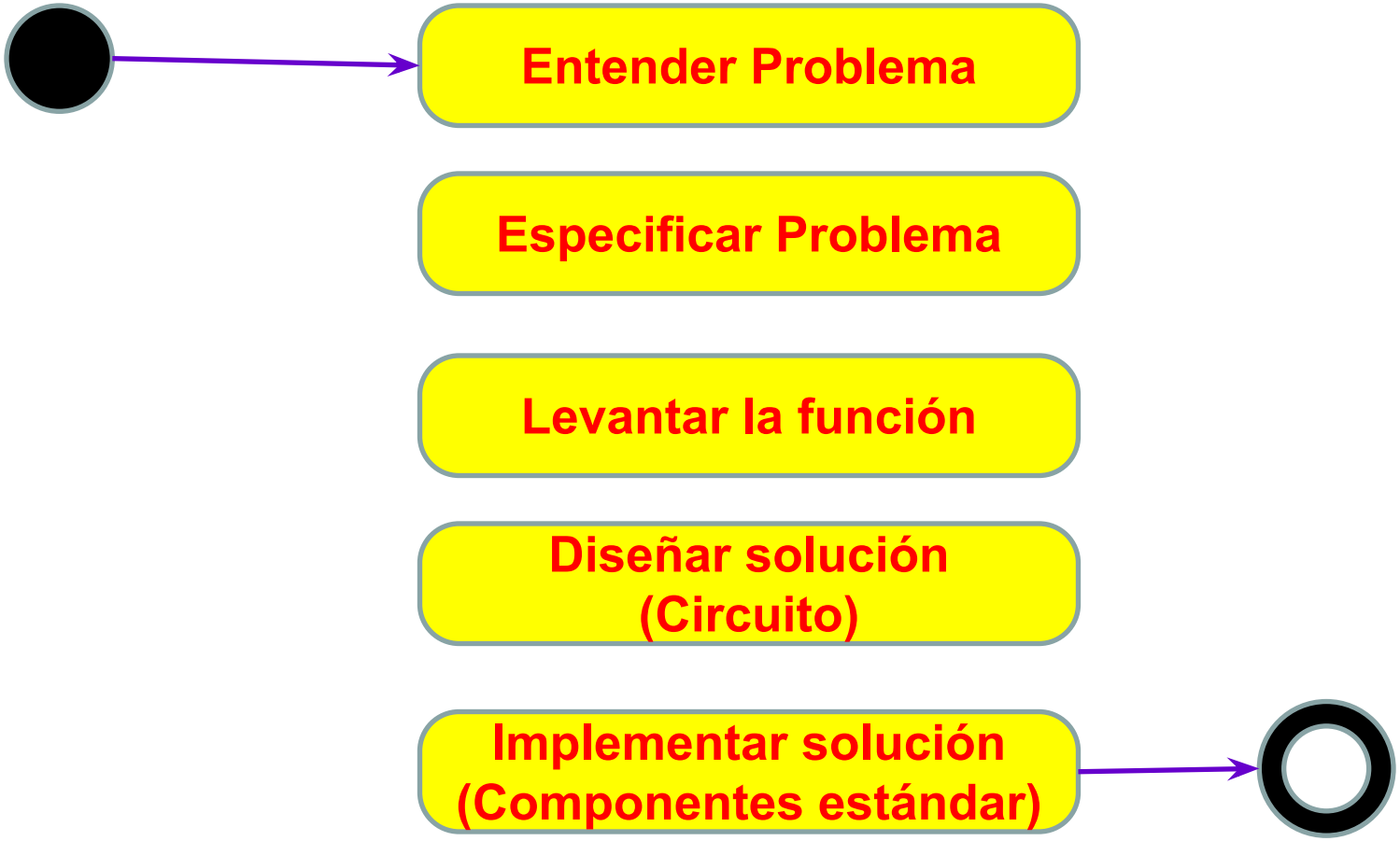


Ejemplo: Circuitos Minimización

AL = **S2**.S1.S0 + S2.**S1**.S0 + S2.S1.**S0** + S2.S1.S0



AL = S1.S0 + S2.S0 + S2.S1

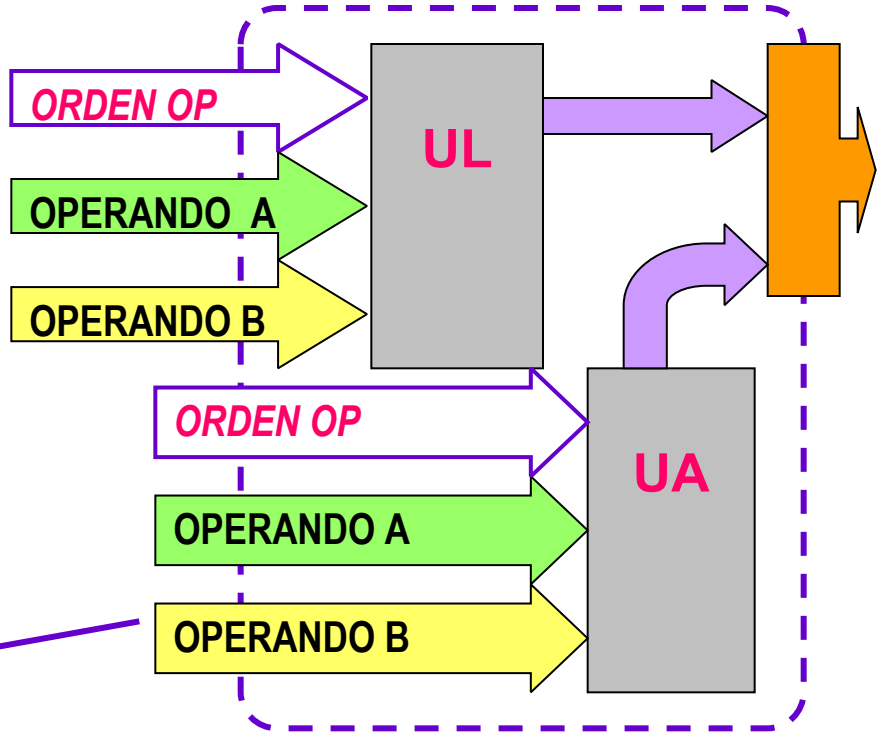

Entender Problema
Especificar Problema
Levantar la función
Diseñar solución
(Circuito)
Implementar solución
(Componentes estándar)

Aplicaciones

Concepto de UAL

- La Unidad Aritmética y Lógica es el componente de la CPU que realiza las operaciones lógicas (AND, OR, XOR, etc.) y aritméticas (en principio suma y resta).

- Podemos pensar entonces en una estructura con dos operadores básicos, uno lógico –**UL** y uno aritmético –**UA**, como muestra la figura adjunta



Como se observa en esta figura, los operadores reciben los operandos (datos) y una orden de operación (para indicar QUE operación deben hacer).

Operadores Elementales

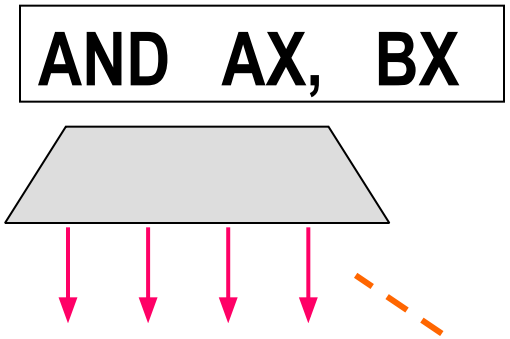
- Un operador elemental es un artefacto “para un bit” que puede realizar una operación aritmética o lógica
  - El **operador lógico elemental** recibirá dos **bits** “a” y “b” que pertenecen a los operandos A y B respectivamente y entregará como resultado el valor que resulte de realizar la operación lógica –AND, OR, XOR, NOT, solicitada.
  - El **operador aritmético elemental** recibirá dos **bits** “a” y “b” pertenecientes a los operandos A y B respectivamente y entregará como resultado el valor que resulte de realizar la operación aritmética –suma o resta, solicitada.

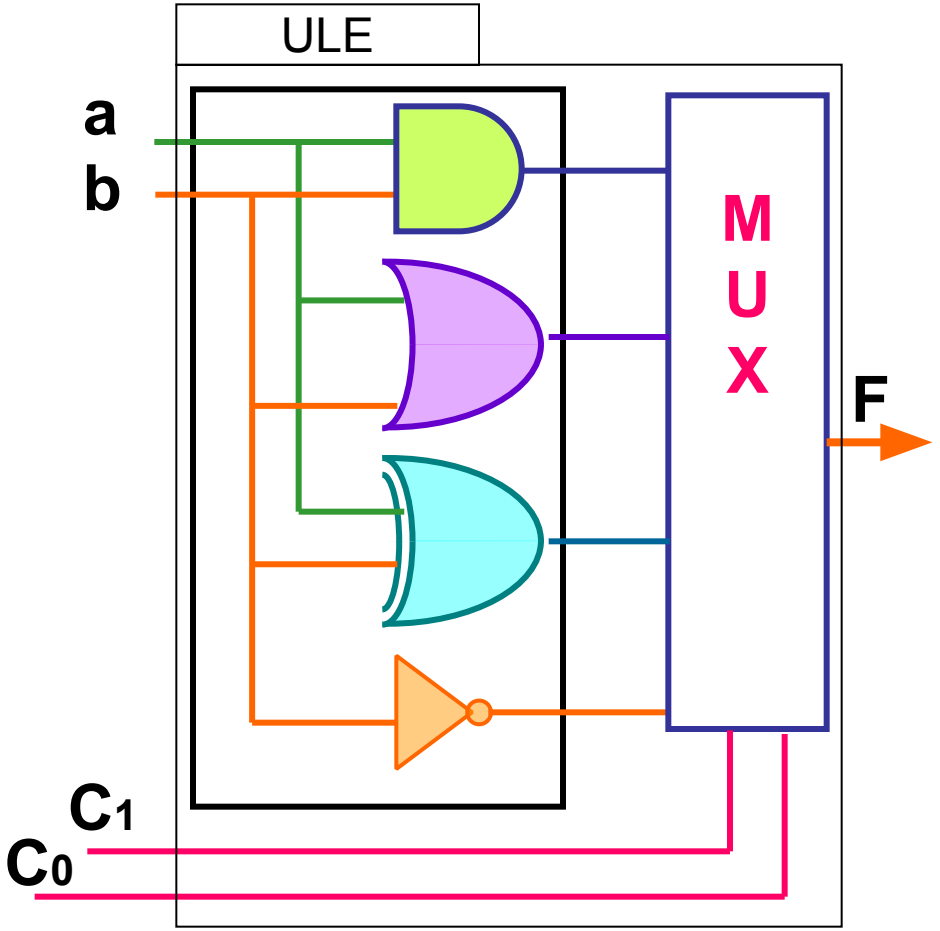
Operador Lógico Elemental

- **Entrada:** Los **bits** “a” y “b” y las señales de control C_0 C_1;
- **Mecanismo Interno:**
  - Realiza simultáneamente **todas** las operaciones lógicas permitidas.
  - Un **multiplexor** permite seleccionar la operación de interés.
  - Los bits de control de este MUX son generados por la Unidad de Control a partir del código de operación.
- **Salida:** El bit cuyo valor resulta de la operación indicada (AND, OR, XOR, NOT)

Unidad Lógica Elemental

C_1	C_0	Función
0	0	$a \cdot b$
0	1	$a + b$
1	0	$a \oplus b$
1	1	$\overline{b}$

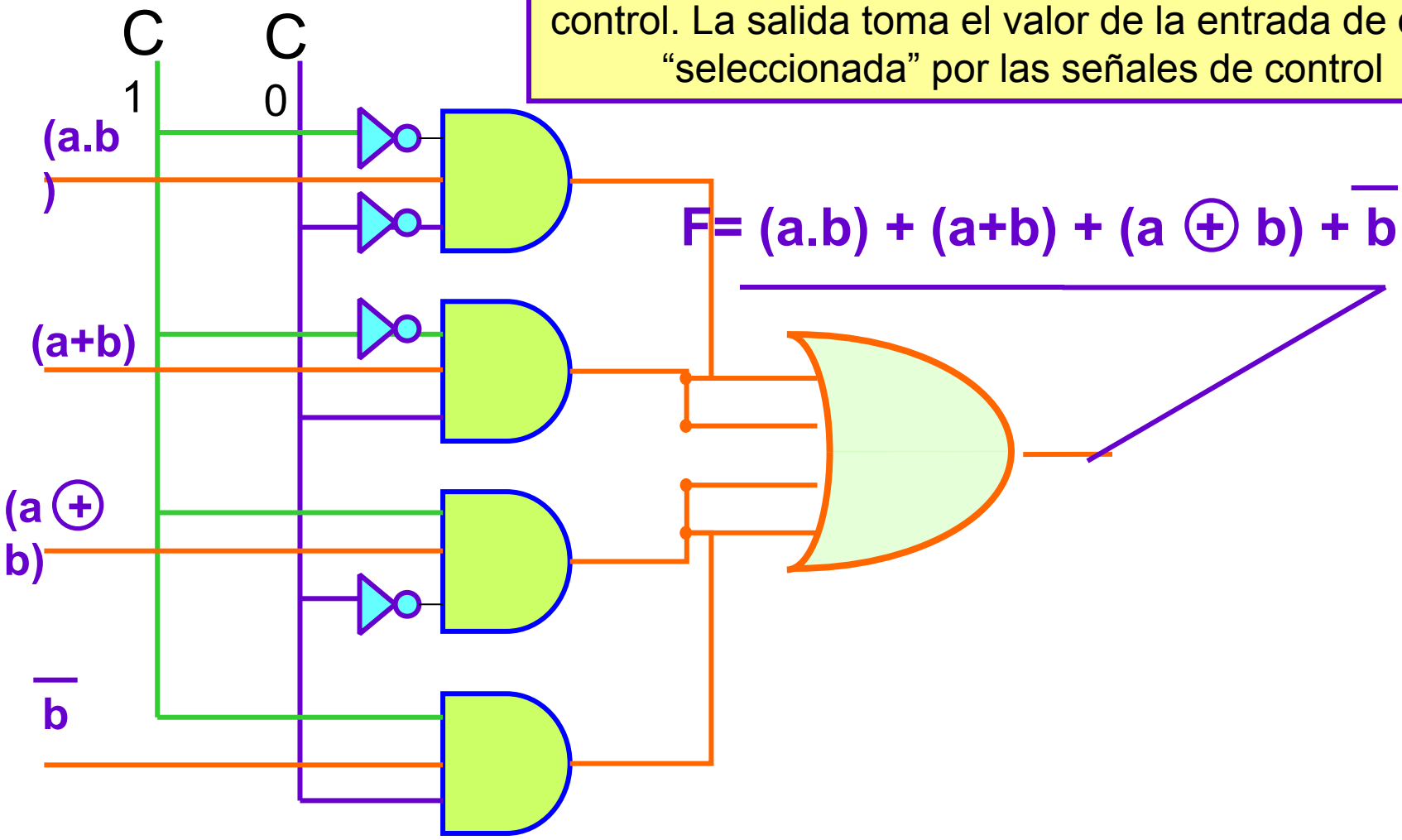




Decodificación de una instrucción

Ejemplo: Circuito Multiplexor

Un MUX tiene 2^n entradas de dato y **n** entradas de control. La salida toma el valor de la entrada de datos "seleccionada" por las señales de control



Unidad Lógica Elemental

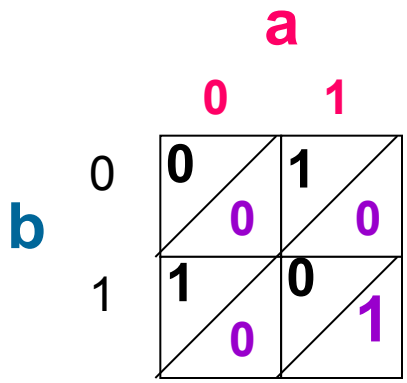
- La conclusión más importante es que aquí vemos cómo se transforma el **código de operación** dado en la instrucción, en las señales binarias –comandos, a nivel del hardware que está involucrado en el funcionamiento de la máquina.

Operador Aritmético Elemental

- **Entrada:** Los **bits** “a” y “b” y la señal de control que indica suma o resta [a+b; a-b];
- **Mecanismo Interno:**
  - Realiza la operación de **suma** sin modificar los bits recibidos y la de **resta** utilizando el complemento auténtico del sustraendo.
  - La operación es ejecutada por un ***sumador elemental***
  - El bit de comando S/R –suma/resta es generado por la Unidad de Control a partir del código de operación (ADD, SUB).
- **Salida:** La suma y el acarreo

Sumador Elemental: Semisumador

- El sumador elemental será capaz de sumar un par de bits “a” y “b”.
- La Tabla de Verdad se deriva a partir de la Tabla de Suma Aritmética.



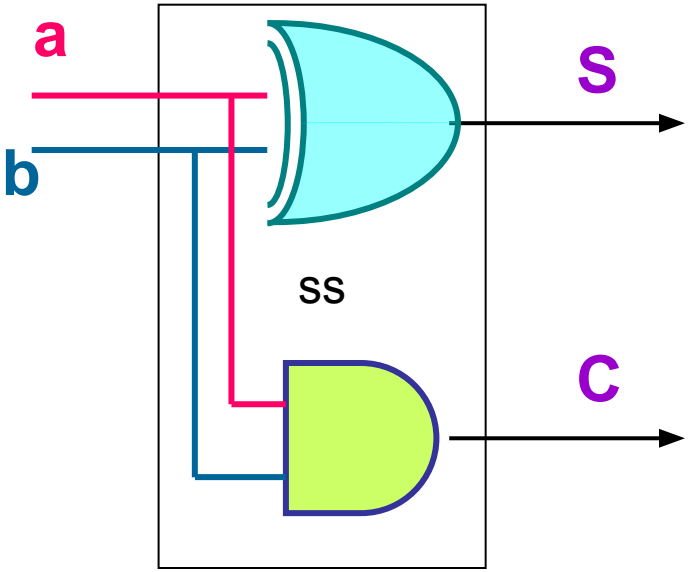
a	b	S suma	C acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

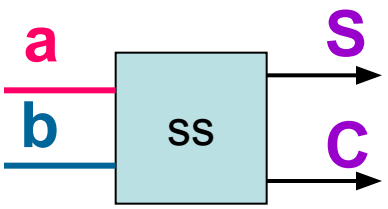
$$S = a \oplus b$$

$$C = a \cdot b$$

Sumador Elemental: Semisumador

- Para sumar dos cantidades de *n* bits es necesario disponer de *n* bloques funcionales como el obtenido.
- Si bien permiten sumar bit a bit, al no tener en cuenta el acarreo no implementan correctamente la suma. Por este motivo este bloque se denomina **semisumador** o *half adder*.





Sumador Elemental: Sumador Completo

- Para realizar la suma de dos cadenas de bits es necesario tener en cuenta el acarreo que cada "etapa" le pasa a la siguiente. La Tabla de Verdad adjunta tiene en cuenta esta situación.
- Ahora las funciones son

$$S = a \oplus b \oplus c_{-1}$$

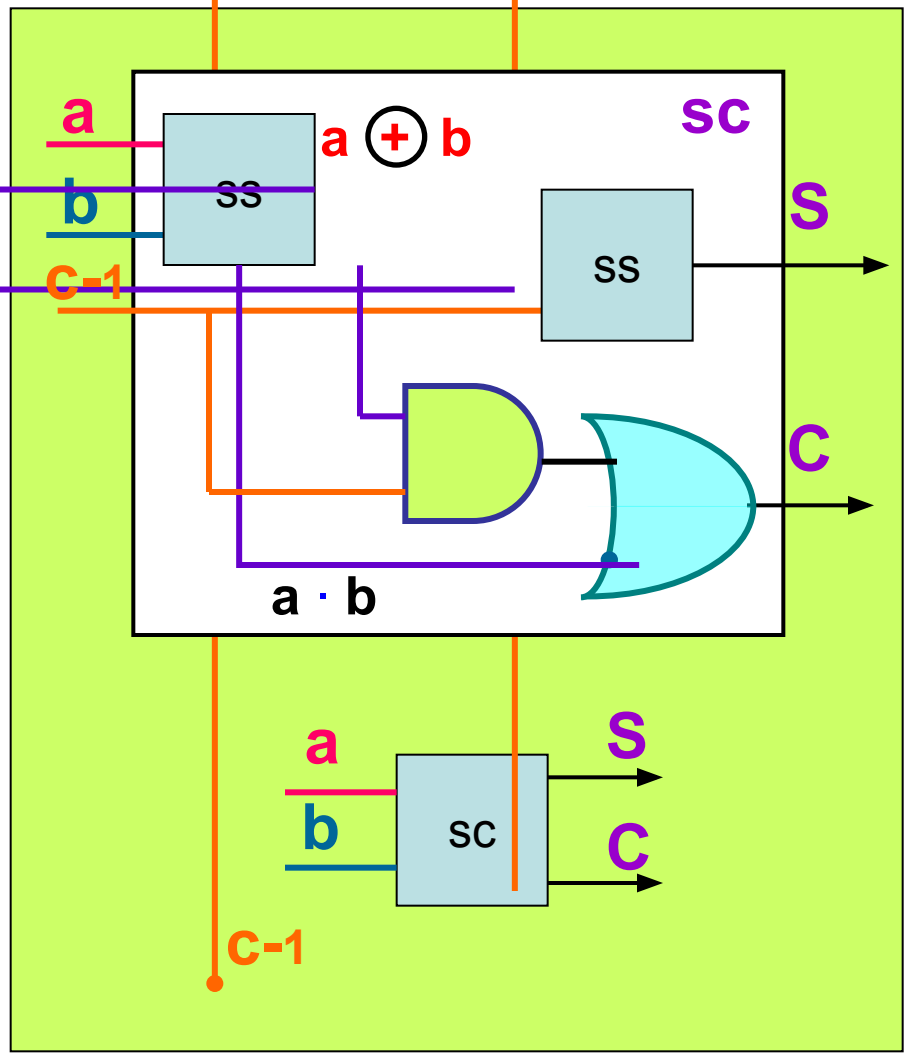
$$C = (a \cdot b) + (a \oplus b) \cdot c_{-1}$$

- El bloque funcional que las implementa se denomina **sumador completo** *o full adder*.

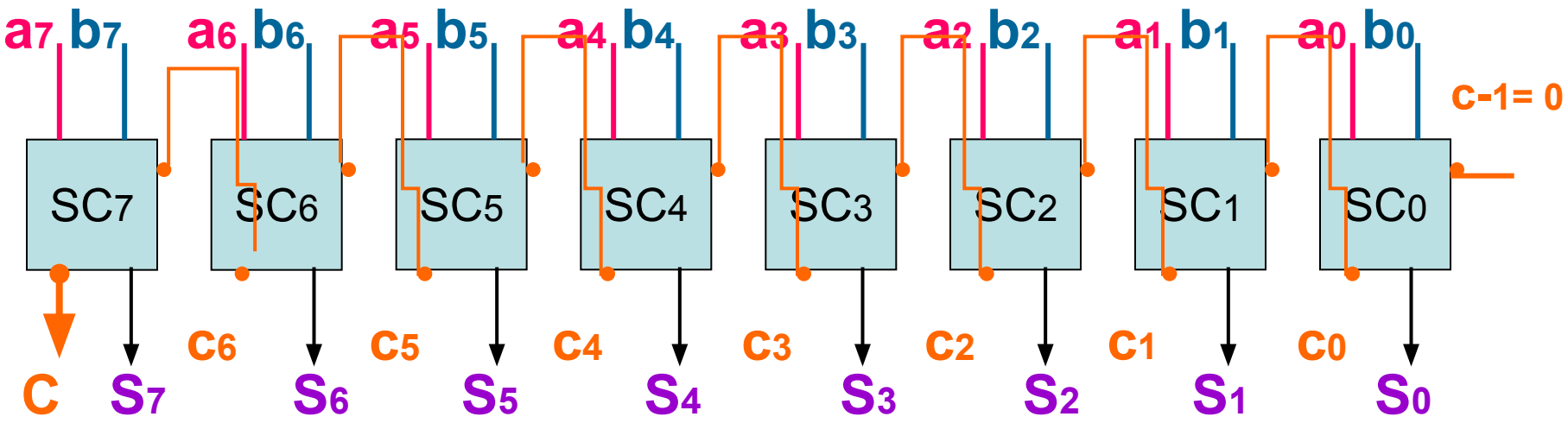
a	b	c_{-1}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	**1**
1	0	0	1	0
1	0	1	0	**1**
1	1	0	0	**1**
1	1	1	1	**1**

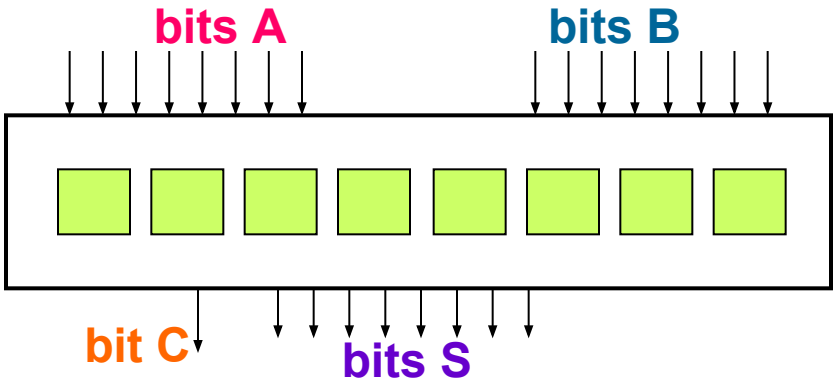
Sumador Elemental: Sumador Completo

- Este bloque se denomina **sumador completo** o *full adder*.
- Permite sumar bit a bit, y dado que tiene en cuenta el acarreo posibilita la implementación correcta de una suma de n bits.
  - Justificar la implementación propuesta para el sumador completo.



Sumador de 8 bits





La Resta en la UAL

- La resta se implementa utilizando el mismo dispositivo sumador, gracias a la propiedad de la notación posicional de cantidades denominada COMPLEMENTO.
- Para un número N de “p” dígitos expresado en base “b” se definen:
  - COMPLEMENTO DIRECTO (a la base menos uno)

    $$CD(N) = (b^p - 1) - N$$

  - COMPLEMENTO AUTENTICO (a la base)

    $$CA(N) = (b^p) - N$$

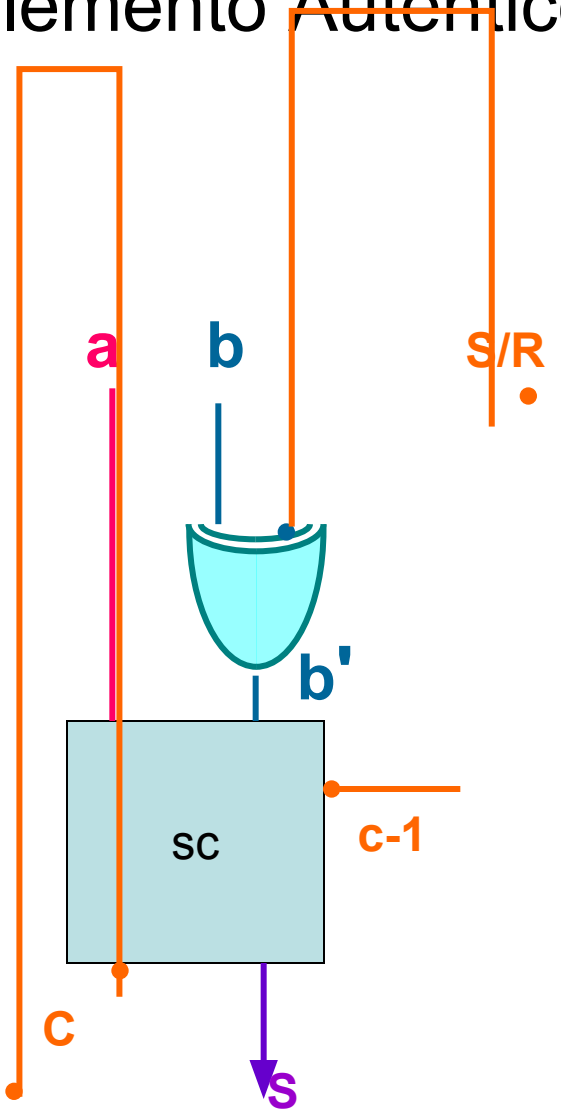
    Y de allí

    $$CA(N) = CD(N) + 1$$
- Según se ha visto, la ***resta*** puede obtenerse “***sumando***” al minuendo el ***complemento auténtico*** del sustraendo.
- En binario el complemento directo se obtiene invirtiendo todos los bits del registro. Por lo tanto, es sencillo deducir el esquema del sumador-restador elemental que realice las operaciones (a + b) y (a - b), según se indique.

Sumador – Restador en Complemento Auténtico

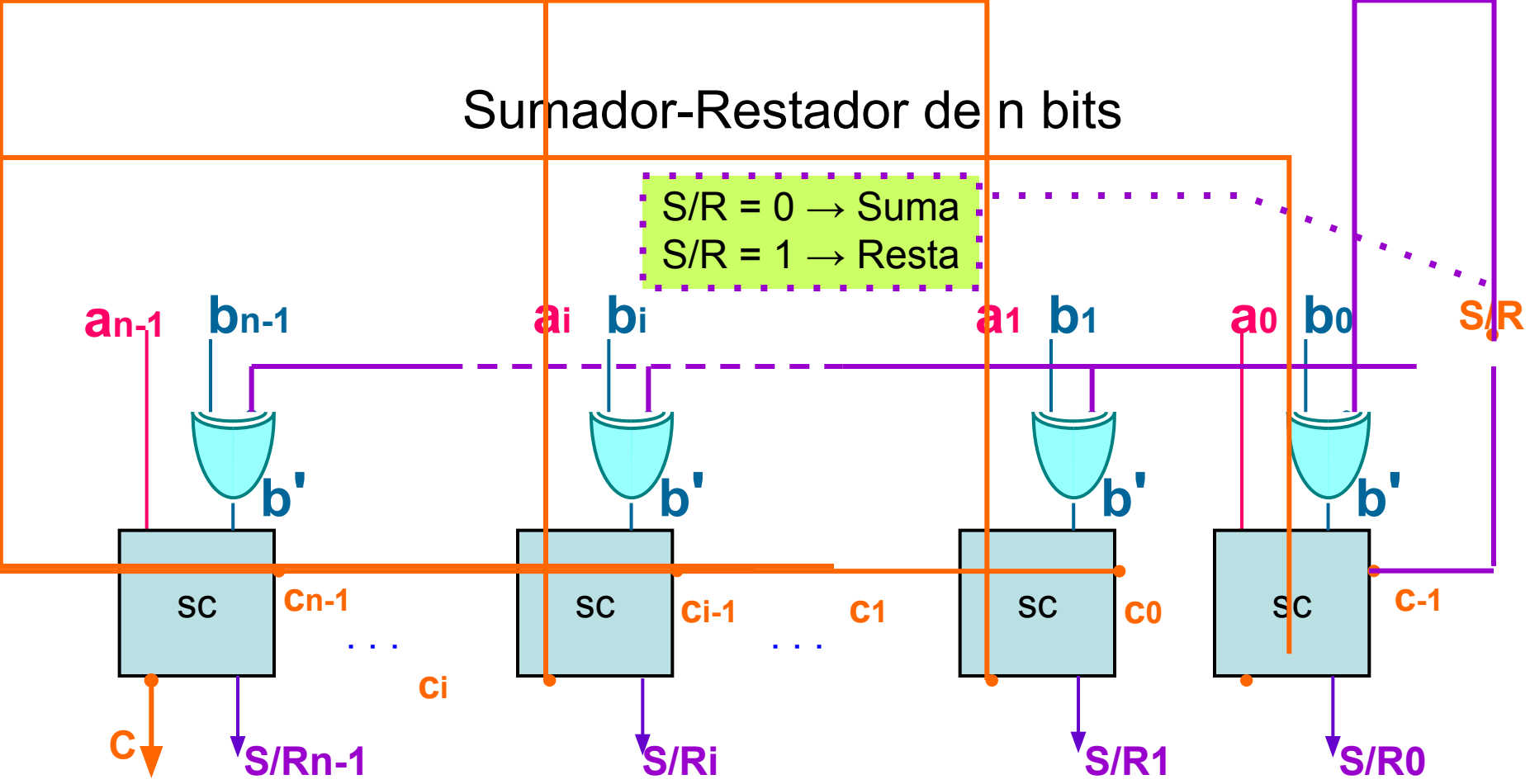
S/R	b	b'
0	0	0
0	1	1
1	**0**	**1**
1	**1**	**0**

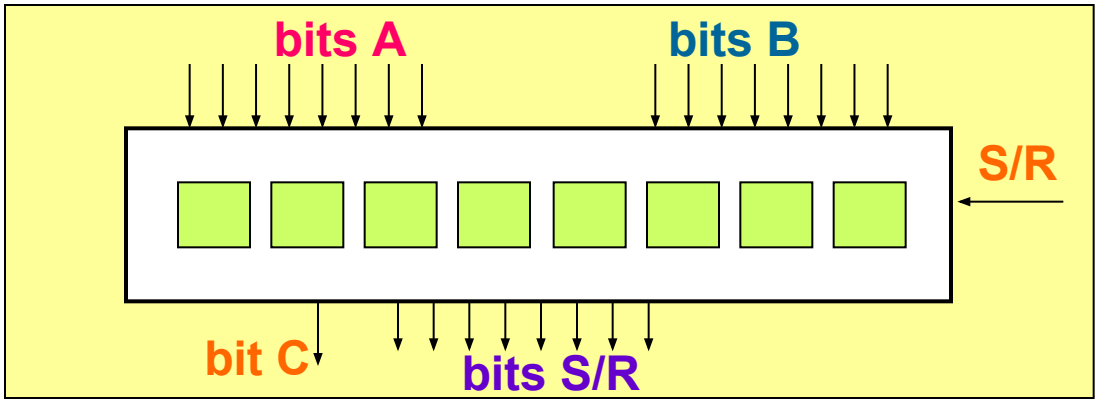
Si S/R es cero (0) el bit “b” se transfiere sin cambios; si es uno (1), el bit transferido es el complemento directo del recibido



Sumador-Restador de n bits

S/R = 0 → Suma
S/R = 1 → Resta





Lógica Digital

Circuitos Combinacionales

- Son dispositivos cuya salida depende exclusivamente de la entrada en un instante dado
- Se implementan mediante compuertas, en forma exclusiva.

Dispositivos Combinacionales Típicos

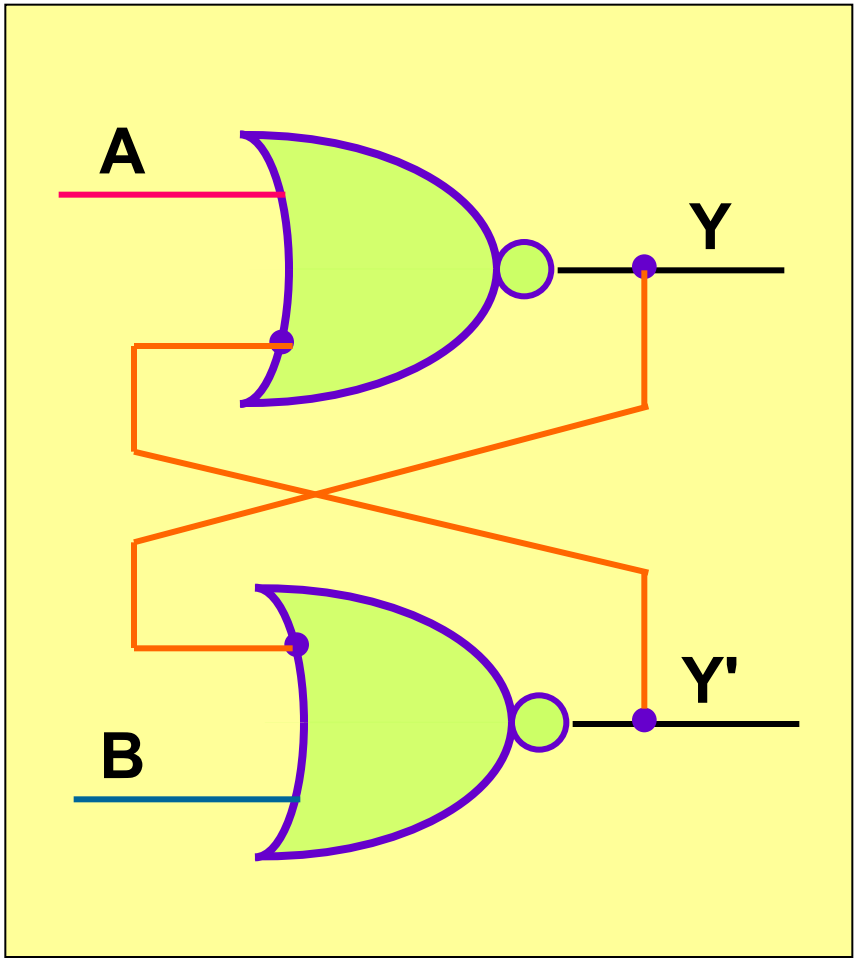
- Generador y Verificador de Paridad;
- Comparador de magnitud;
- Codificadores y Decodificadores
- Multiplexores y Demultiplexores;
- Bus asociado a un Multiplexor-Demultiplexor;
- Circuitos "programables" para multiples funciones;
- Memorias sólo de Lectura;
- Dispositivos tipo PLD

Circuitos Secuenciales

- En estos dispositivos el valor de la salida depende de la entrada y del historial, es decir de las entradas y consecuentes salidas previas.
- Se puede decir que un dispositivo secuencial se caracteriza porque a partir de una entrada y un estado (interno) actual, produce una salida y un nuevo estado interno.
- La implementación de estos dispositivos requiere la incorporación de elementos de memoria destinados a retener los estados internos.

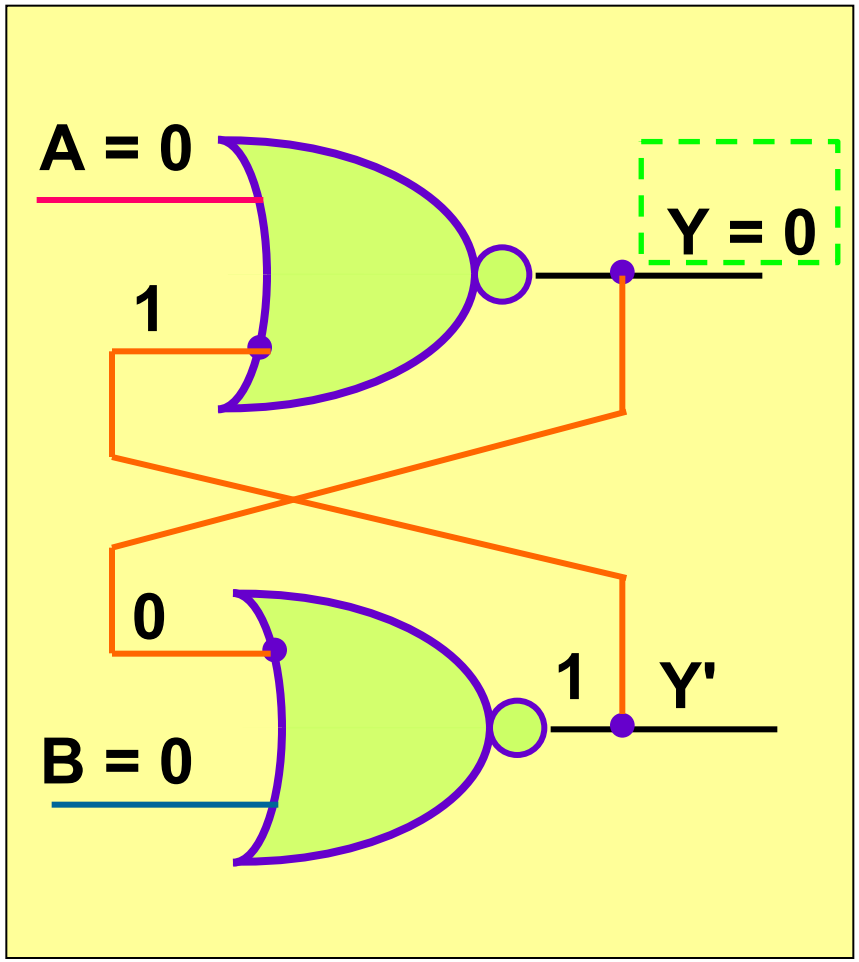
Elementos de Memoria

- Un **elemento de memoria** es un componente electrónico capaz de almacenar el valor de un bit.
- Vamos a considerar el arreglo de compuertas NOR mostrado en la figura adjunta.
- El funcionamiento del dispositivo queda especificado en la Tablas de Funcionamiento que describen como evolucionan los estados conforme cambian las estradas.



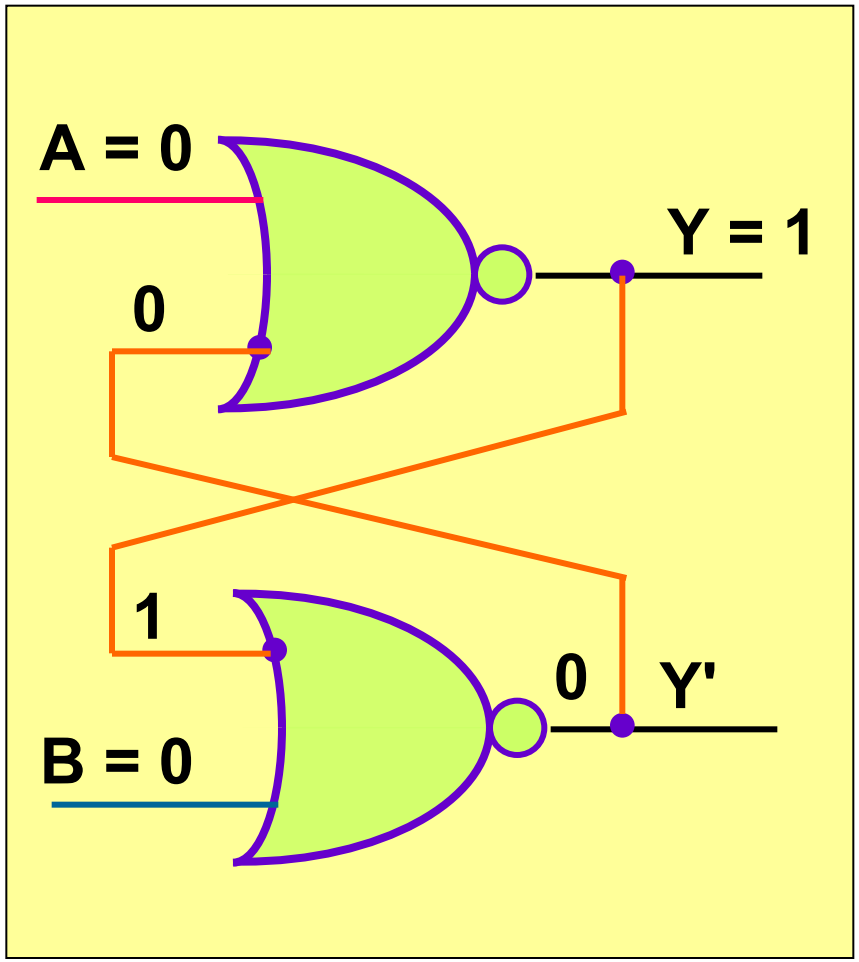
Elementos de Memoria

A= 0; B= 0; Y=0



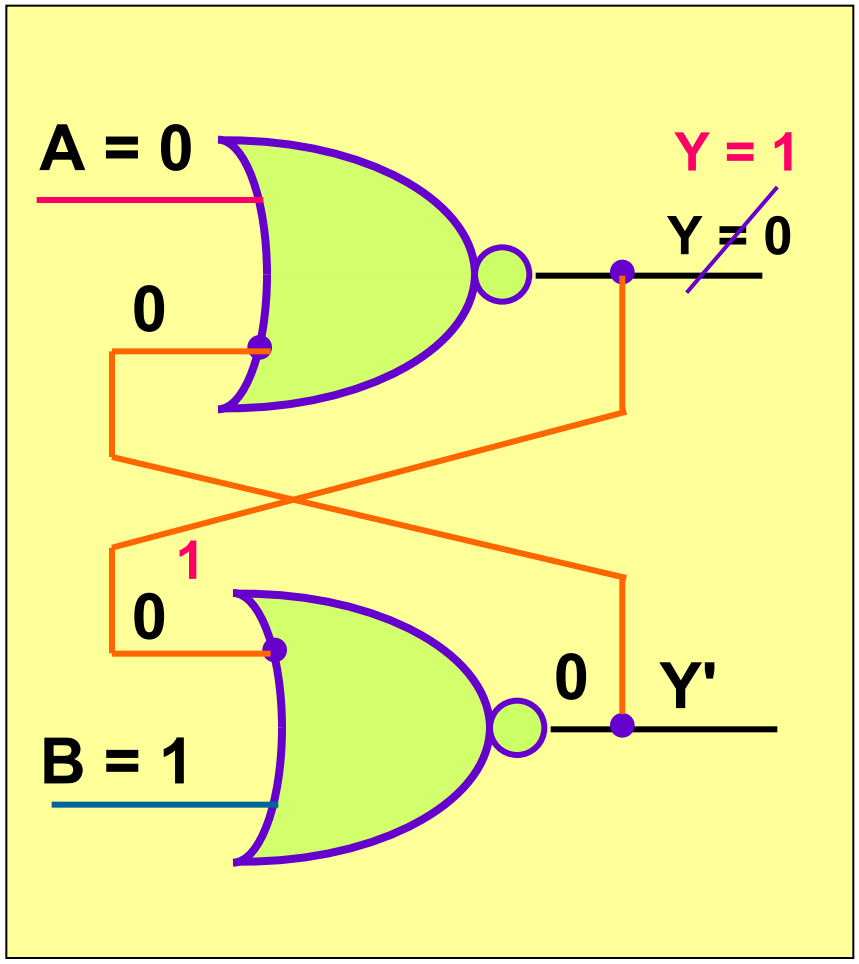
Elementos de Memoria

A= 0; B= 0; Y=1



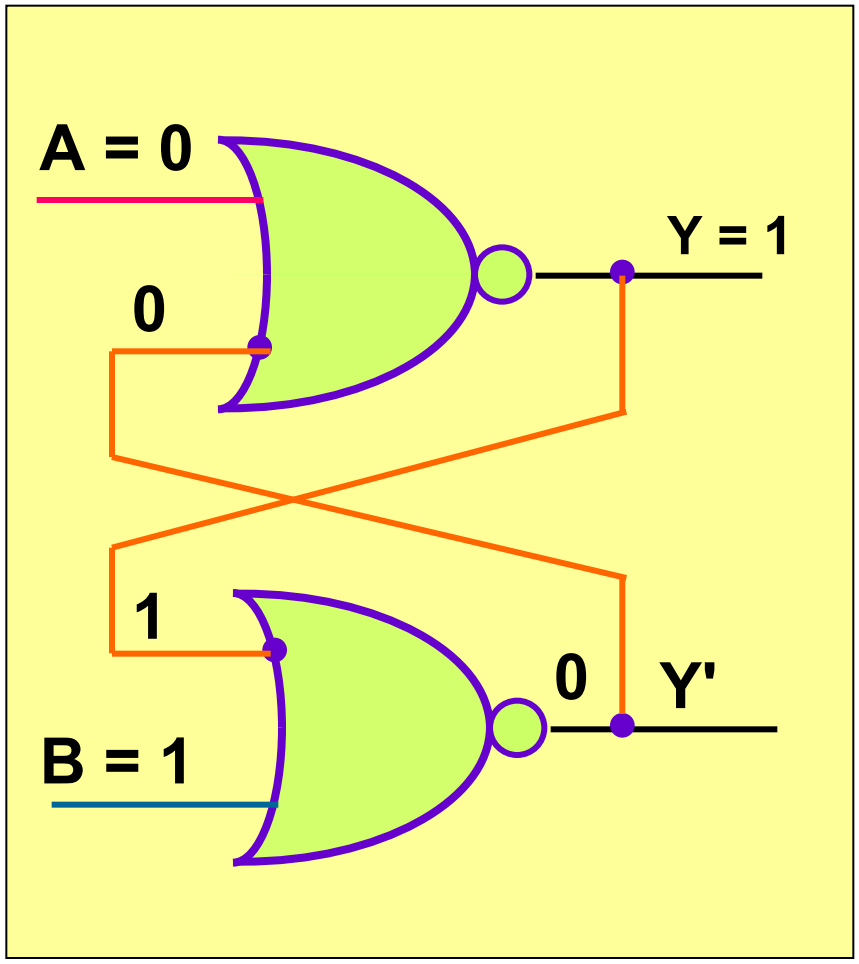
Elementos de Memoria

A= 0; B= 1; Y=0



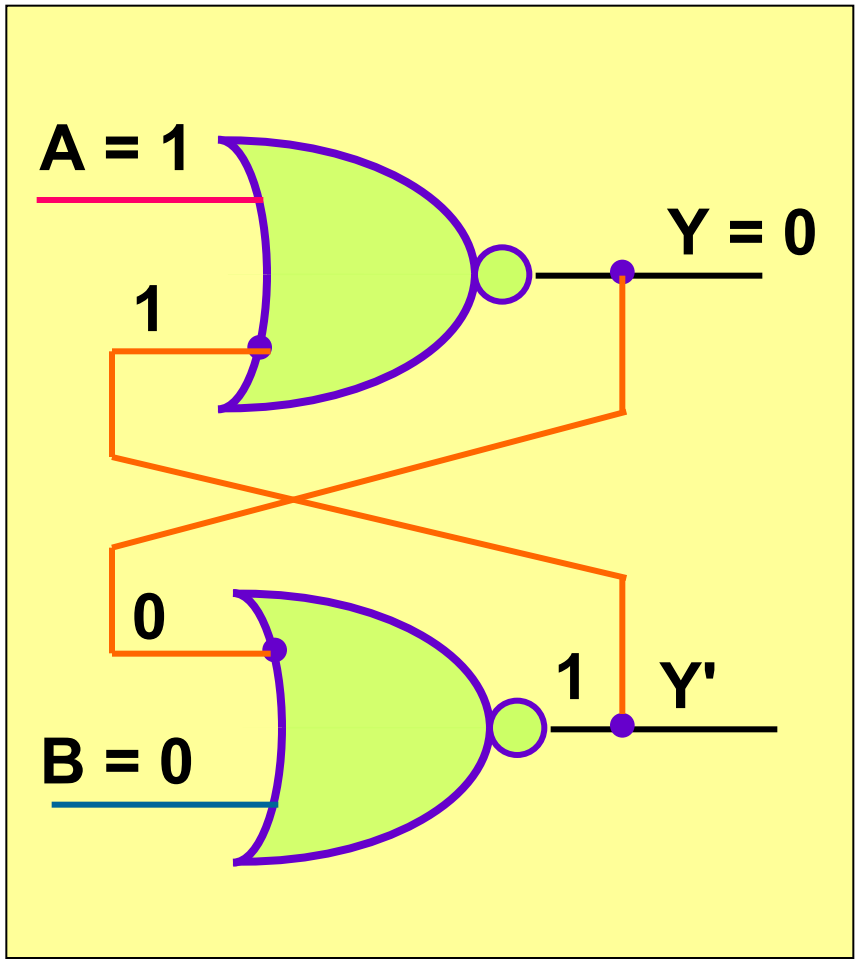
Elementos de Memoria

A= 0; B= 1; Y=1



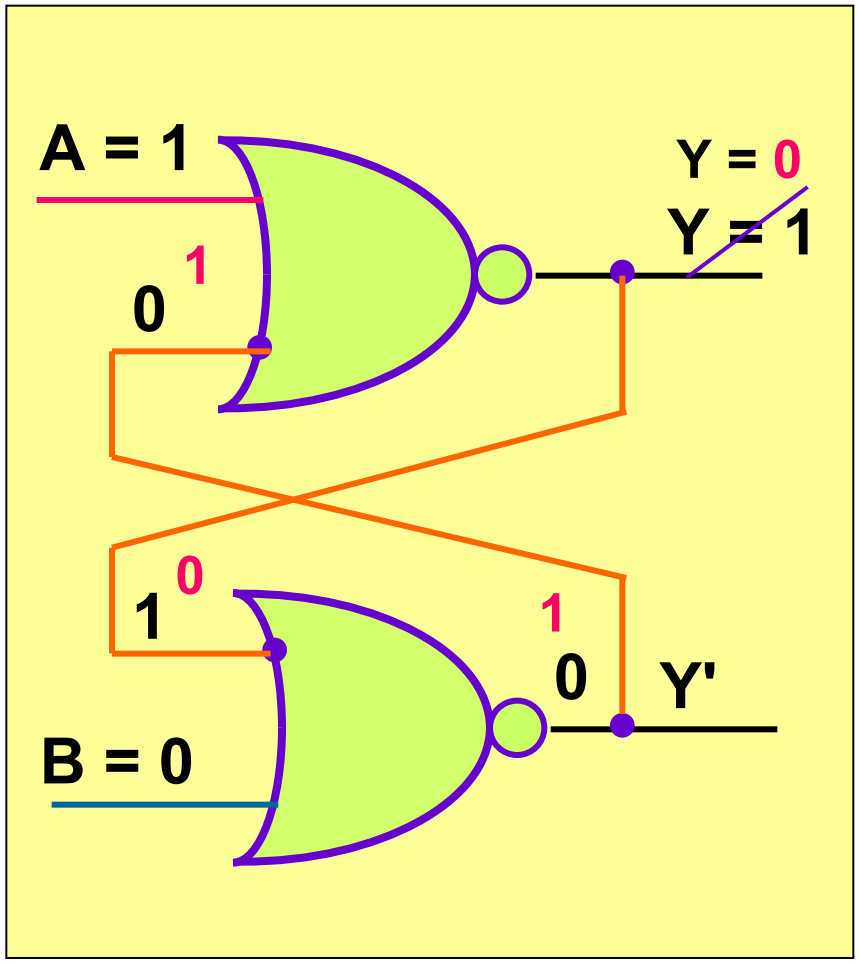
Elementos de Memoria

A= 1; B= 0; Y=0



Elementos de Memoria

A= 1; B= 0; Y=1



Tablas de Funcionamiento

Tabla Nº 1

#	A	B	Estado Actual (Y)	Próxima Salida (Y+1)	Y'
0	0	0	0	0	1
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	1
6	1	1	0	Estados Prohibidos	
7	1	1	1		

Tabla Nº 2

#	A	B	Estado Actual (Y)	Próxima Salida (Y+1)
2	0	1	0	1
3	0	1	1	1
1	0	0	1	1

Tabla Nº 3

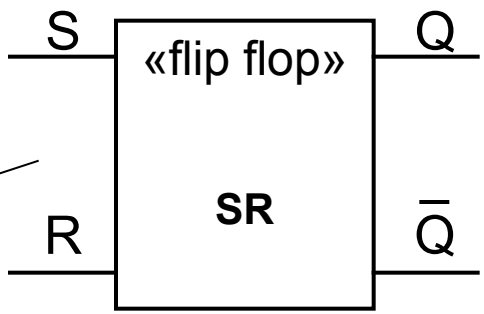
#	A	B	Estado Actual (Y)	Próxima Salida (Y+1)
5	1	0	1	0
4	1	0	0	0
0	0	0	0	0

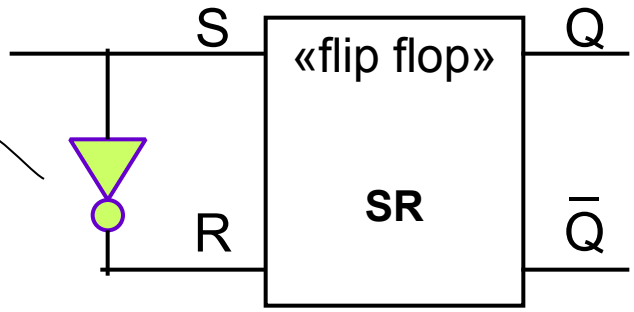
El comportamiento de este artefacto no es determinístico cuando ambas entradas son “1”

Dispositivos Biestables

- Para comprobar si este circuito se comporta de manera secuencial se toman las instancias de la Tabla Nº 1 en forma encadenada:
- Se puede ver (Tabla Nº 2) que la entrada B hace que la salida tome el valor 1, por lo tanto se la denomina ***Set***;
- Por otra parte (Tabla Nº 3) la entrada A obliga a que la salida tome el valor 0 y se la denomina ***Reset***.
- El arreglo de compuertas obtenido es un ***biestable***, es decir un artefacto que tiene dos estados estables.
- Cuando se alcanza un estado estable la salida se mantiene aunque la entrada que lo ha originado esté inactiva.
- Este circuito biestable o *flip-flop* recibe el nombre de **SetReset** o simplemente ***SR***.

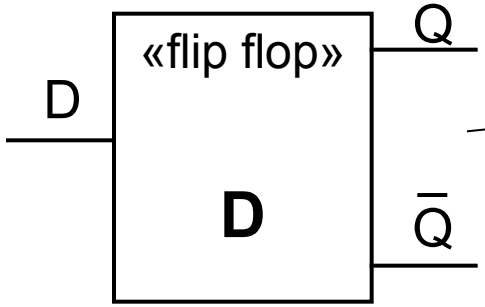
Biestables o ***flip-flops***





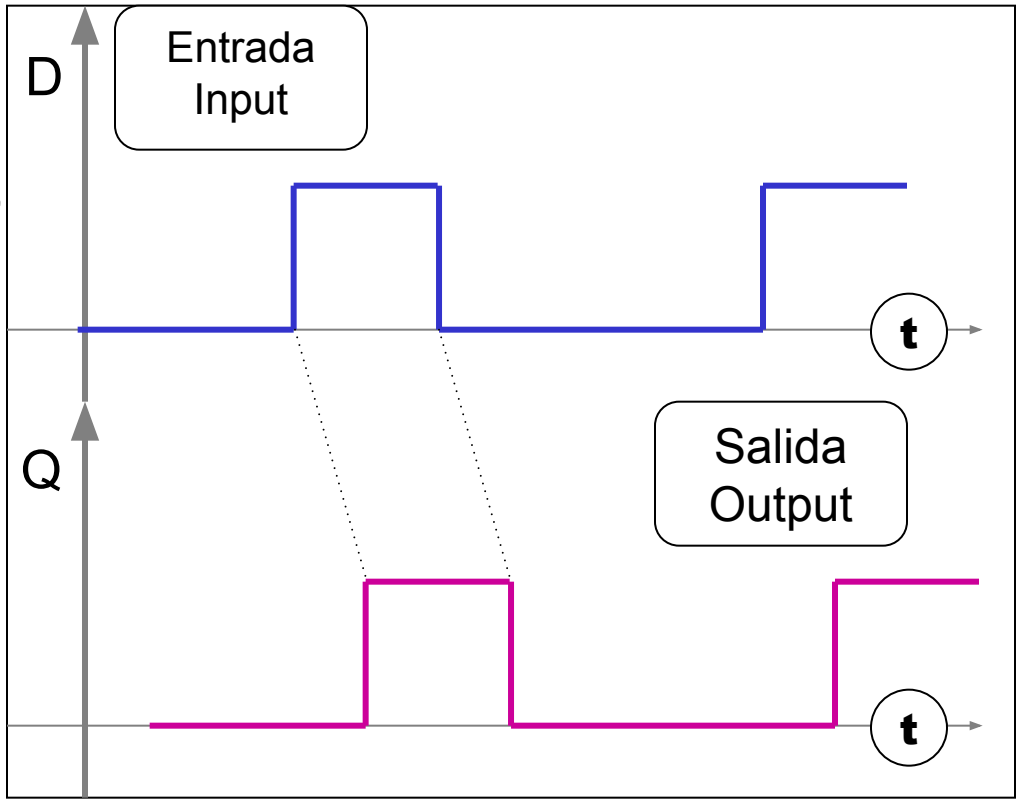
Para impedir que ambas entradas tomen el valor **1** en forma simultánea se agrega un inversor

El resultado es un flip flop D, cuya salida repite la entrada.



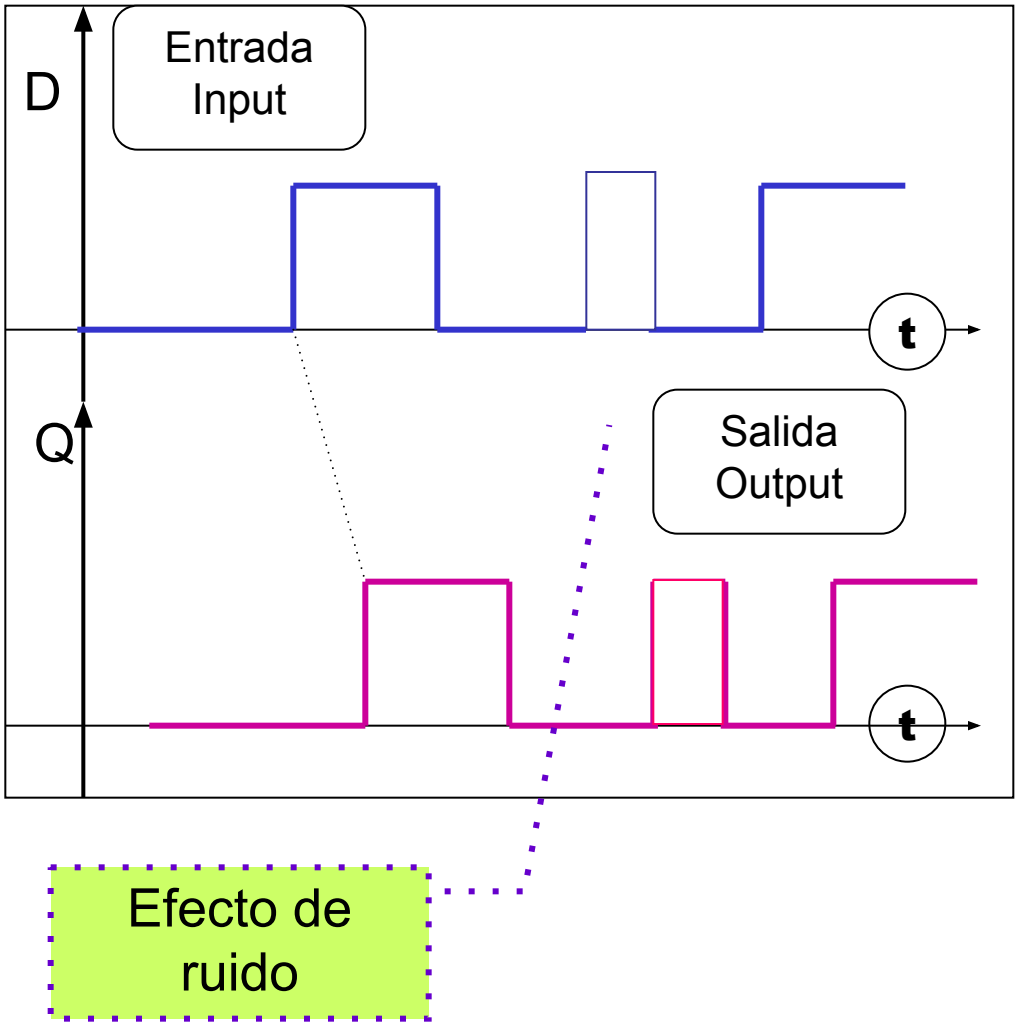
Diagramas Temporales

- Se puede realizar un diagrama temporal para presentar el comportamiento de la salida de un flip-flop en función de los cambios de sus entradas.
- Para el FF D se obtiene el gráfico de la figura adjunta

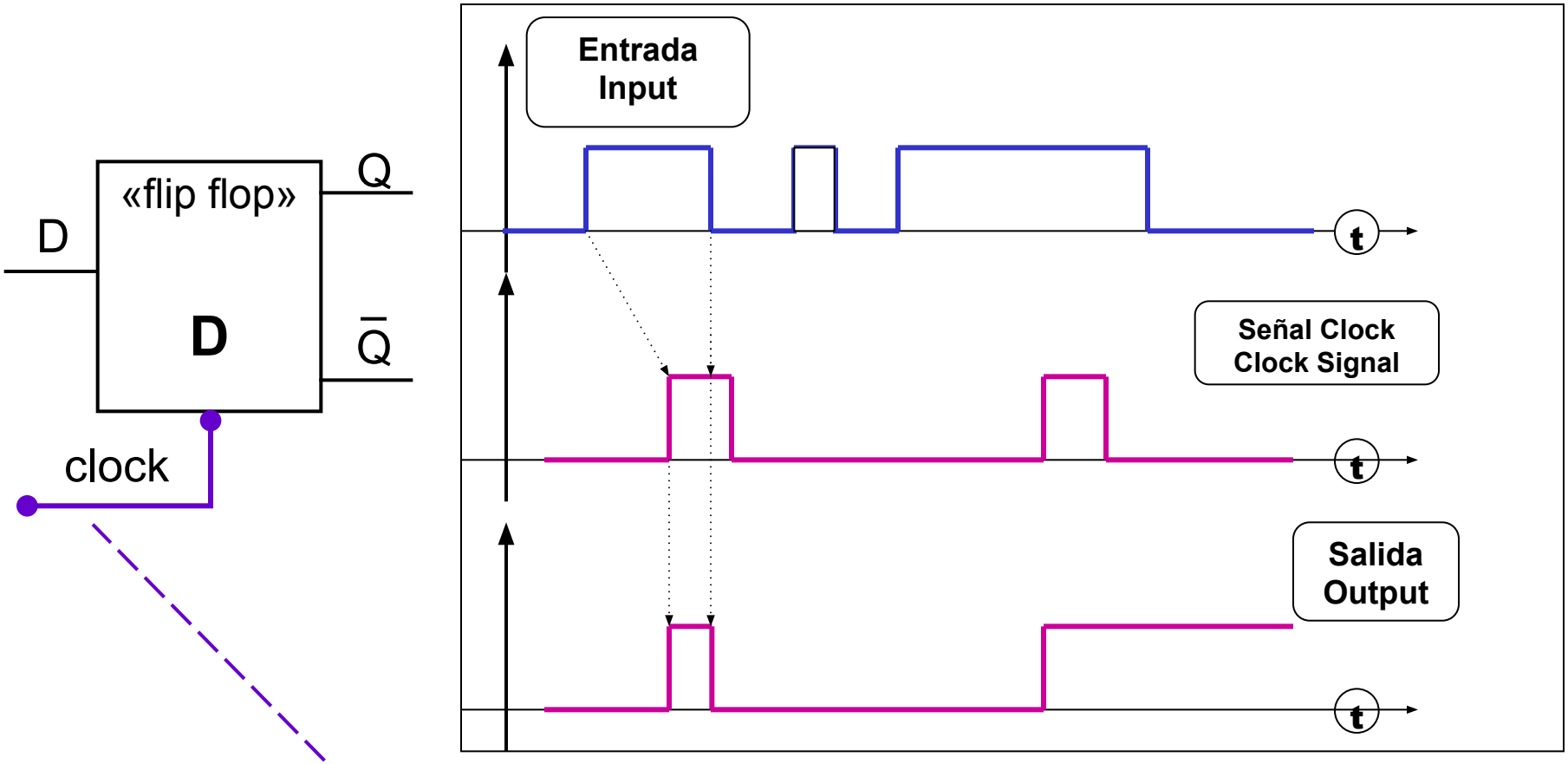


Sincronización

- En un flip-flop D convencional todos los cambios experimentados por la entrada se copian directamente en la salida.
- Por lo tanto se dice que tienen baja inmunidad al ruido.
- Para conseguir un comportamiento menos vulnerable al ruido se implementan dispositivos sincronizables.
- Los FF sincronizables poseen una entrada adicional de control o sincronismo (a veces llamada entrada de reloj).
- La entrada tiene efecto sobre la salida solamente cuando está presente la señal de sincronismo, de lo contrario los cambios quedan bloqueados



Sincronización



La sincronización puede realizarse por nivel (alto-bajo) o por flanco (ascendente-descendente)

Otros Biestables

- Biestable R-S sincrónico;
- Biestable J-K sincrónico;
- Biestable T sincrónico;
- Master-Slave

Registros y Operaciones con Registros

- Registros
- Registros contadores
- Registro contador progresivo de 8 eventos (una aplicación con biestables T); Contador regresivo de 8 eventos (con biestables T);
- Registros con facilidad de desplazamiento:
  - Desplazamientos lógicos;
  - Desplazamientos circulares;
  - Desplazamientos aritméticos;
  - Desplazamientos concatenados.

FIN