Unidad 3.1 – CIRCUITOS LÓGICOS COMBINACIONALES

DISEÑO DE CIRCUITOS COMBINACIONALES

Contenido

- Concepto y definición
- Logigramas
  - Técnicas de trazado
  - Logigrama desde la TV
  - Función desde logigrama
  - Compuertas tri-state
- Proceso de diseño
  - Definición de variables
  - Asignación de estados
  - Diseño directo
  - Diseño por T.V.
- Ejemplos

Referencias (Aula Virtual)

> Floyd T. (2006). FUNDAMENTOS DE SISTEMAS DIGITALES. Capítulo 3: Puertas lógicas; capítulo 5: Análisis de lógica combinacional.

> Ramos J. (2012). SISTEMAS DIGITALES. Capítulo 3: Lógica combinacional.

> Tocci R. (2007). SISTEMAS DIGITALES PRINCIPIOS Y APLICACIONES. Capítulo 4: Circuitos lógicos combinacionales.

> Mano M. (2003). DISEÑO DIGITAL. Capítulo 4: Lógica combinacional.

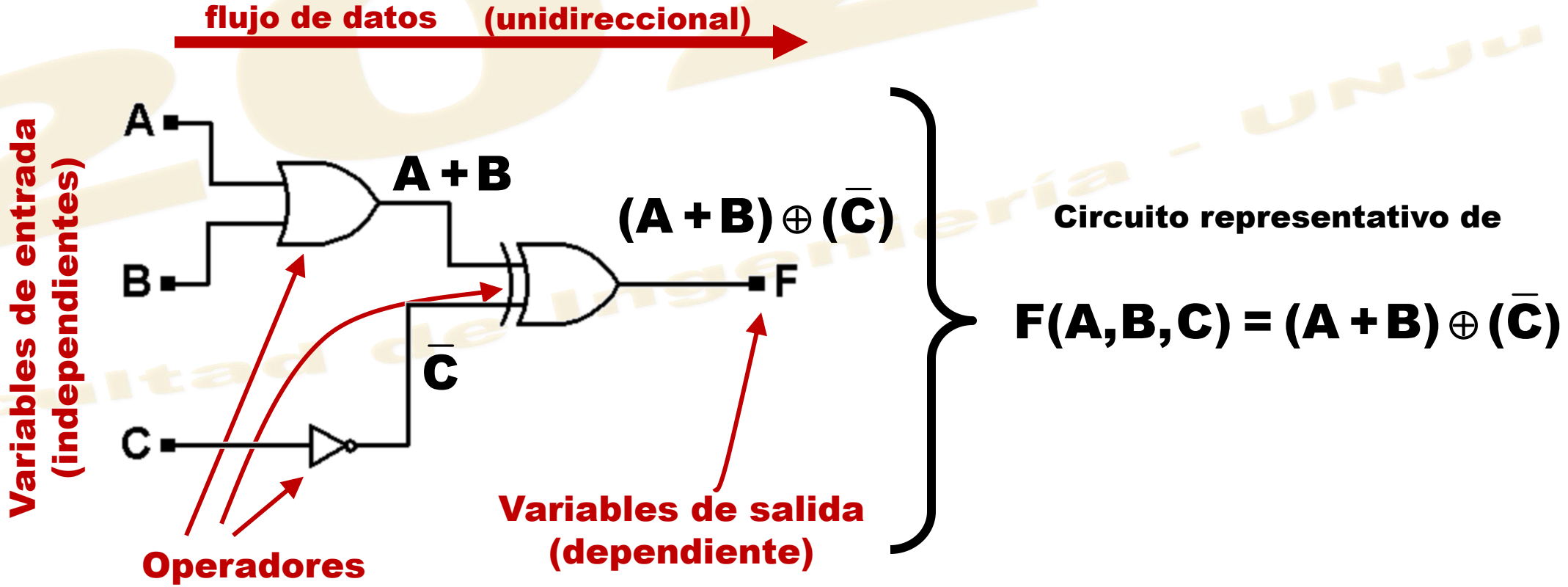
> Brown S. (2006). FUNDAMENTOS DE LÓGICA DIGITAL CON DISEÑO VHDL. Capítulo 2: Introducción a los circuitos lógicos.

Concepto

(Floyd, pg. 277)

Los circuitos lógicos, también conocidos como *logigramas*, constituyen otra forma equivalente de representación de una función lógica.

Se puede considerar como la representación gráfica de los operadores lógicos (compuertas), que relacionan a las variables lógicas que se representan a través de las líneas de interconexión.



LOGIGRAMAS

Es la representación de funciones lógicas a través de símbolos gráficos y líneas de conexión (circuitos lógicos).

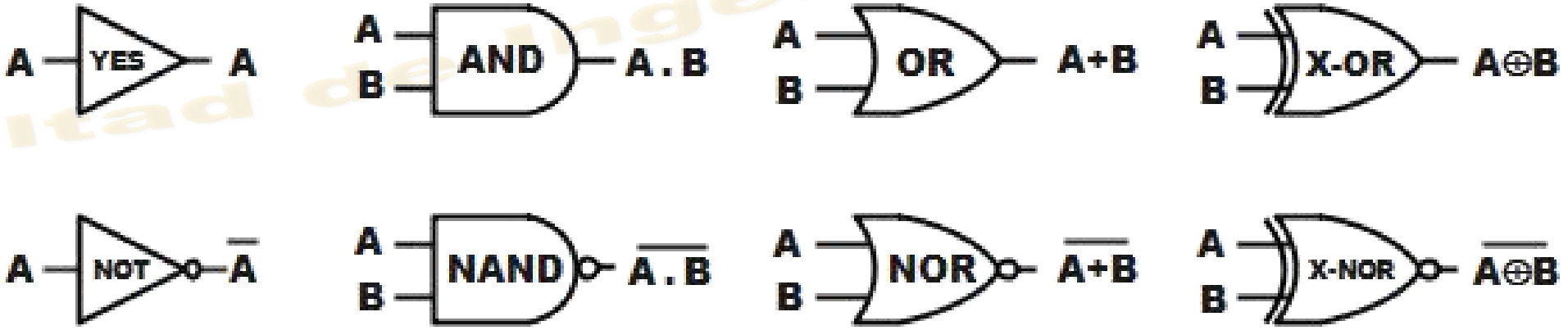
Los logigramas, diagramas lógicos o circuitos lógicos, permiten obtener una visión global de las variables lógicas y sus relaciones, dentro de una función lógica. Además, trazados sobre un software conveniente, se puede simular su funcionamiento.

Compuertas lógicas

(Floyd, pg. 124; Brown, pg. 25)

- Cada uno de los **operadores lógicos** tiene una definición y una representación gráfica llamada **compuerta lógica**.
- Las compuertas lógicas son estructuras **unidireccionales** que pueden relacionar dos o más variables (**excepto el negador - 1 variable**), a través de la operación que cada una representa.

Formato original



LOGIGRAMAS

Es la representación de funciones lógicas a través de símbolos gráficos y líneas de conexión (circuitos lógicos).

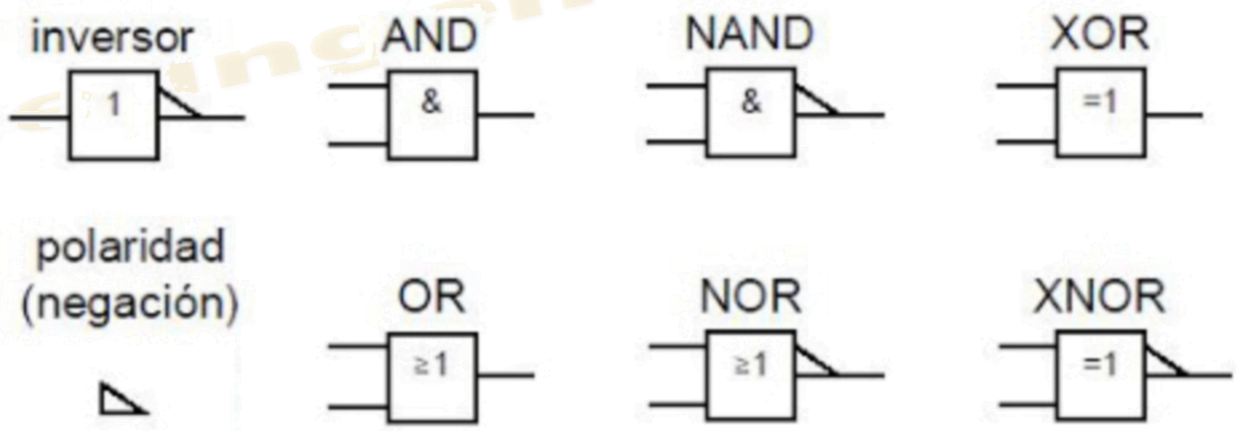
Los logigramas, diagramas lógicos o circuitos lógicos, permiten obtener una visión global de las variables lógicas y sus relaciones, dentro de una función lógica. Además, trazados sobre un software conveniente, se puede simular su funcionamiento.

Compuertas lógicas

(Floyd, pg. 124; Brown, pg. 25)

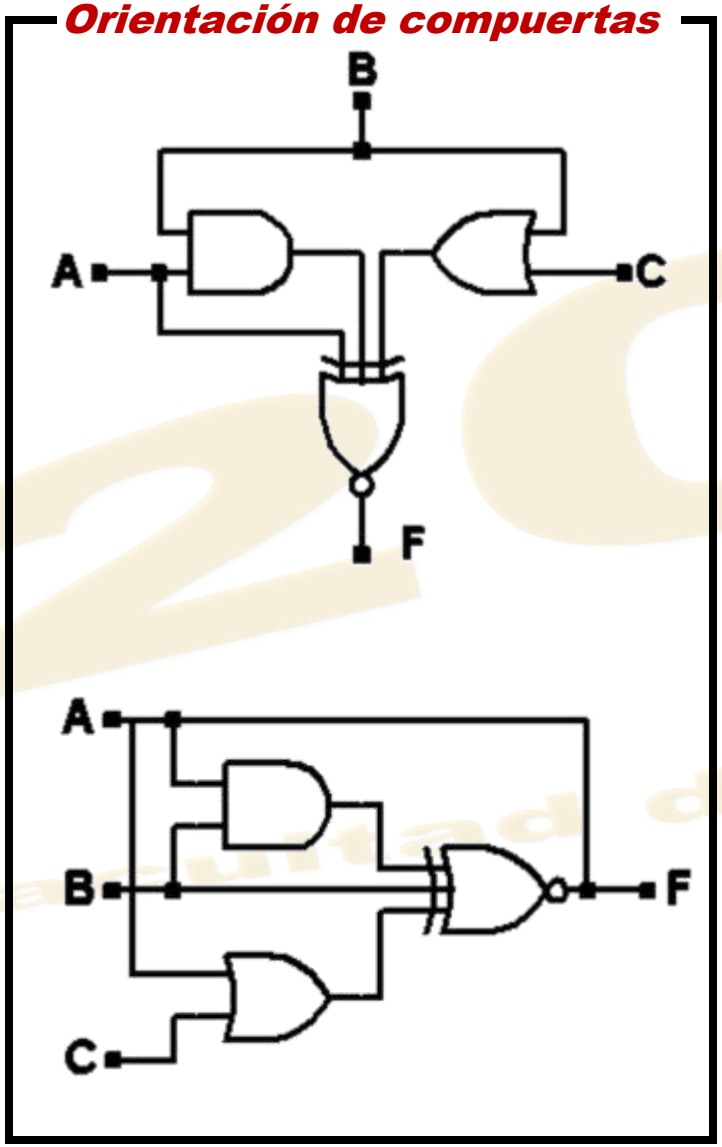
- Cada uno de los operadores lógicos tiene una definición y una representación gráfica llamada compuerta lógica.
- Las compuertas lógicas son estructuras unidireccionales que pueden relacionar dos o más variables (excepto el negador - 1 variable), a través de la operación que cada una representa.

Formato IEEE



Técnicas de trazado - compuertas

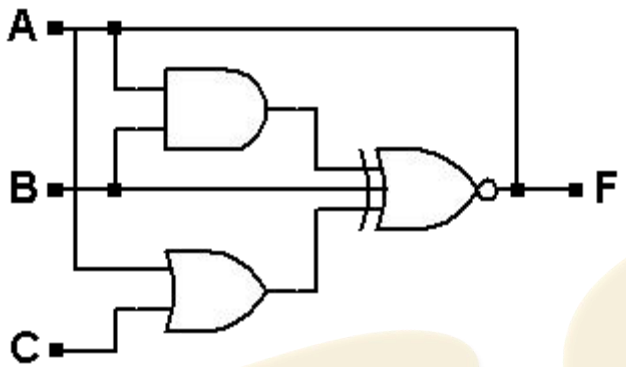
Orientación de compuertas



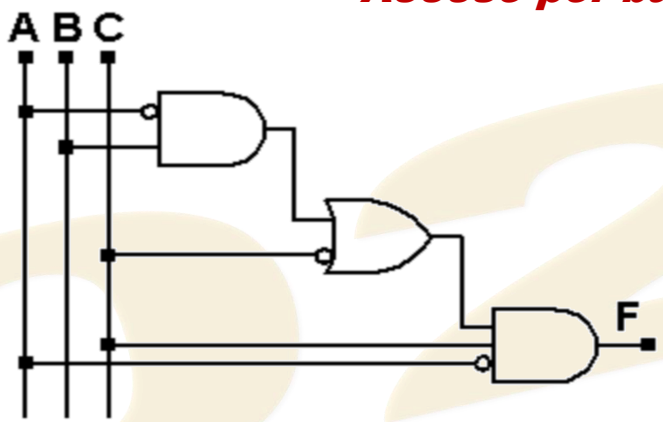
- Las compuertas se pueden disponer en **cualquier posición** y **orientación** dentro del logigrama.
- El circuito es completamente funcional y (eventualmente) correcto, pero una disposición desordenada o no convencional dificulta la interpretación y análisis del circuito.
- El **trazado estándar** es el más conveniente y usual: disposición horizontal, compuertas orientadas de izquierda a derecha, todas de igual tamaño, flujo de datos de izquierda a derecha.

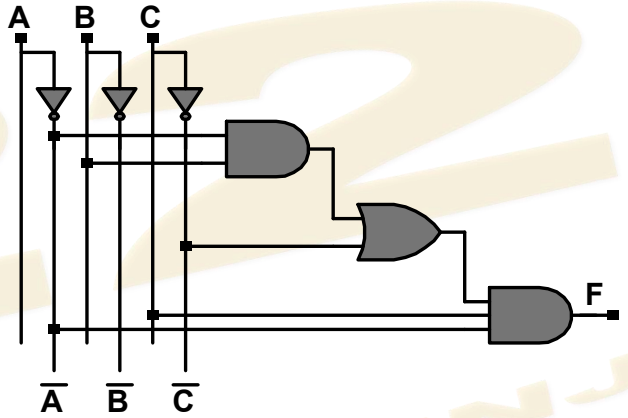
Técnicas de trazado - Ingreso de datos

Acceso puntual



Acceso por barras

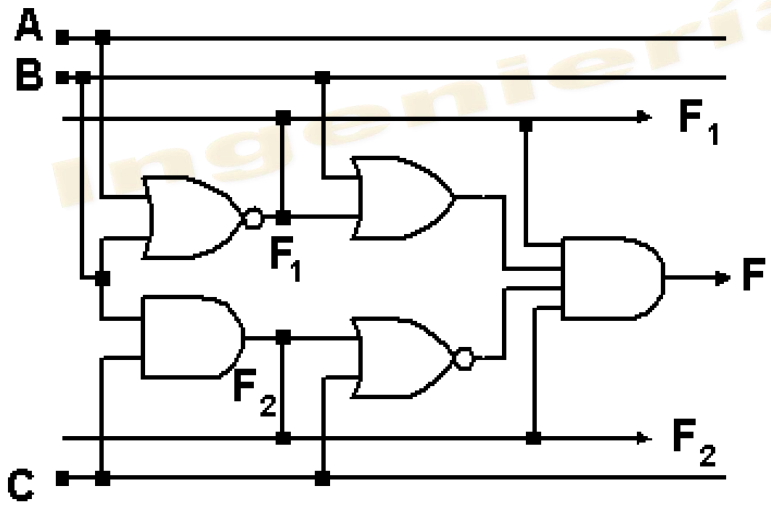


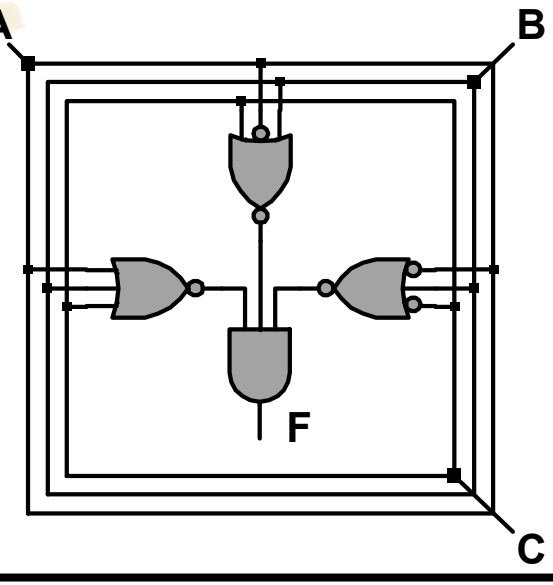


El ingreso de datos a los circuitos, suele ser complicado de trazar y de interpretar cuando hay muchas entradas o circuitos complejos.

Pueden utilizarse diversas disposiciones para ingresar los datos, según conveniencia y simplicidad.

Acceso por esquemas

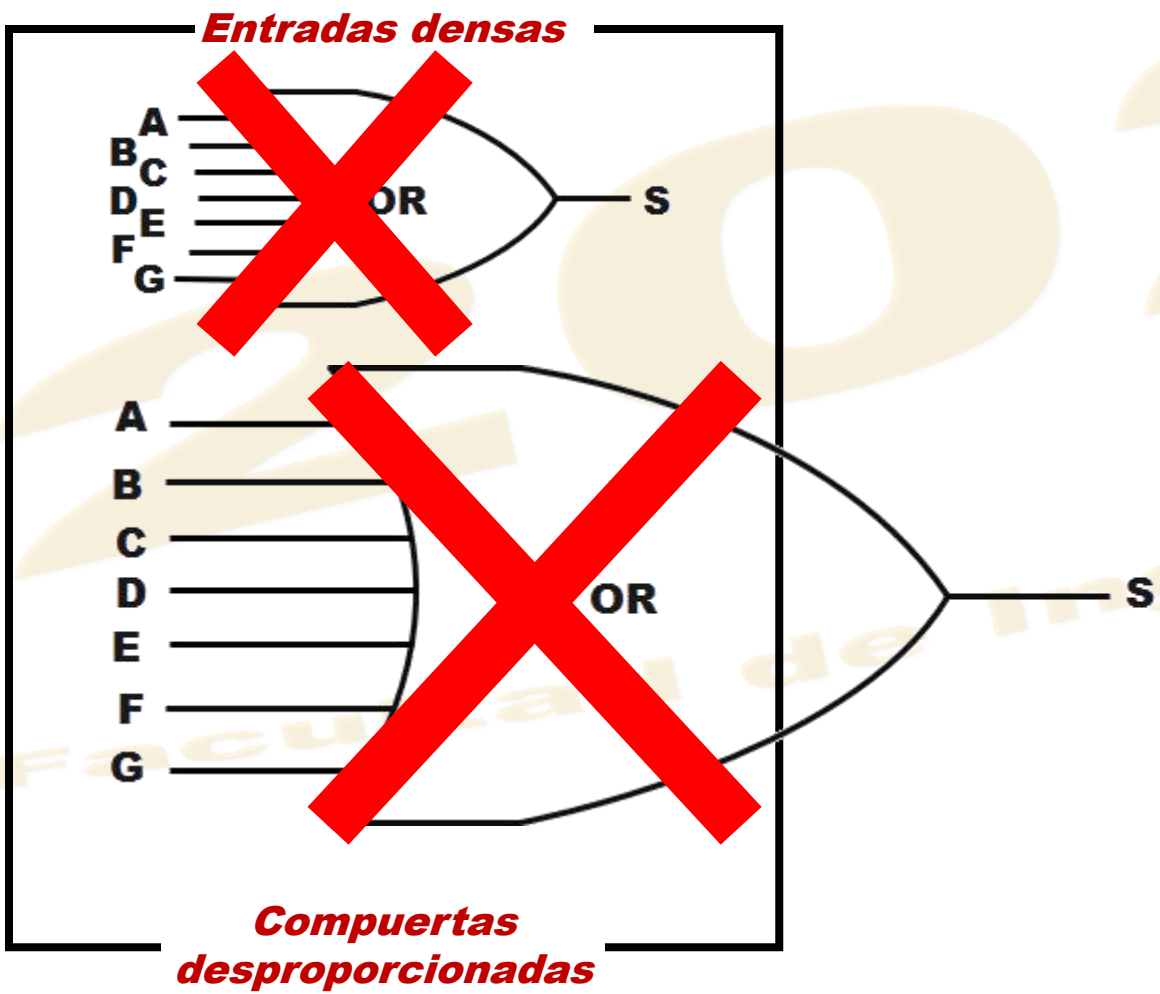


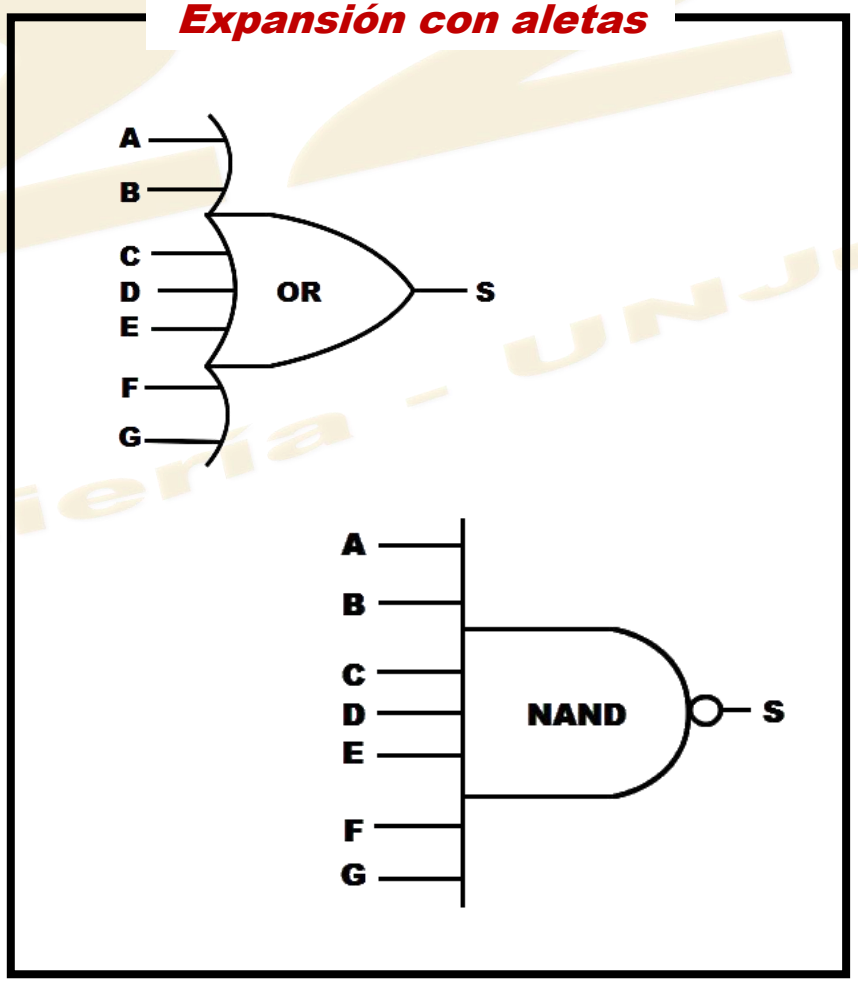


Técnicas de trazado - expansión

Se deben evitar las entradas demasiado densas o utilizar compuertas desproporcionadas.

Para ello se agregan aletas de expansión, que organizan las entradas en posiciones cómodas y accesibles.

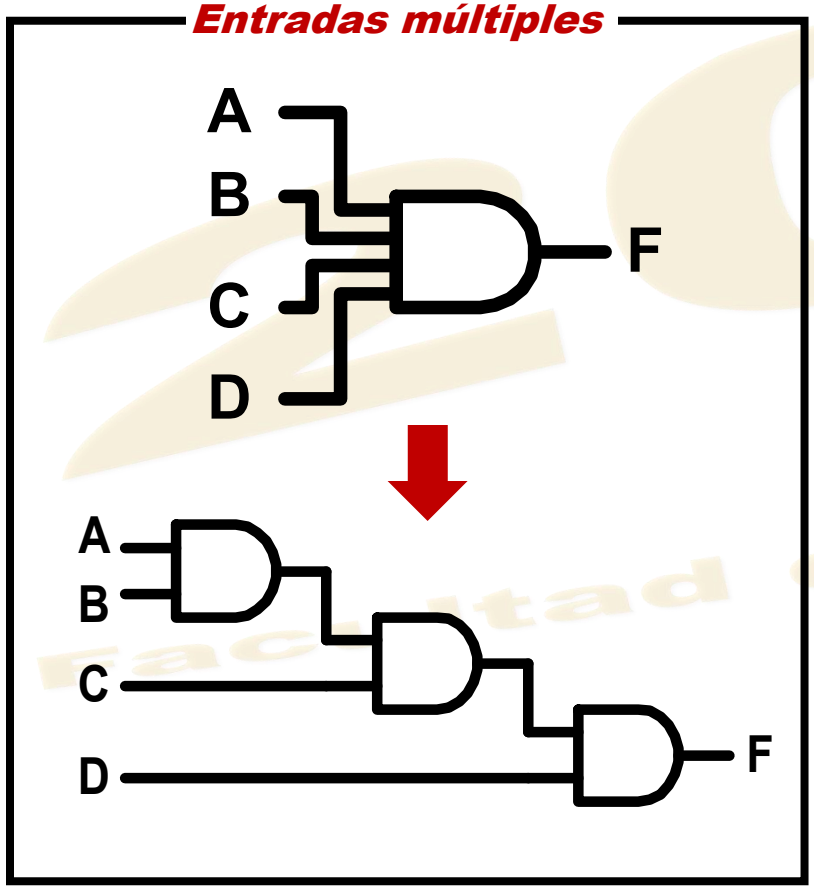




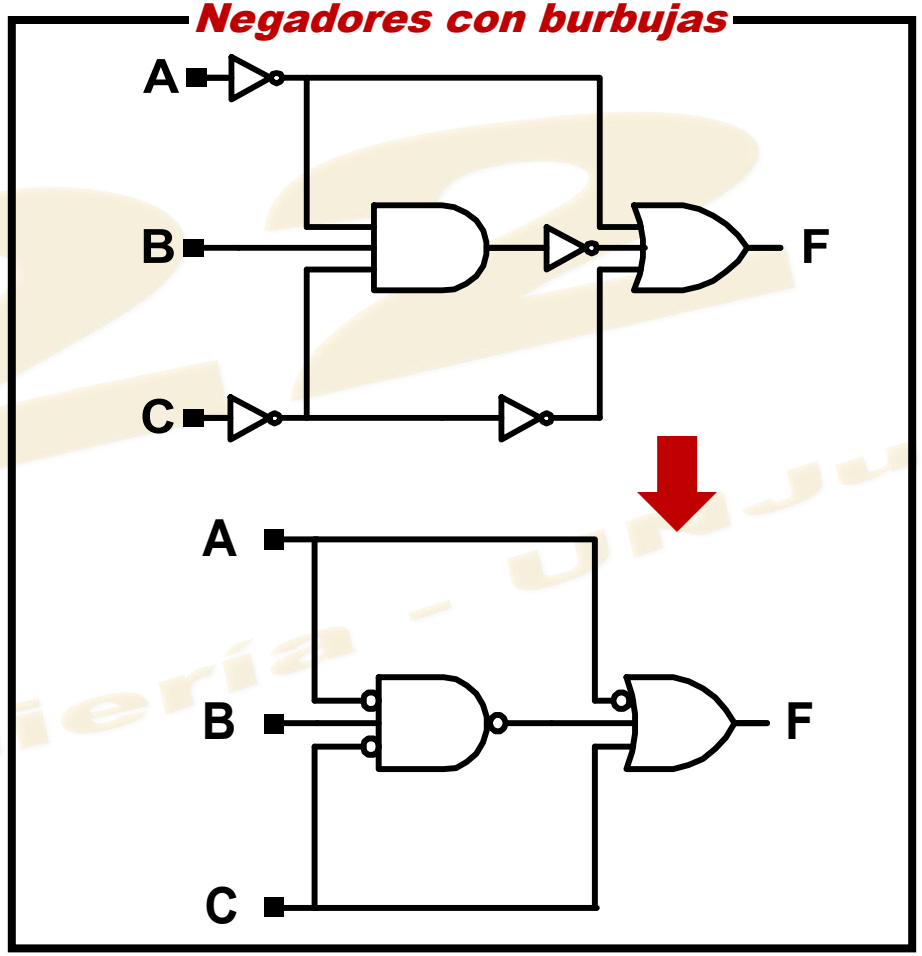
Técnicas de trazado

El uso de múltiples negadores puede entorpecer la interpretación del circuito. Es buena práctica reemplazar los negadores, o algunos, con su formato de burbuja.

Entradas múltiples

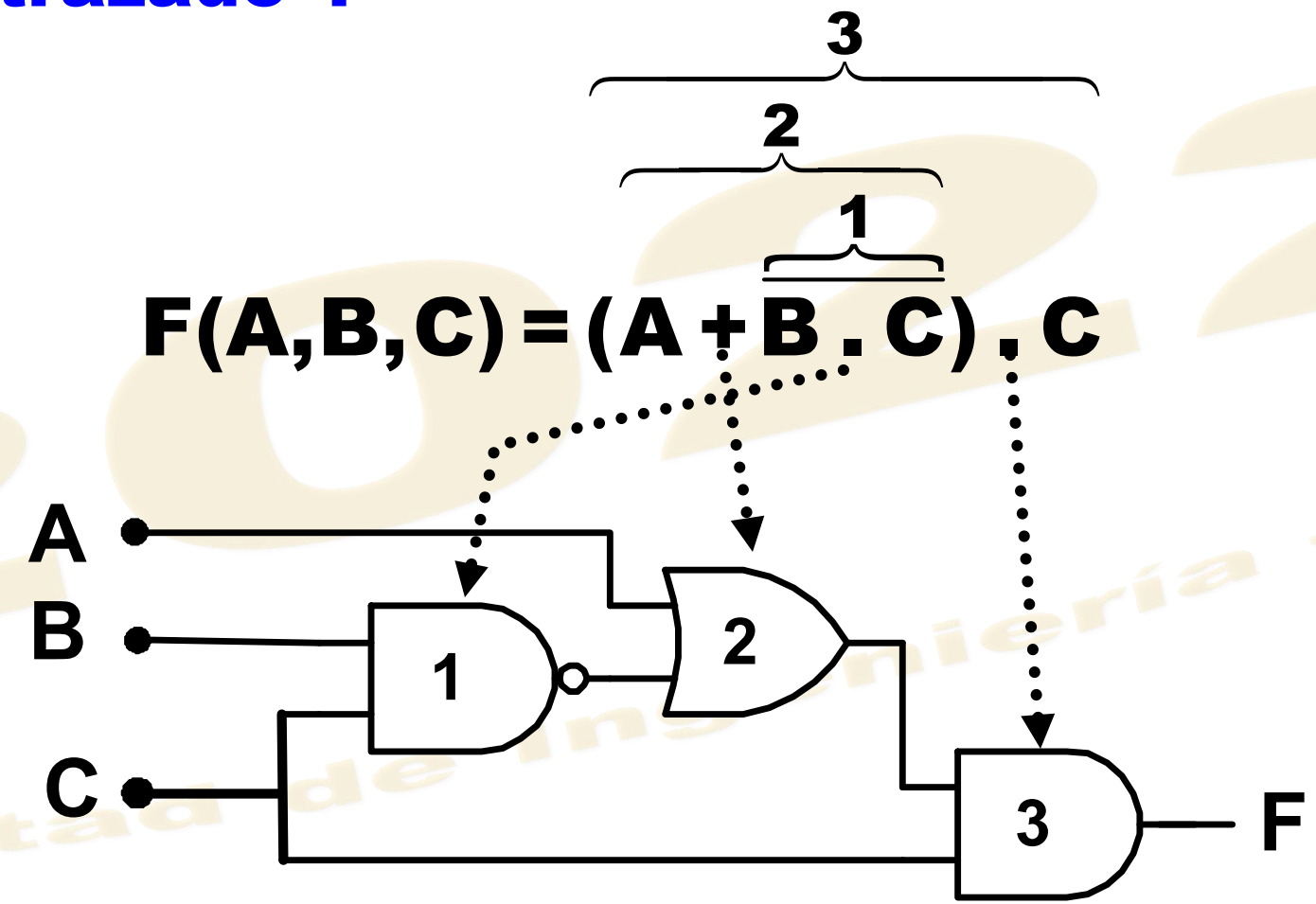


Negadores con burbujas



Cuando se requieren compuertas con varias entradas y no hay disponibles, se pueden asociar. Esto generalmente ocurre en la construcción física o simulación; en el dibujo siempre hay disponibilidad.

Ejemplo de trazado 1

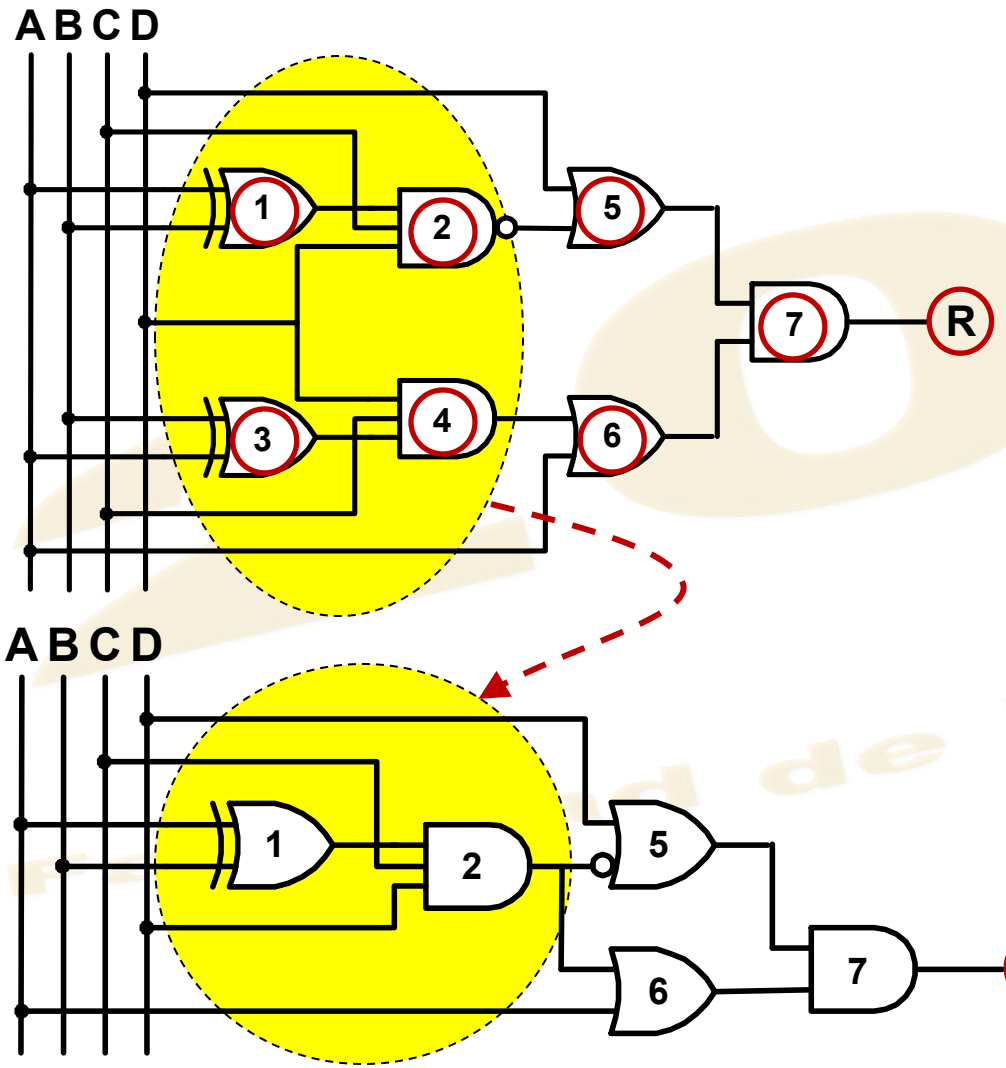


El logigrama debe respetar la jerarquía de los operadores igual que en la función.

Ejemplo de trazado 2

$$R(A,B,C,D) =$$

$$= \overline{(A \oplus B).C.D + D)} \,.\, ((A \oplus B).C.D + A)$$

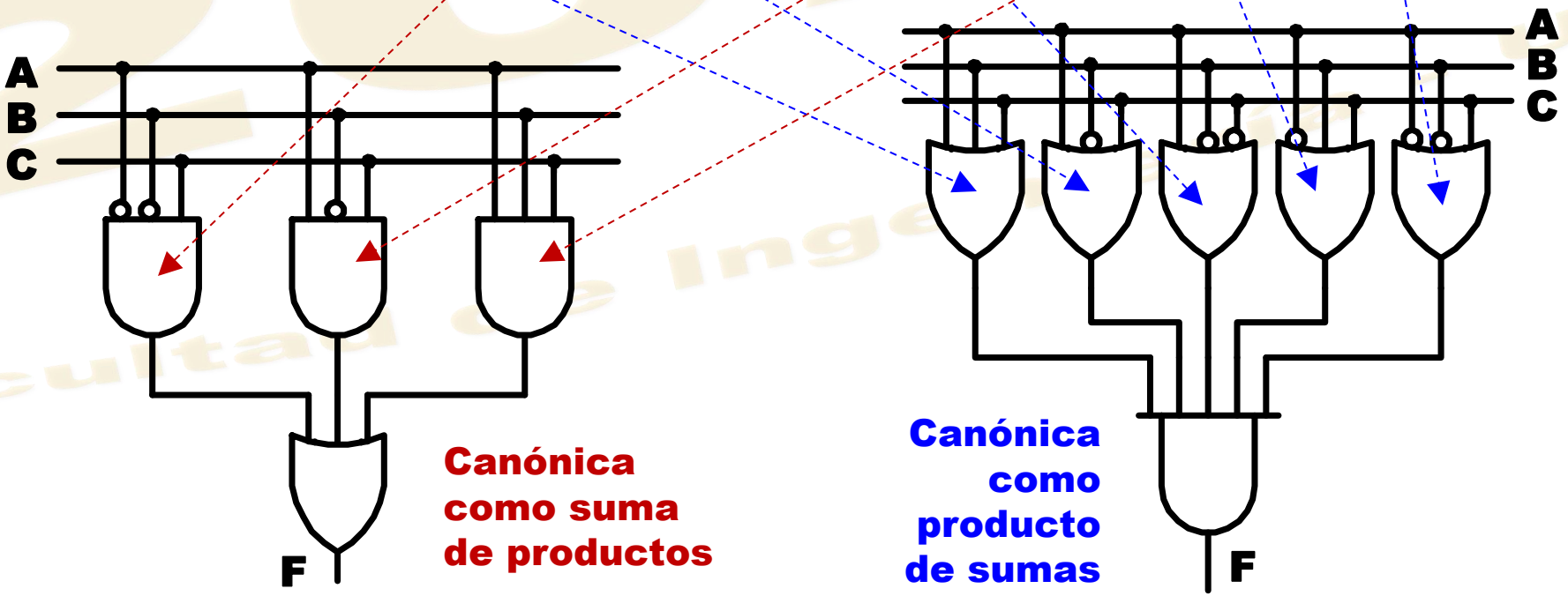


Segmentos equivalentes en la función lógica se pueden graficar unificados, utilizando una estructura común en el logigrama.

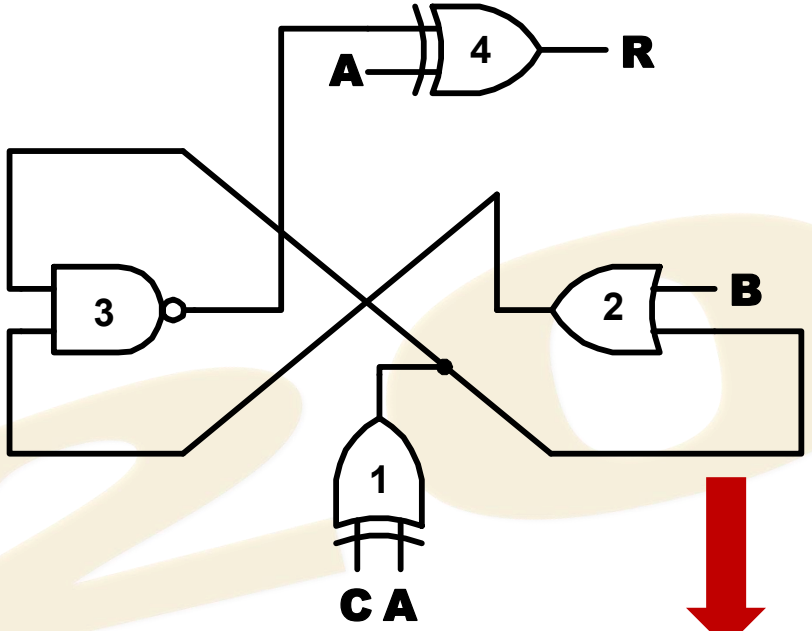
Logigramas desde la T.V.

(Floyd, pg. 278)

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



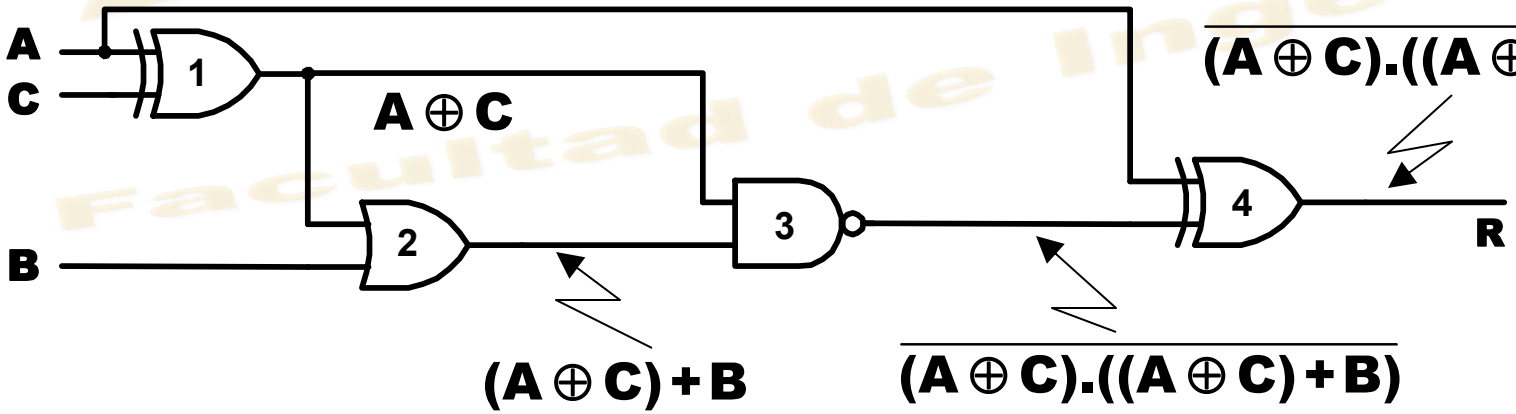
Función lógica desde el logigrama - entrada-salida



Se debe respetar la jerarquía de cálculo, establecida en el logigrama.

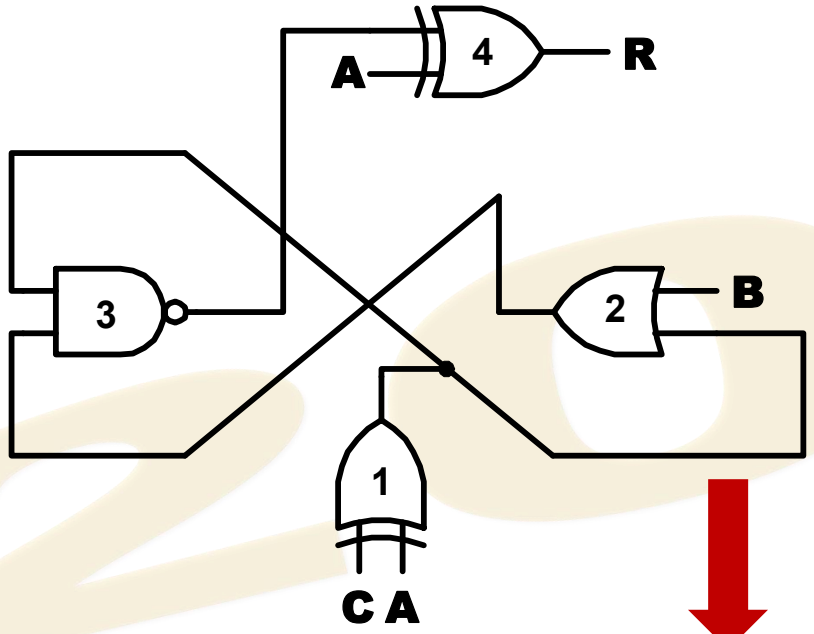
En este caso conviene redibujar el circuito si las interconexiones no están muy claras.

Método de entrada - salida $A \oplus C$ ...



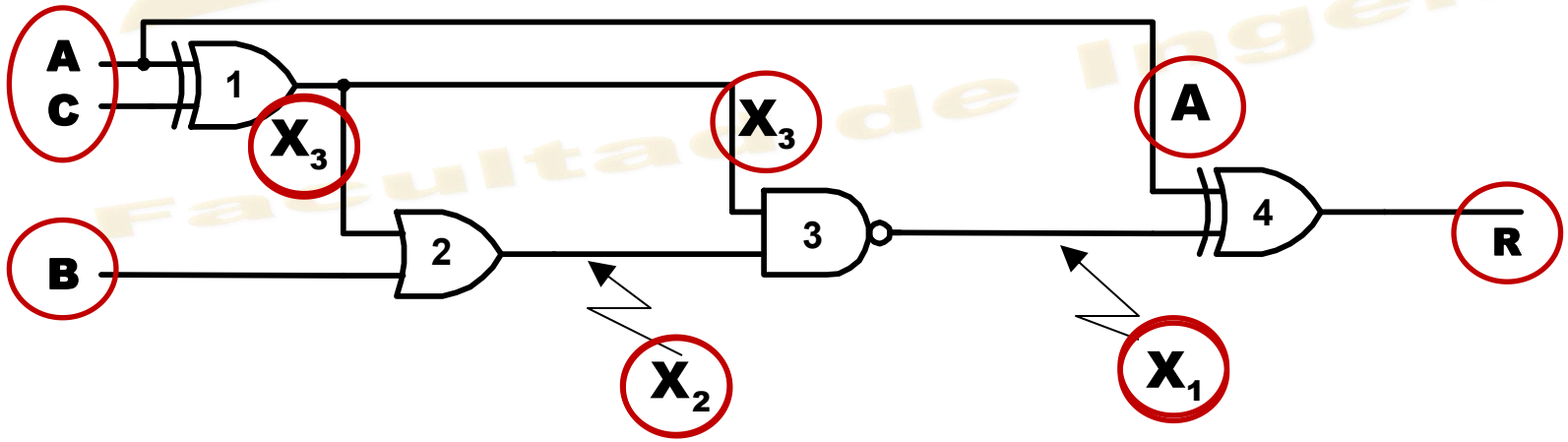
$$R = \overline{(A \oplus C).((A \oplus C) + B)} \oplus A$$

Función lógica desde el logigrama - salida-entrada



Se debe respetar la jerarquía de cálculo, establecida en el logigrama.

En este caso conviene redibujar el circuito si las interconexiones no están muy claras.



Método de salida - entrada

$$R = A \oplus X_1$$

$$X_1 = \overline{X_2 . X_3}$$

$$X_2 = B + X_3$$

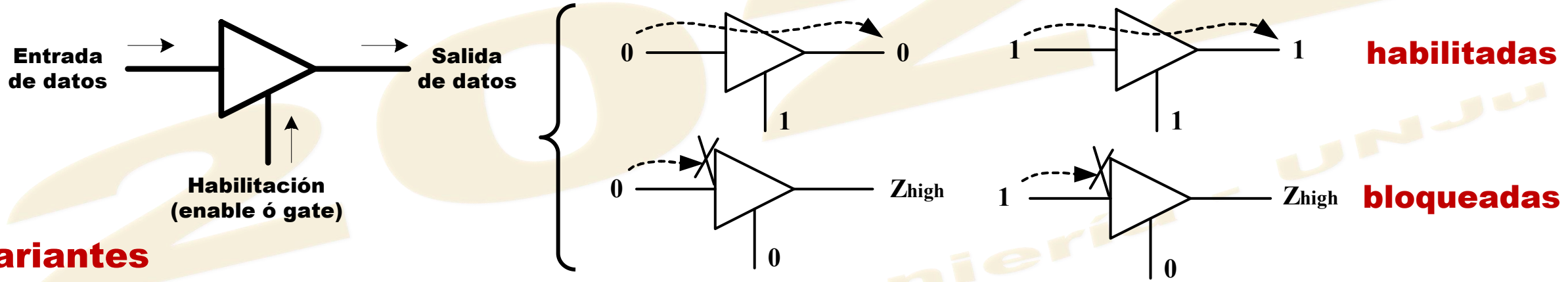
$$X_3 = A \oplus C$$

$$R = \overline{(A \oplus C).((A \oplus C) + B)} \oplus A$$

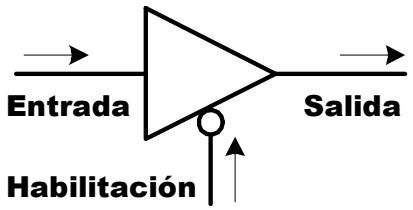
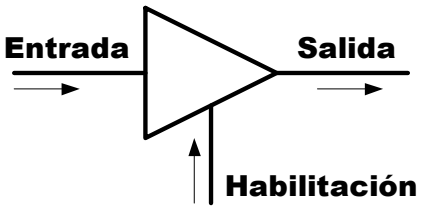
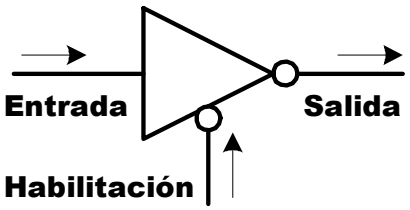
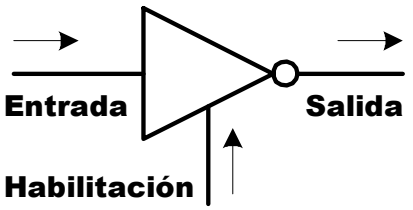
Compuertas tri-state

Mano, pg. 146

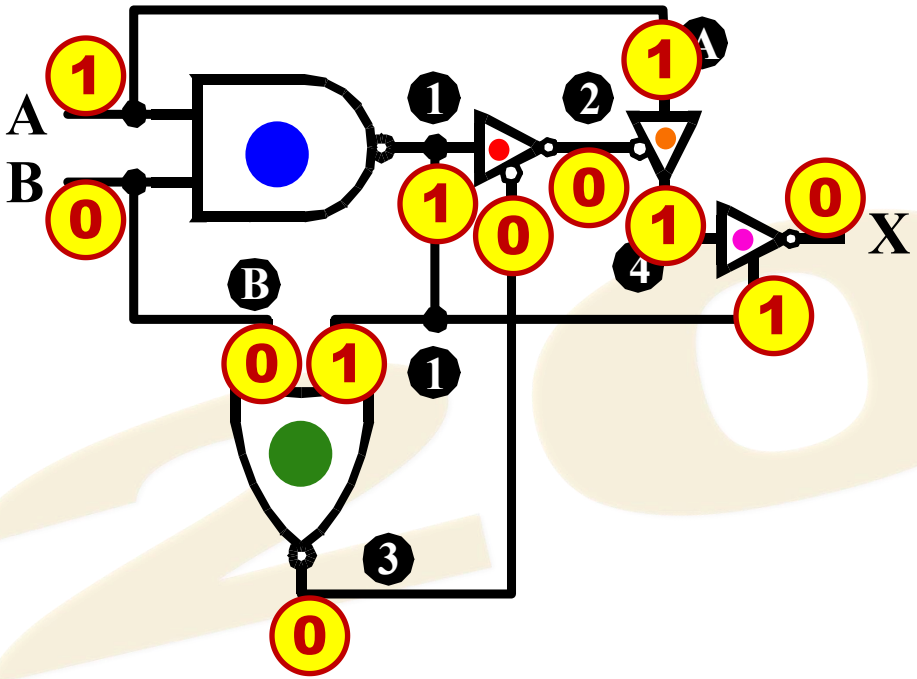
Las compuertas tri-state (tres estados) actúan como elementos de habilitación o interrupción del flujo de señales, ya sean señales lógicas (0 ó 1) o señales analógicas, utilizando una entrada especial de habilitación.



Variantes

			
Buffer directo Habilita con 0	**Buffer directo Habilita con 1**	**Buffer inversor Habilita con 0**	**Buffer inversor Habilita con 1**

Aplicación de compuertas tri-state



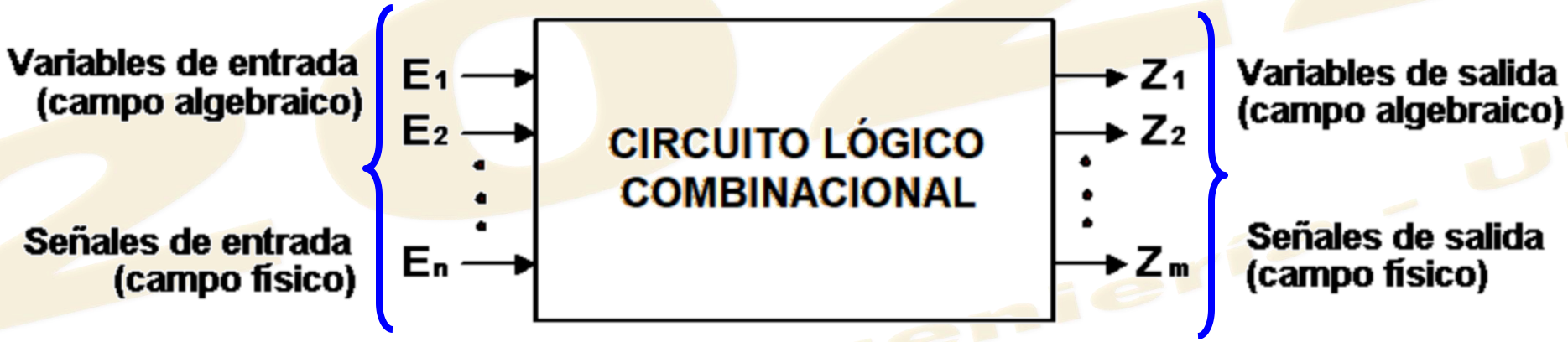
A	B	❶	❸	❷	❹	X
0	0	1	0	0	0	1
0	1	1	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	HZ	HZ

- HZ representa un estado de alta impedancia → tercer estado.
- El estado de alta impedancia se puede considerar como una desconexión virtual (no es 0 ni 1).
- Cuando la compuerta tri-tate se habilita, lo que está en la entrada pasa a la salida, directo o negado, según la configuración de la compuerta.

CIRCUITOS LÓGICOS COMBINACIONALES

Definición

Son aquellas configuraciones de *n* entradas y *m* salidas, tal que su salida o salidas dependen exclusivamente del estado *actual* de sus variables de entrada, independientemente del factor *tiempo*.



Todo circuito lógico combinacional representa a una determinada función lógica que no contiene componentes de realimentación.

El flujo de señales es unidireccional, de la entrada hacia la salida.

CIRCUITOS LÓGICOS COMBINACIONALES

Innumerables procesos físicos pueden modelarse con circuitos lógicos combinacionales, con la condición de que no se requiera almacenamiento ni realimentación de datos.

Procesos de diseño

(Ramos, pg. 45; Tocci, pg. 127)

- Comprensión del problema.
- Definición de variables binarias (entrada y salida).
- Asignación de estados.
- Síntesis de la función lógica
  - Directa
  - Por Tabla de Verdad
- Trazado del logigrama.
- Simulación y prueba.
- Implementación física.

Definición de variables

- Las variables a utilizar son discretas y binarias (variables lógicas).
- Deben ser representativas de los eventos a incorporar en el sistema (tanto de entrada como de salida).
- Si el evento a incorporar es binario, se representa con una variable; por ej. un interruptor (abierto-cerrado).
- Si el evento a incorporar tiene una cantidad de estados finita, se deben incorporar tantas variables de tal forma que:

  $$\text{Cantidad de estados} \leq 2^{\text{cantidad de variables}}$$

  Por ej. días de la semana (7) requiere 3 variables (2^3); los dígitos numéricos (10) requieren de 4 variables (2^4).
- Si el proceso a modelar es continuo o con una cantidad de estados muy grande se debe reducir/discretizar, por ejemplo separando en rangos o grupos.

Asignación de estados binarios

- Los modelos lógicos combinacionales se configuran con una o más funciones lógicas; y las variables lógicas representan a los eventos que tendrán 2 estados.
- Como todo el sistema/modelo trabaja en forma algebraica, cada estado de cada evento (entrada y salida) se debe asociar (codificar) a un bit, en principio en forma arbitraria.
- Por ejemplo, para el evento Interruptor, cuyos estados podrían ser cerrado-abierto, se puede asignar:

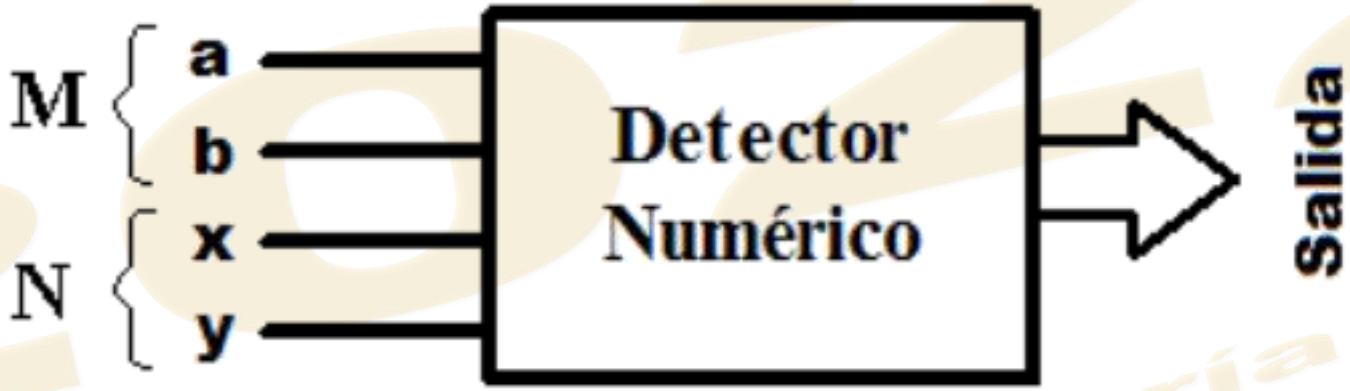
  Interruptor → cerrado = 1 || abierto = 0
- Para eventos multi-estados, se utilizan más de una variable y los estados se asocian a combinaciones de bits.
  Por ej. días de la semana ⇒ 3 variables V1, V2 y V3, se puede asignar:

Día	V1	V2	V3	Día	V1	V2	V3	Día	V1	V2	V3
Lunes	0	0	0	Martes	0	0	1	Miércoles	0	1	0
Jueves	0	1	1	Viernes	1	0	0	Sábado	1	0	1
Domingo	1	1	0					No usado	1	1	1

DISEÑO COMBINACIONAL

Ejemplo: Asignación de **estados de entrada** para detector de números M y N de 2 bits, con las siguientes condiciones:

a) M > N **b)** M divisible por N **c)** M y N primos



a, b, x, y, serían las variables lógicas que conforman a los números M y N y cada una representa a un bit (0 ó 1):

$$M = a_2 \cdot 10_2 + b_2 \qquad N = x_2 \cdot 10_2 + y_2$$

En este caso, la asignación de estados binarios es muy simple, porque ya está implícita. Se puede cambiar, pero no sería conveniente.

Asignación de estados binarios - salida

- La salida del sistema/modelo es también binaria numérica (resultado de una función lógica), por lo tanto se debe decodificar para que sea asociada a una respuesta interpretable por el usuario.
- Si la respuesta corresponde a un evento de dos estados, basta con una variable. Por ej. salida = luz; estados:

  0 = apagada || 1 = prendida.
- Para más de dos estados en la respuesta, se deben disponer más de una variable de salida.
- Se pueden utilizar diferentes variantes, eventos A, B y C:

salida en 2 variables S1 y S2

$$S1,S2 = \begin{cases} 00 \rightarrow \text{A verdad, resto falso} \\ 01 \rightarrow \text{B verdad, resto falso} \\ 10 \rightarrow \text{C verdad, resto falso} \\ 11 \rightarrow \text{no usado} \end{cases}$$

salida en 3 variables S1, S2 y S3

$$S1 = \begin{cases} 0 \rightarrow \text{A falso} \\ 1 \rightarrow \text{A verdadero} \end{cases}$$

$$S2 = \begin{cases} 0 \rightarrow \text{B falso} \\ 1 \rightarrow \text{B verdadero} \end{cases}$$

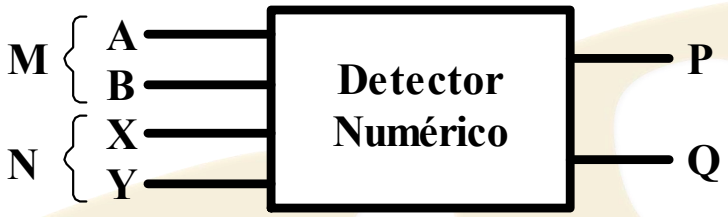
$$S3 = \begin{cases} 0 \rightarrow \text{C falso} \\ 1 \rightarrow \text{C verdadero} \end{cases}$$

DISEÑO COMBINACIONAL

Ejemplo: Asignación de **estados de salida** para detector de números M y N de 2 bits, con las siguientes condiciones:

a) M > N **b)** M divisible por N **c)** M y N primos

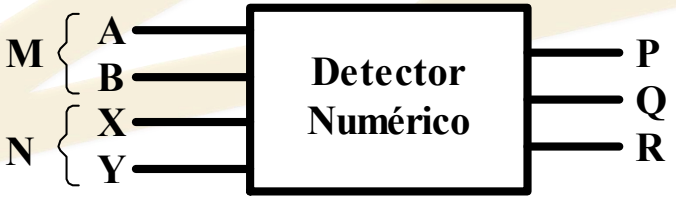
Esquema de salida 1



$$P, Q = \begin{cases} 0,0 \Rightarrow \text{evento a) verdadero, los otros falsos.} \\ 0,1 \Rightarrow \text{evento b) verdadero, los otros falsos.} \\ 1,0 \Rightarrow \text{evento c) verdadero, los otros falsos.} \\ 1,1 \text{ no utilizado.} \end{cases}$$

No es muy eficiente. Sólo puede aplicarse cuando los eventos son mutuamente excluyentes.
No sirve cuando hay coincidencia de eventos verdaderos.

Esquema de salida 2



$$P = \begin{cases} 1 \Rightarrow \text{evento a) verdadero.} \\ 0 \Rightarrow \text{evento a) falso.} \end{cases}$$

$$Q = \begin{cases} 1 \Rightarrow \text{evento b) verdadero.} \\ 0 \Rightarrow \text{evento b) falso.} \end{cases}$$

$$R = \begin{cases} 1 \Rightarrow \text{evento c) verdadero.} \\ 0 \Rightarrow \text{evento c) falso.} \end{cases}$$

Es más eficiente, aunque incorpora más variables de salida.

PROCESO DE DISEÑO – 1. DIRECTO

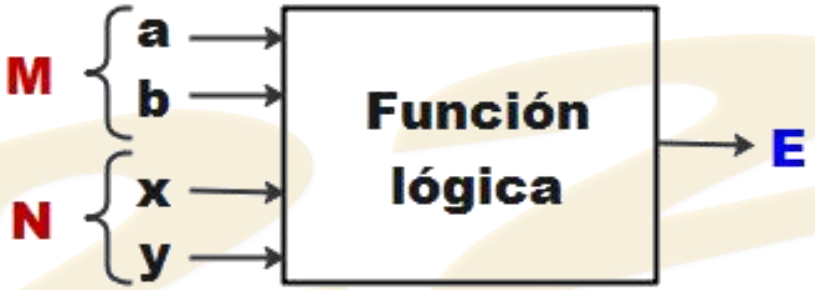
- En problemas complejos, reducir a situaciones más simples.
- Tanto los eventos de entrada, como los objetivos, deben ser binarios o discretizables.
- Se asignan valores lógicos a las variables, en forma arbitraria o con algún criterio.
- Eventos de producción simultáneos se tratarán con compuertas AND; caso contrario OR.
- Se pueden usar operadores combinados, siempre que su acción esté correctamente interpretada.
- Cuando existan restricciones, se pueden tratar con estructuras lógicas separadas que luego se integran.
- La función o funciones lógicas obtenidas, deben probarse con todos los datos.
- En ciertos casos, se puede diseñar directamente el circuito, obtener su función y comprobar.

DISEÑO COMBINACIONAL - DIRECTO

Ejemplo: Diseño de un detector de igualdad para 2 números de 2 bits c/u.

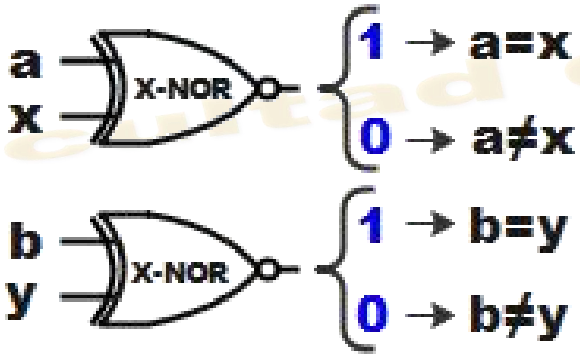
Variables entrada:
1º num. $M(a_2,b_2) = a_2*10_2 + b_2$
2º num. $N(x_2,y_2) = x_2*10_2 + y_2$

Variable de salida: E (binaria)

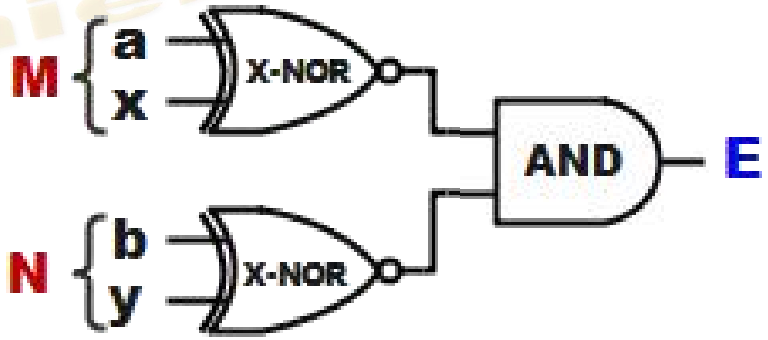


Asignación de estados:
[a, b, x, y] ← (0 ó 1) según instancia
E = 1 ⇒ iguales; E = 0 ⇒ distintos

Estrategia de diseño: comparar los bits de posiciones homólogas. Utilizar operador de igualdad (X-NOR):



Concatenar comparaciones: Se puede pensar en el mismo operador XNOR (error si ambas comparaciones son diferentes)
Conviene concatenar con AND.



$$E = \overline{(a \oplus x)} \,.\, \overline{(b \oplus y)}$$

PROCESO DE DISEÑO 2 – CON T.V.

(Mano, pg. 115)

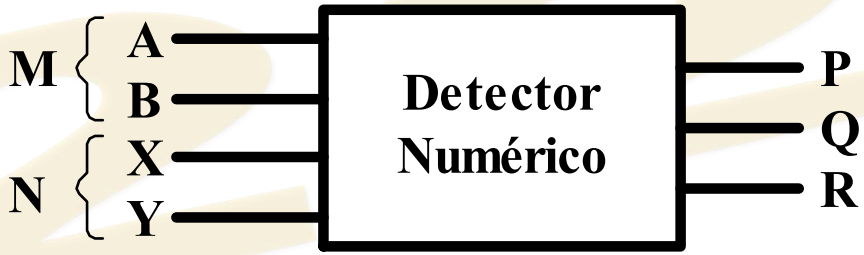
- **Comprender** el problema y analizar la situación.
- Tanto los eventos de entrada, como los objetivos, deben ser **binarios o discretizables**.
- Se asignan **valores lógicos a las variables de entrada y salida**, en forma arbitraria o con algún criterio.
- Construir la T.V. con **todas las combinaciones** de las variables de entrada.
- **IMPONER los valores lógicos** a las variables de salida, de acuerdo a los valores que toman las variables de entrada, considerando restricciones si las hubiere.
- **Sintetizar la función** desde la T.V., según necesidades: general, canónica, minimizada, etc.
- **Simular, comprobar** y eventualmente reproducir en hardware.

DISEÑO COMBINACIONAL - CON T.V.

Ejemplo: Detector de números M y N de 2 bits

a) P→M > N **b) Q→M divisible N** **c) R→M y N primos**

M		N		Salidas		
A	B	X	Y	P	Q	R
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	0	1	0
		1	1	0	1	0
0	1	0	0	1	0	0
		0	1	0	1	1
		1	0	0	0	1
		1	1	0	0	1
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	0	1	1
		1	1	0	0	1
1	1	0	0	1	0	0
		0	1	1	1	1
		1	0	1	0	1
		1	1	0	1	1



$$P = \begin{cases} 1 \Rightarrow \text{evento a) verdadero.} \\ 0 \Rightarrow \text{evento a) falso.} \end{cases}$$

$$Q = \begin{cases} 1 \Rightarrow \text{evento b) verdadero.} \\ 0 \Rightarrow \text{evento b) falso.} \end{cases}$$

$$R = \begin{cases} 1 \Rightarrow \text{evento c) verdadero.} \\ 0 \Rightarrow \text{evento c) falso.} \end{cases}$$

DISEÑO COMBINACIONAL - CON T.V.

Ejemplo: Detector de números M y N de 2 bits

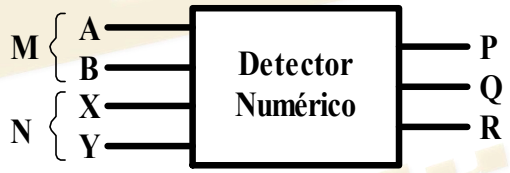
a) P→M > N **b)** Q→M divisible N **c)** R→M y N primos

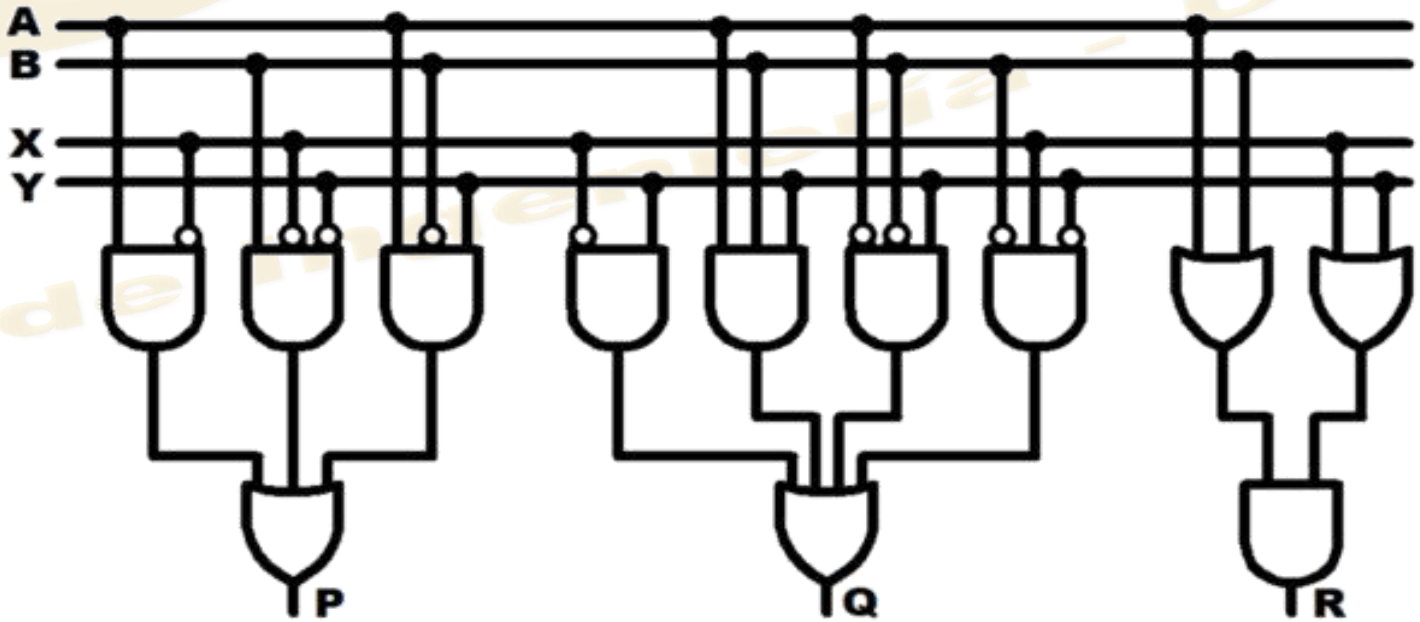
M		N		Salidas		
A	B	X	Y	P	Q	R
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	0	1	0
		1	1	0	1	0
0	1	0	0	1	0	0
		0	1	0	1	1
		1	0	0	0	1
		1	1	0	0	1
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	0	1	1
		1	1	0	0	1
1	1	0	0	1	0	0
		0	1	1	1	1
		1	0	1	0	1
		1	1	0	1	1

$$P = A.\bar{X} + B.\bar{X}.\bar{Y} + A.\bar{B}.Y$$

$$Q = \bar{X}.Y + \bar{A}.\bar{B}.Y + A.B.Y + \bar{B}.X.\bar{Y}$$

$$R = (A + B) . (X + Y)$$

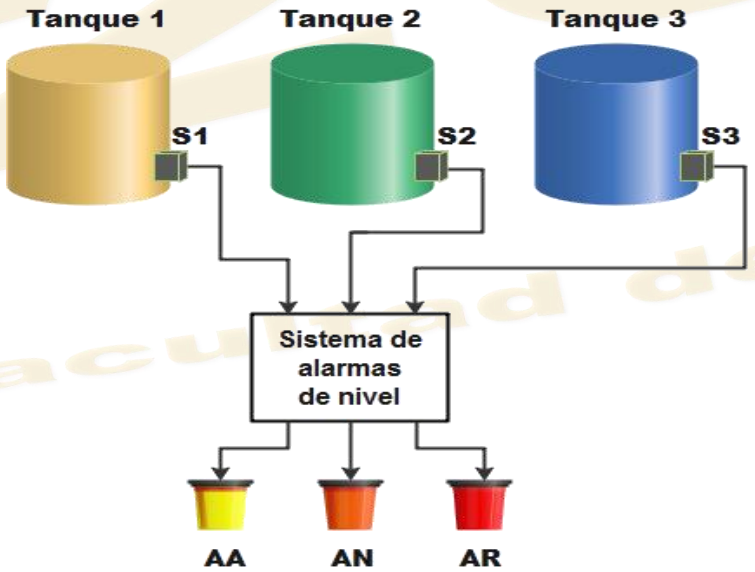




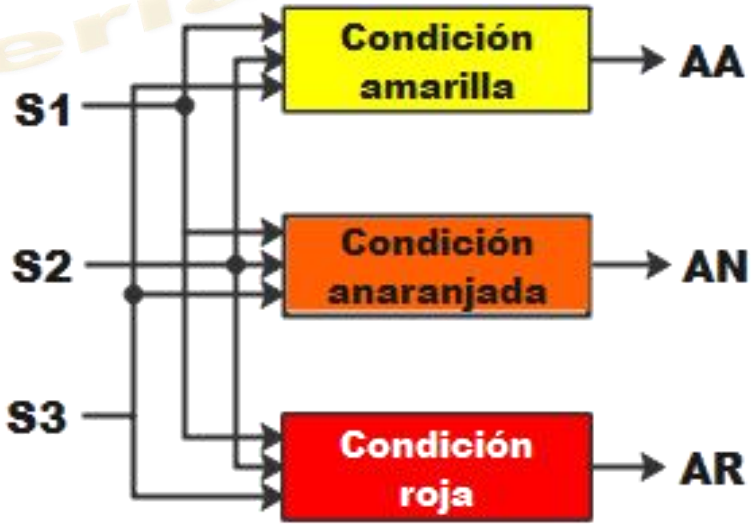
DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: Una fábrica utiliza 3 tanques de reserva para productos químicos. Un sensor de nivel ubicado cerca de la base de cada tanque, indica cuando alguno de ellos está por vaciarse. Diseñar un sistema de alarma selectivo que dé aviso cuando está por vaciarse uno de los tanques (condición amarilla), dos tanques simultáneamente (condición anaranjada) o los tres tanques (condición roja).

Variables entrada:
Sensores on/off (S1, S2, S3)
Sensor sin líquido = 1
Sensor con líquido = 0



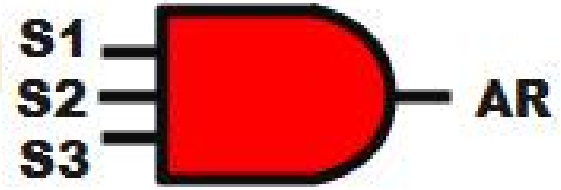
Variables de salida:
Alarma amarilla: AA
Alarma anaranjada: AN
Alarma roja: AR
Alarma activada = 1
Alarma inactiva = 0



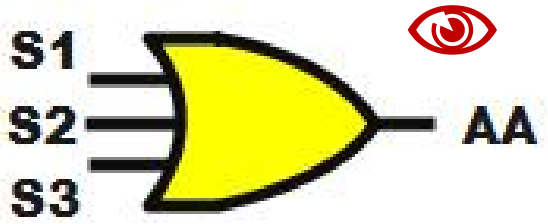
DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*

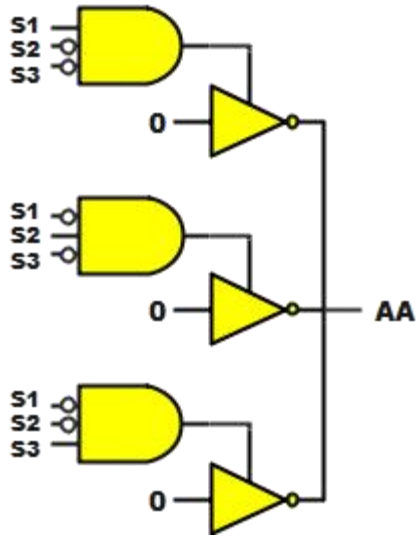
Condición roja (AR): Es la condición más simple de configurar. Se tiene cuando los tres tanques alcancen el nivel bajo. En este caso, las señales de los tres sensores deben impactar simultáneamente, es decir, S1 y S2 y S3. Esta situación también nos indica que tipo de operación lógica debe implantarse.



Condición amarilla (AA): Aparentemente es una condición sencilla; cualquiera de los tanques que alcance el nivel bajo, debe activar la alarma amarilla; es decir, S1 ó S2 ó S3, lo cual ya estaría indicando la operación a aplicar.
Error: Si se activan 2 ó 3 sensores, esta alarma también se activa.



Condición amarilla (AA) - 2º propuesta: Sólo hay 3 combinaciones de sensores que deben activar esta alarma (100 - 010 - 001). Se puede pensar en una compuerta AND como método de habilitación para dejar pasar una señal, a través de una compuerta tri-state, para que active la alarma. Como sólo una de las tri-state se habilitará por vez, se pueden unir cableadas a la salida.

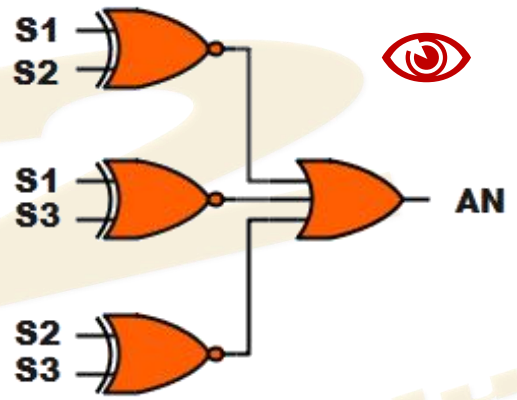


DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*

Condición anaranjada (AN): Esta situación no es tan directa en la interpretación.

Propuesta 1: Se pueden formar pares de sensores (S1-S2; S2-S3; S1-S3) y operar con función coincidencia (X-NOR); luego se concatenan con función OR final. Cualquier par que se active, prende la alarma naranja.



Error 1: si se activan los tres sensores, se activan dos pares y también se prende esta alarma, pero la condición debería ser roja.

Error 2: Cuando no hay sensores activados, las compuertas X-NOR tambien dan salida 1.

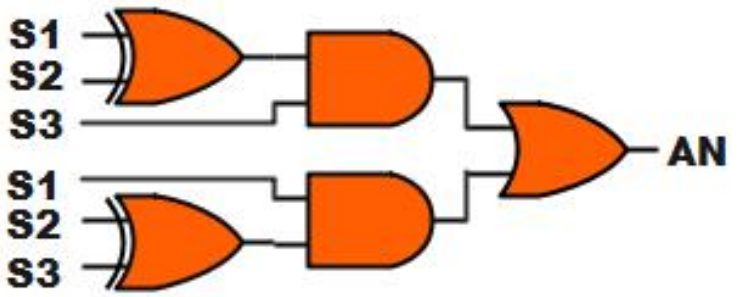
Propuesta 2: Plantear una T.V. (puntual para esta situación).

S1	S2	S3	AN
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$AN = \overline{S1}.S2.S3 + S1.\overline{S2}.S3 + S1.S2.\overline{S3}$$

$$AN = (\overline{S1}.S2 + S1.\overline{S2}).S3 + S1.(\overline{S2}.S3 + S2.\overline{S3})$$

$$AN = (S1 \oplus S2).S3 + S1.(S2 \oplus S3)$$

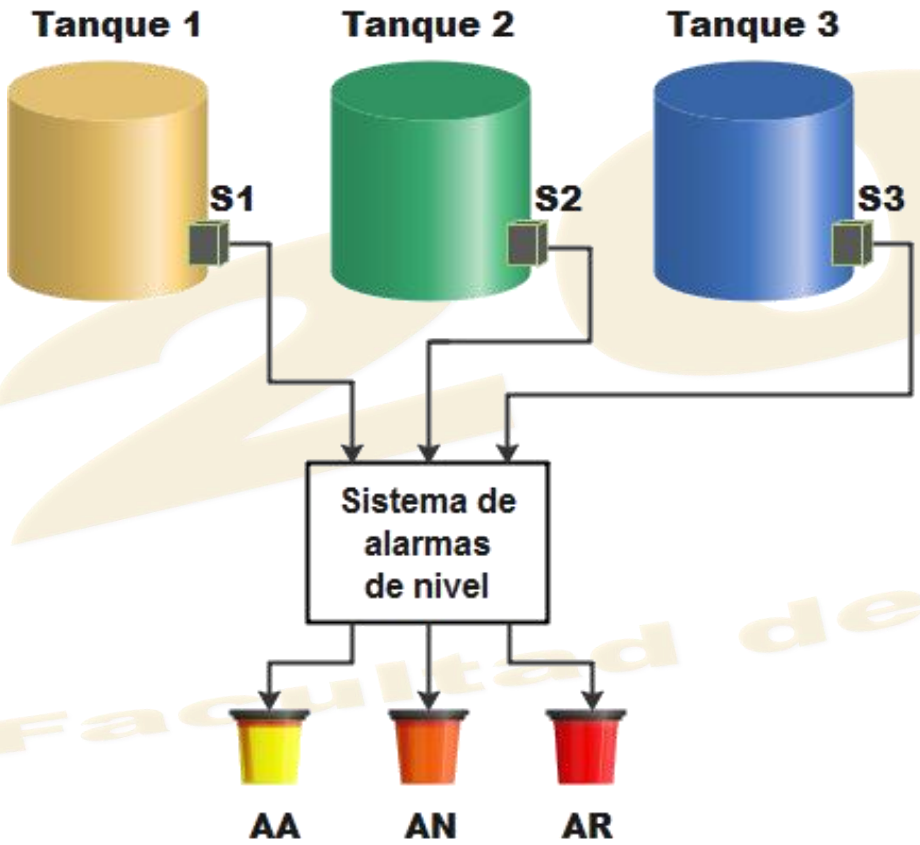


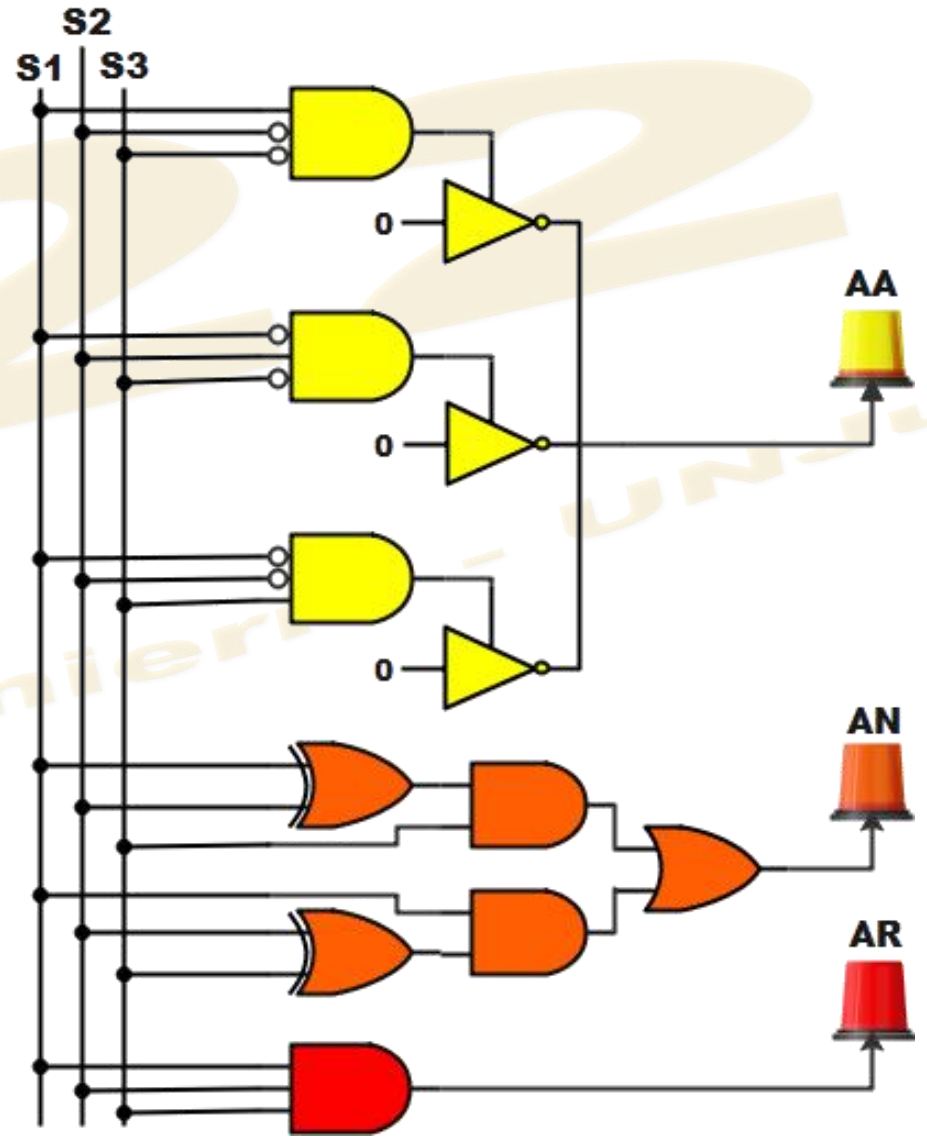
DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*

(Floyd, pg. 275)

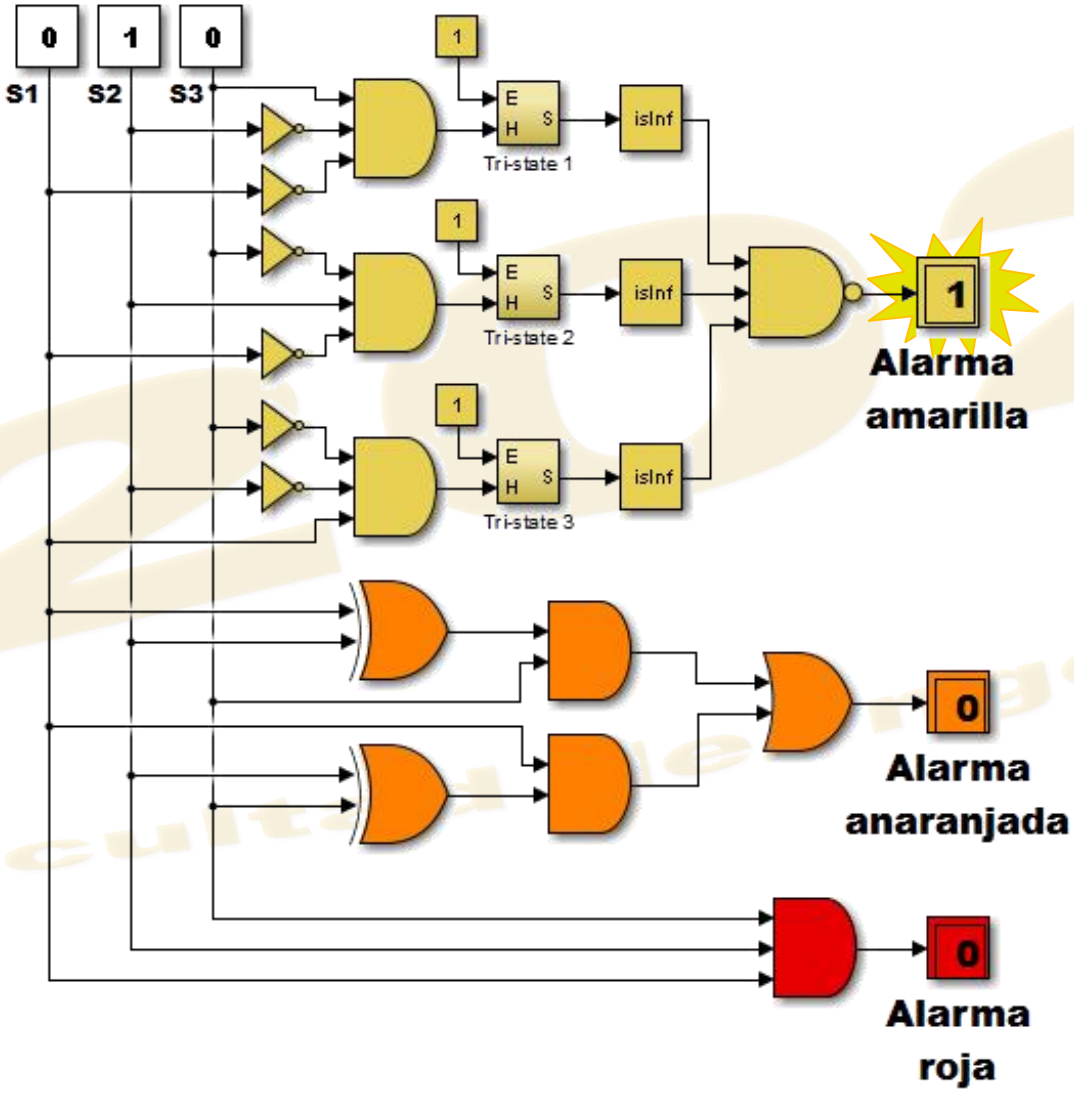
Circuito final

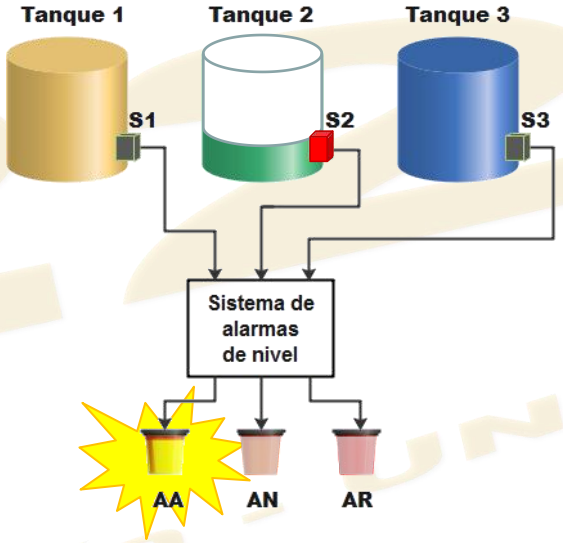




DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*





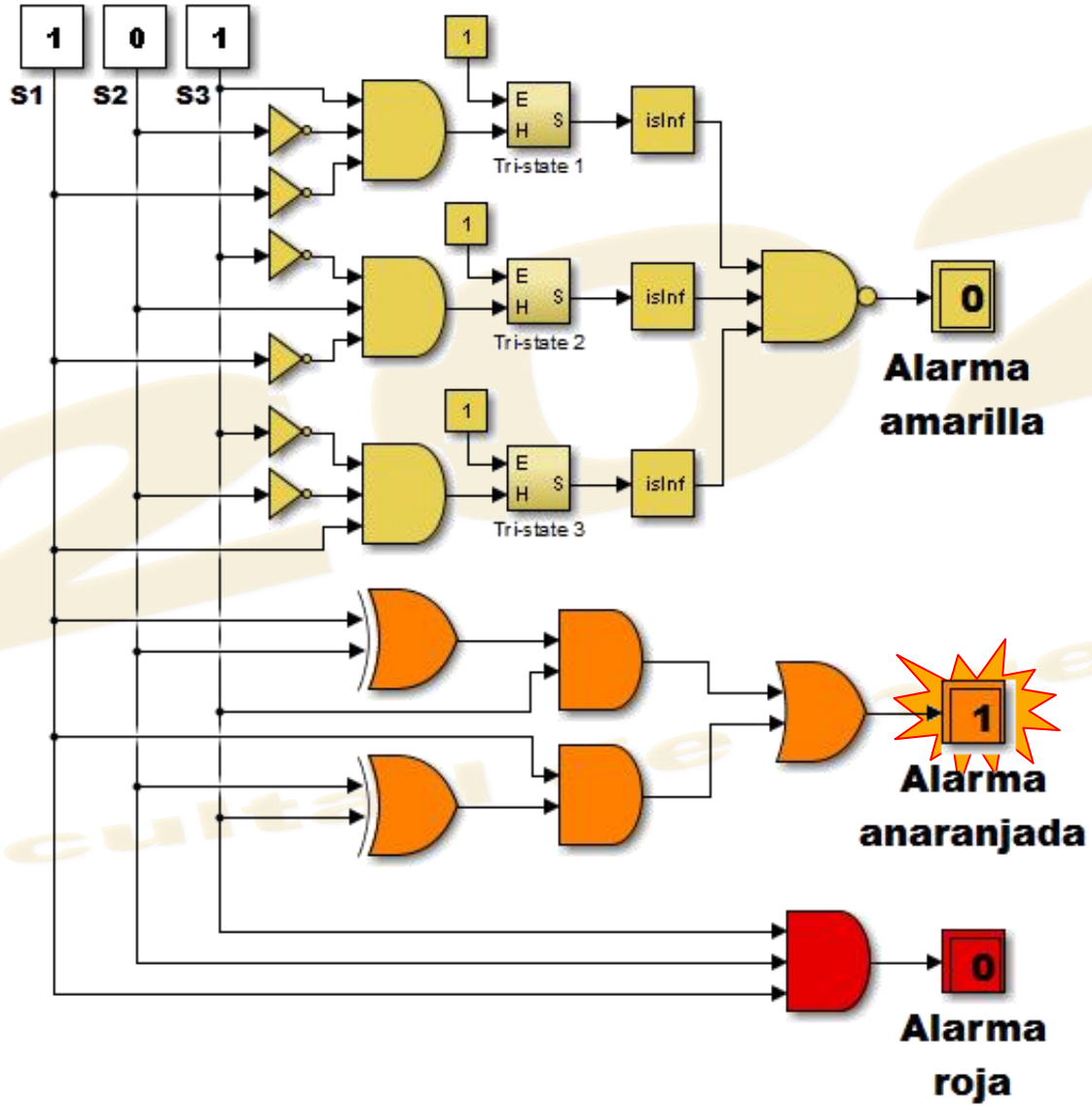
Simulación en Matlab

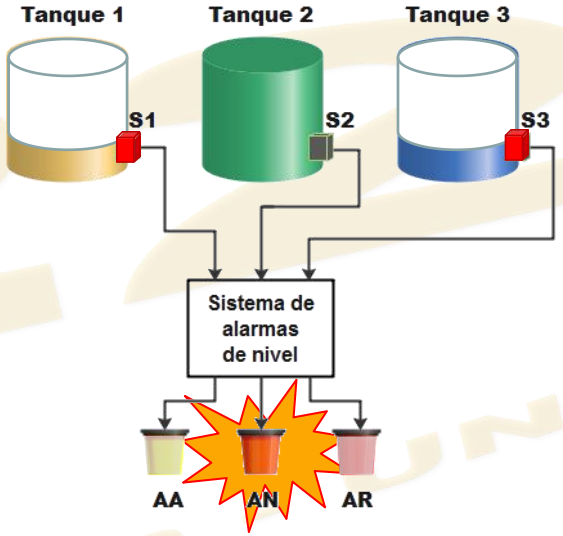
Sensor 2 activado

Condición amarilla

DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*





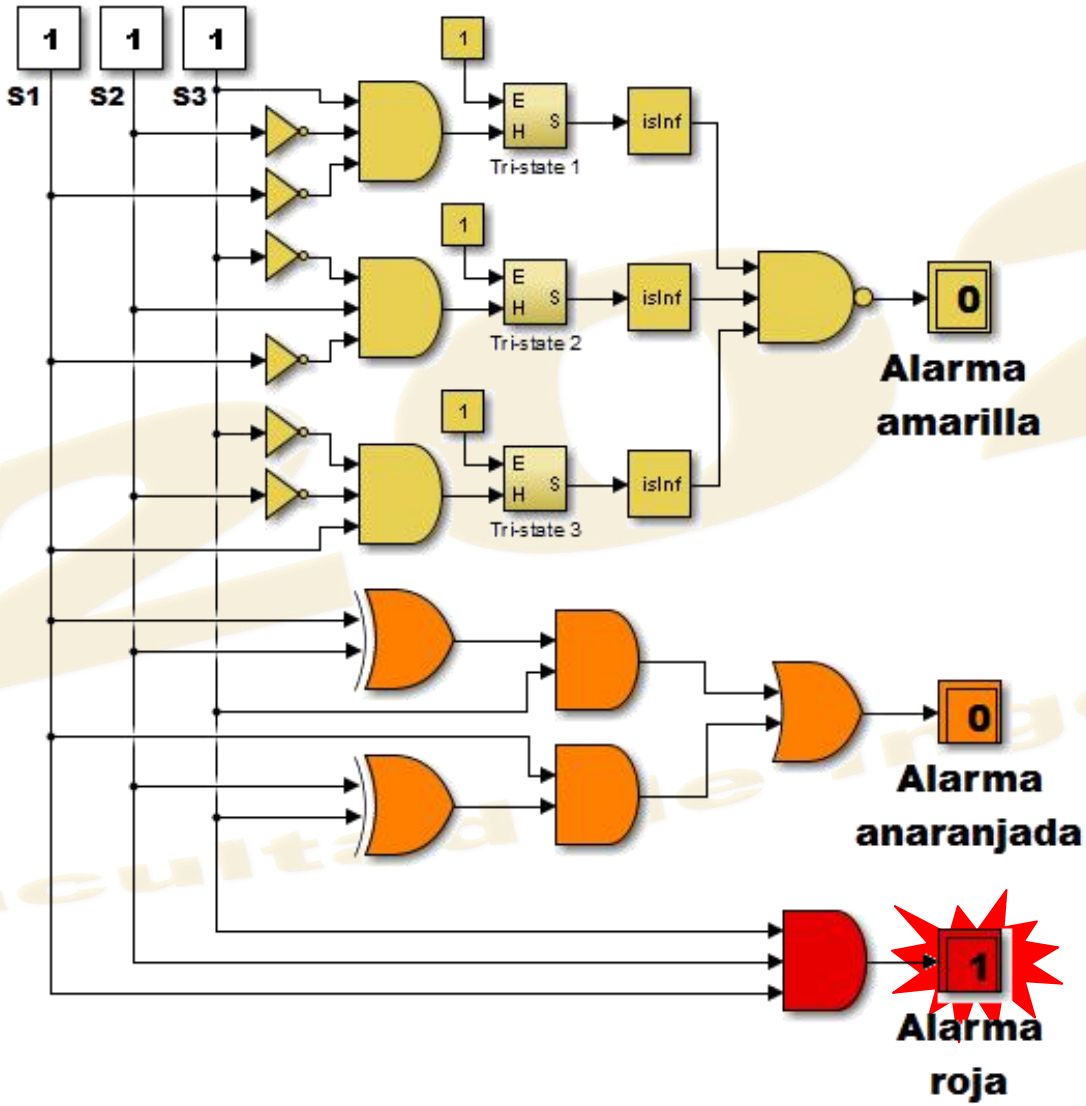
Simulación en Matlab

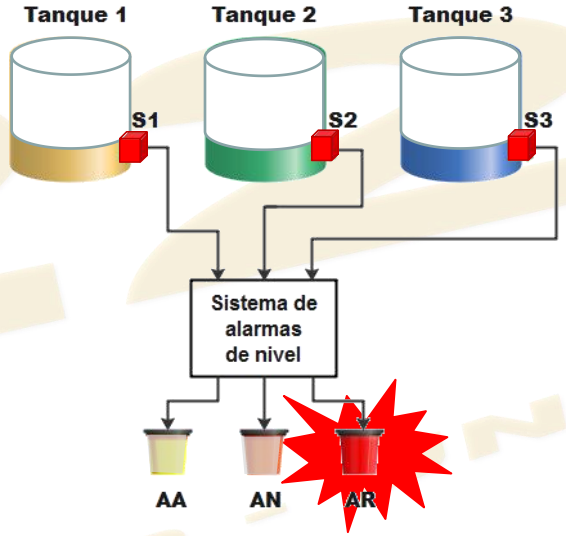
Sensores 1 y 3 activados

Condición anaranjada

DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño directo: sistema de alarma selectivo *(continuación)*





Simulación en Matlab

Sensores 1, 2 y 3 activados

Condición roja

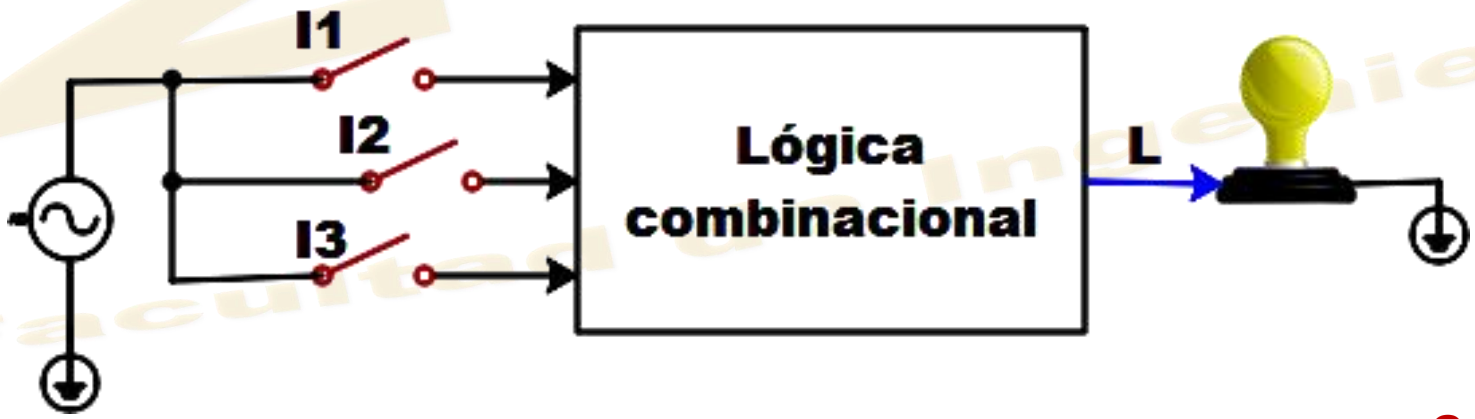
DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño por T.V.: Encender una lámpara desde tres puntos diferentes.

Esta configuración quizás no sea muy fácil de analizar en forma directa, por lo que se propone plantear la situación mediante una tabla de verdad (T.V.).

Variables entrada:
Interruptores (I1, I2, I3)
Interruptor abierto = 0
Interruptor cerrado = 1

Variable de salida:
Luz: L
Luz encendida = 1
Luz apagada = 0



I1	I2	I3	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Como la función no se conoce, se imponen los valores deseados (a L) en la T.V.

(Brown, pg. 50)

DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño por T.V.: Encender una lámpara desde tres puntos diferentes.

Como la función no se conoce, se imponen los valores deseados (a L) en la T.V.

Nótese que tienen salida 1 sólo aquellas combinaciones que tengan un número impar de interruptores activados. En otras palabras, el diseño se hace cambiando de estado de la salida por adyacencias (siempre cambia un solo interruptor).

Luego se sintetiza la función. Como no hay posibilidad de minimización con Karnaugh, se puede extraer un formato canónico y reagrupar.

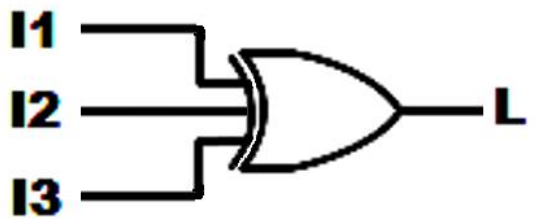
I1	I2	I3	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

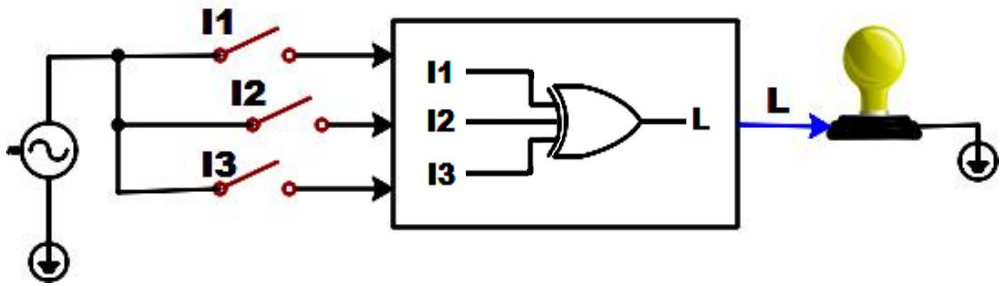
$$L = \overline{I1}.I2.\overline{I3} + I1.\overline{I2}.\overline{I3} + \overline{I1}.\overline{I2}.I3 + I1.I2.I3$$

$$L = (\overline{I1}.I2 + I1.\overline{I2}).\overline{I3} + (\overline{I1}.\overline{I2} + I1.I2).I3$$

$$L = (I1 \oplus I2).\overline{I3} + \overline{(I1 \oplus I2)}.I3 = I1 \oplus I2 \oplus I3$$

L	I1 I2			
	00	01	11	10
I3 = 0		×		×
I3 = 1	×		×	

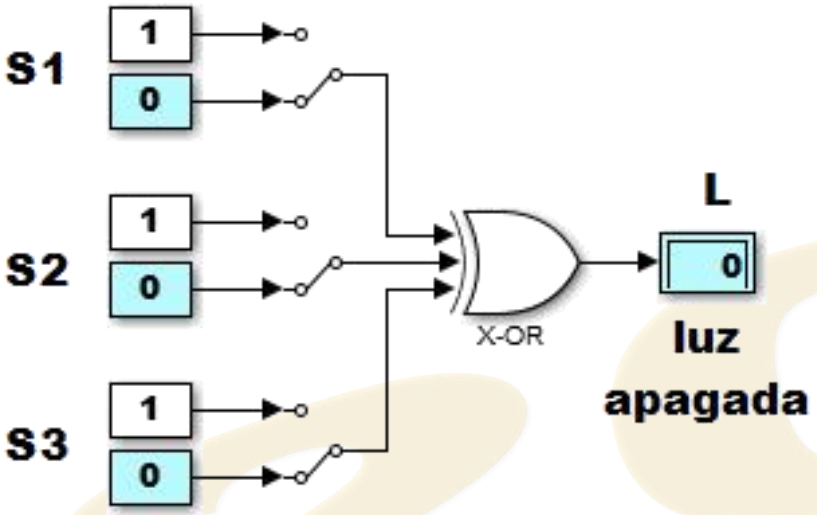




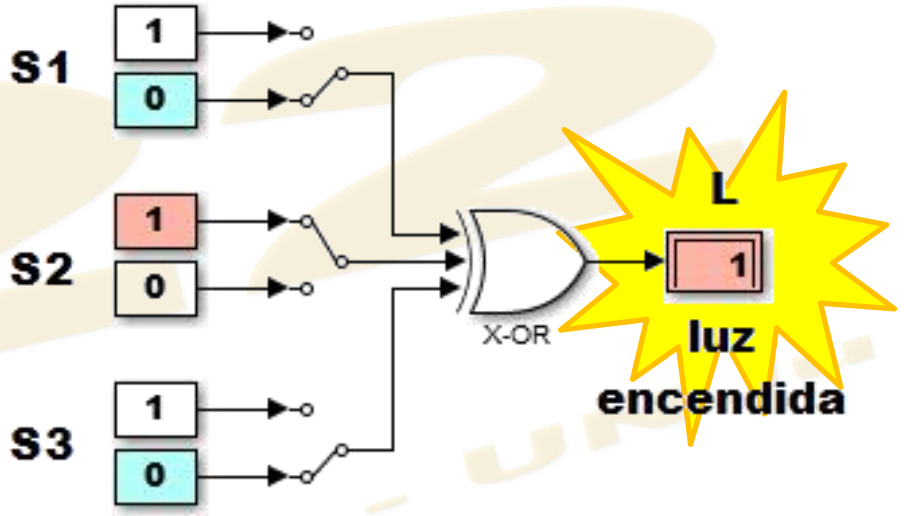
Ver Brown, pg. 50

DISEÑO COMBINACIONAL - EJEMPLO

Ejemplo diseño por T.V.: Encender una lámpara desde tres puntos diferentes.



I1	I2	I3	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Simulación en Matlab