Circuitos Electrónicos Digitales

Tema 2
Análisis y diseño de circuitos combinacionales

Universidad de Sevilla

Indice

1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

NEW 24/25

Indice

1. *Análisis lógico de circuitos combinacionales*
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

Análisis lógico de de Circuitos Combinacionales

El análisis lógico de un circuito consiste en encontrar la función lógica que realiza.

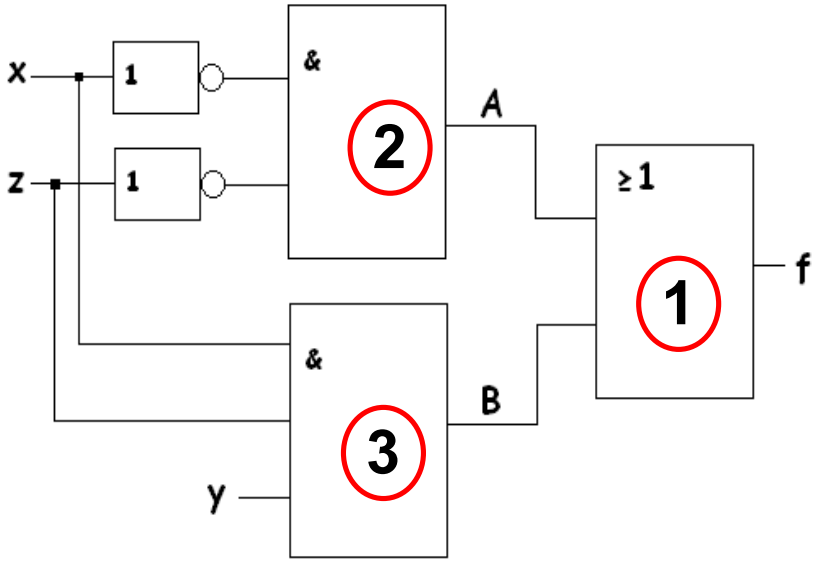
La solución podrá darse en forma de:

- Expresión algebraica normalizada (sp o ps)
- Tabla de Verdad o K-mapa
- En algunas ocasiones, descripción verbal de su función

Análisis de Circuitos Combinacionales

Ejemplo:

Circuito:



De la puerta de salida:
1: $f(x,y,z) = A + B$

De las puertas 2 y 3
2: $A = \bar{x}\bar{z}$
3: $B = xyz$

Sustituyendo:
$\boldsymbol{f(x,y,z) = \bar{x}\,\bar{z} + xyz}$

Tabla de Verdad

xyz	f(x,y,z)
000	1
001	0
010	1
011	0
100	0
101	0
110	0
111	1

Analizando la expresión algebraica:

$f(x,y,z) = 1$ sii

- $xyz = 1$, si (x=1, y=1, z=1)
- ó
- $\bar{x}\,\bar{z} = 1$, si (x=0, z=0)

Indice

1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

¿En qué consiste el diseño de un Circ. Combinacional?

El diseño (o síntesis) de un circuito es el proceso inverso al análisis:

> *A partir de la especificación inicial de un problema, se debe obtener (diseñar) un circuito lógico que cumpla determinados requisites*

El término **combinacional** hace referencia a que en valor en la salida depende de la combinación de los valores de entrada. Por tanto, pueden modelarse mediante funciones lógicas

Con carácter general, el problema de diseño se reduce a encontrar una expresión algebraica simplificada a partir de la cual se obtendrá el circuito correspondiente.

En la medida de lo posible, buscaremos un circuito **lo más reducido posible** (menor consumo, más fiable, más barato, etc.)

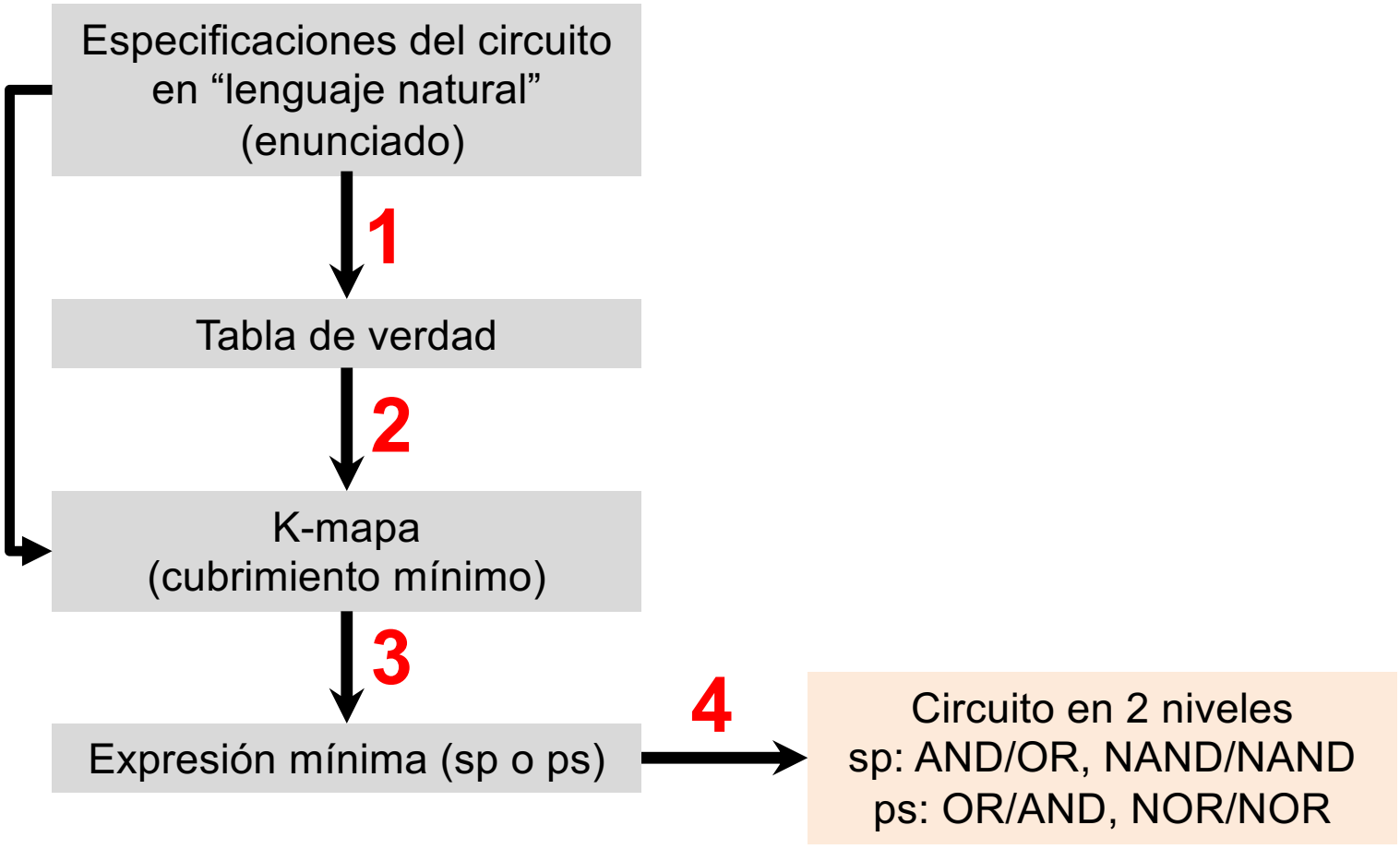
Objetivos, restricciones y criterios de coste

Objetivo:
Dada una especificación inicial de una función, obtener un circuito "óptimo" que la realice

Criterios de coste:
Obtener un circuito con:
- el menor número de puertas
- el menor número de entradas (lo más pequeñas posibles)

Restricciones:
- Estructura en **dos niveles** (tres para simple rail)
- Uso de puertas AND, OR, NOT, NAND y NOR.
- No considerar fan-in (número de entradas de las puertas) ni fan-out (numero de conexiones de la salida de una puerta) como restricciones.
- No reduce inversores

Pasos en el proceso de diseño



Diseño de Circuitos Combinacionales

Pasos del proceso

Paso 1: Descripción textual -> Tabla de verdad

- Identificar las variables de entrada y la función (o funciones de salida).
- Determinar los criterios de los valores lógicos (qué significa “0” y qué significa “1”)
- Obtener la tabla de verdad (o directamentel el Kmapa) (para cada función de salida)

Paso 2: Obtener el K-mapa de la función

A partir de la tabla de verdad anterior o de la especificación establecida, se obtiene el K-mapa de la función a implementar.

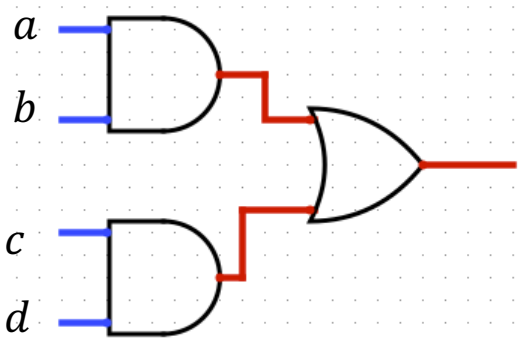
Paso 3: Obtener una expresión algebraica reducida (mínima)

Usando el método gráfico del K-mapa, obtendremos una expresión normalizada (sp o ps) reducida

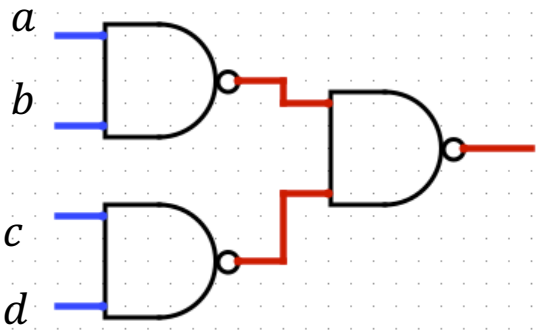
Paso 4: Obtener el circuito en dos niveles a partir de la expresión algebraica

Paso 4: Circuitos en 2 niveles (+inversores)

suma de productos (sp)

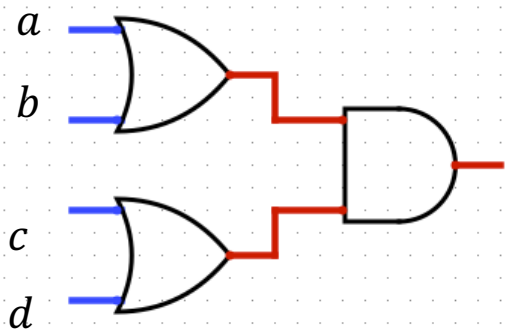


$f = ab + cd$

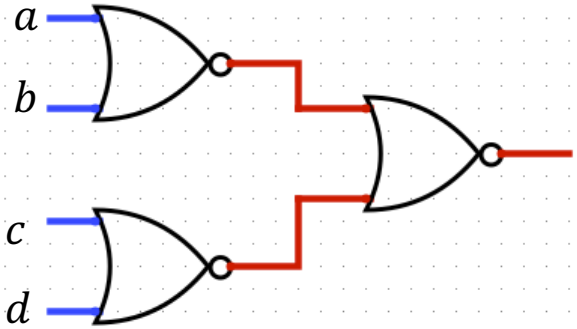


$f = \overline{\overline{(ab + cd)}} = \overline{\overline{ab}} \cdot \overline{\overline{cd}}$

producto de sumas (ps)



$f = (a + b)\,(c + d)$



$f = \overline{\overline{(a + b)(c + d)}} = \overline{\overline{(a + b)} + \overline{(c + d)}}$

¿Cómo nos enfrentamos a un problema de diseño?

- ¿Entiendo bien el enunciado? Esto es, ¿tengo claro qué debe hacer el circuito?

- Identificar señales de entrada (variables) y de salida (función o funciones)

- Identificar o establecer los criterios de activación (qué significa un "0" y qué significa un "1" en las señales y funciones

- Obtener una expresión algebraica reducida de la función (sp mínimo o ps mínimo) => KMAPA (apartado 3)

- A partir de las funciones reducidas, diseñar el circuito con puertas lógicas

Ejemplo de diseño 1: circuito votador de 3 entradas

Diseñe un circuito que genere el resultado de la votación de tres señales binarias.

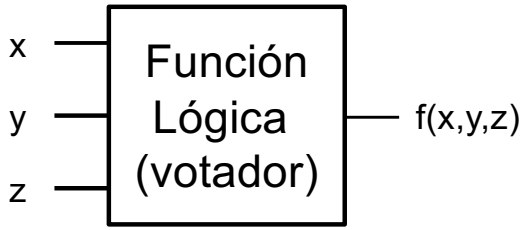


Diagrama de bloques

Ejemplo de diseño 1: circuito votador de 3 entradas

Diseñe un circuito que genere el resultado de la votación de tres señales binarias.

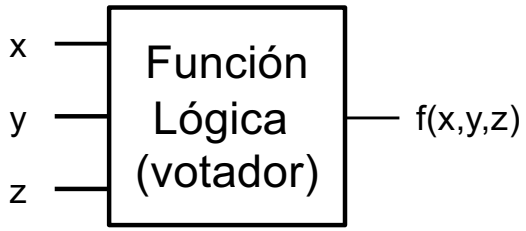


Diagrama de bloques

x y z	f
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	1

Tabla de Verdad

A partir de la Tabla de Verdad (en forma canónica):

$$f(x,y,z) = \sum(3,5,6,7) = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$$

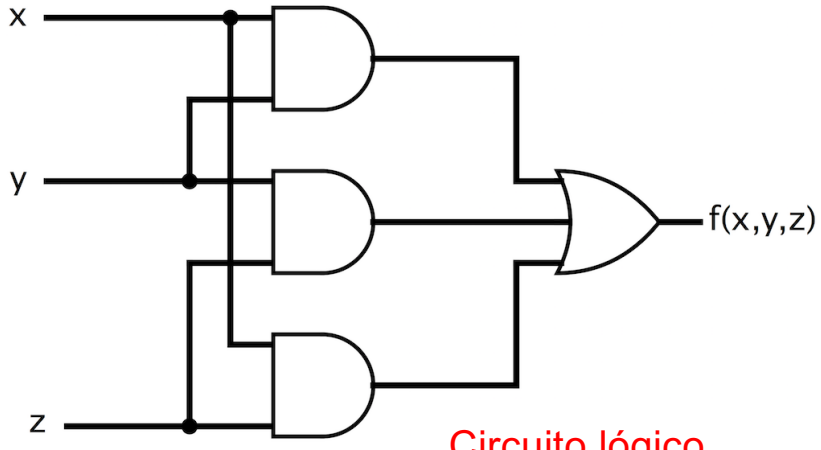
Simplificando (por ahora, algebraicamente):

$$f(x,y,z) = \overbrace{xy(\bar{z}+z)}^{(m6+m7)} + \overbrace{xz(\bar{y}+y)}^{(m5+m7)} + \overbrace{yz(\bar{x}+x)}^{(m3+m7)}$$

Por tanto:

$$\boldsymbol{f(x,y,z) = xy + xz + yz}$$

K-mapa simplificará gráficamente

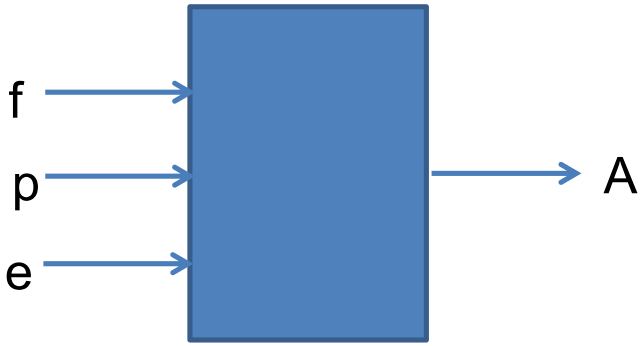


Circuito lógico

Ejemplo de diseño 2: circuito de alarma de un coche

Se desea diseñar un Sistema de alarma muy simple para un coche que deberá sonar si:

- Si el motor está apagado y las puertas abiertas
- Si el motor está encendido y el freno de mano está puesto



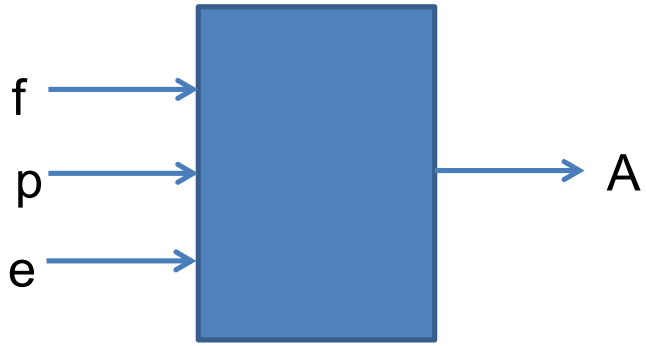
Criterios:

A =1, suena la alarma
f =1, freno de mano puesto
p = 1, alguna puerta abierta
e =1, motor encendido

$A(f,p,e) = 1 \Leftrightarrow (e = 0\ y\ p = 1)\ o\ (e = 1\ y\ f = 1)$, por tanto

$$\boldsymbol{A(f,p,e) = \bar{e} \bullet p + e \bullet f}$$

Ejemplo de diseño 2: circuito de alarma de un coche

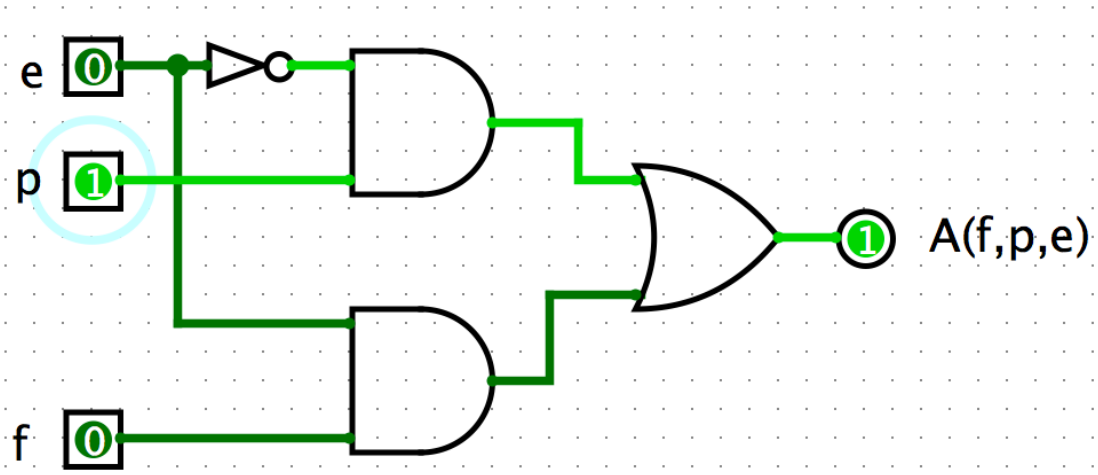


Criterios:

A =1, suena la alarma
f =1, freno de mano puesto
p = 1, alguna puerta abierta
e =1, motor encendido

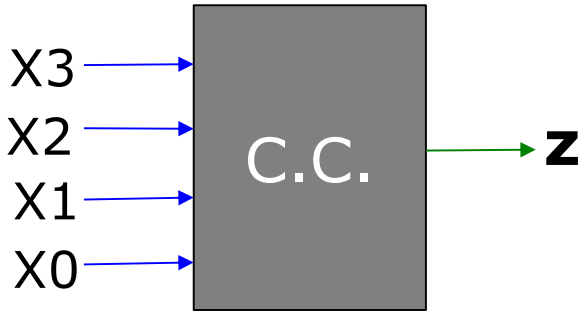
$A(f,p,e) = 1 \Leftrightarrow (e = 0\ y\ p = 1)\ o\ (e = 1\ y\ f = 1)$, por tanto

$$\boldsymbol{A(f,p,e) = \bar{e} \bullet p + e \bullet f}$$



Ejemplo de diseño CC (paso 1)

Ejemplo 3 (paso 1):
Suponga que los números entre 0 y 15 están representados en binario con cuatro bits: X3-X0, donde X3 es el bit más significativo. Diseñe un circuito que de salida Z = 1 si y sólo si el número X3-X0 es primo.



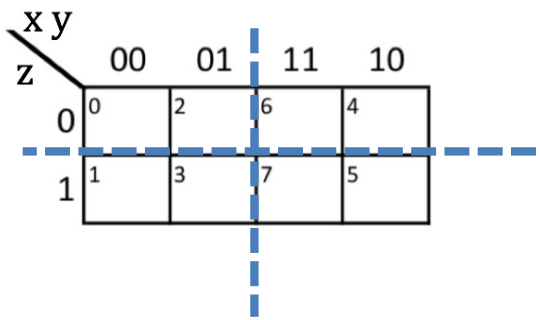
X3 X2 X1 X0	z
0 0 0 0	0
0 0 0 1	0
0 0 1 0	1
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	0
1 1 0 1	1
1 1 1 0	0
1 1 1 1	0

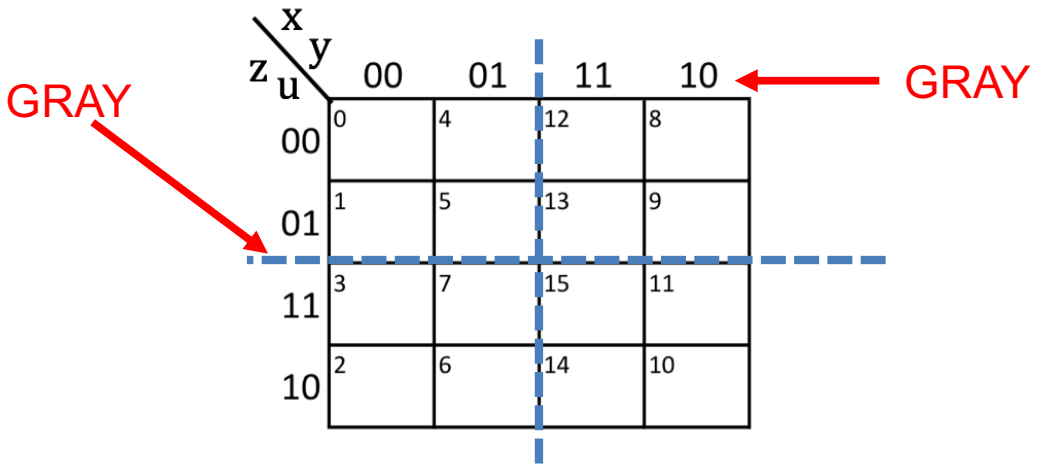
Indice

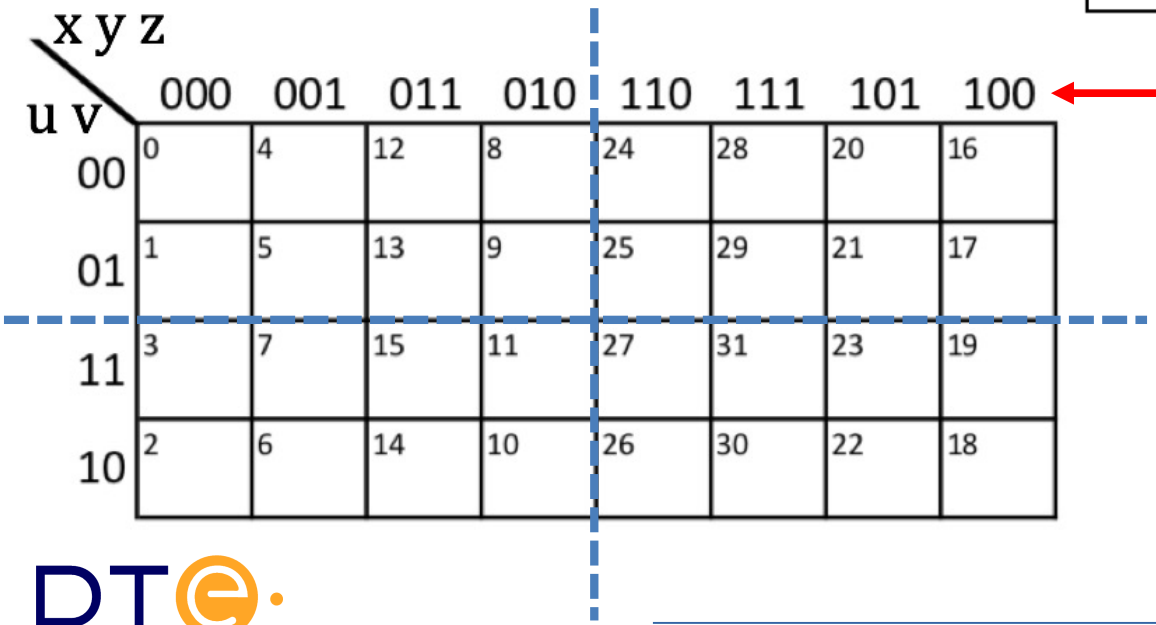
1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

Mapas de Karnaugh

El mapa de Karnaugh (Kmapa) da la misma información que la TV, pero ordena los mintérminos y maxtérminos con propiedades muy interesantes:







Expresiones normalizadas sp y ps a partir del k-mapa

(Paso 3: Obtener la expresión mínima en dos niveles)

- **Expresión mínima como suma de productos**

  - Nos fijamos en los 1's del K-mapa, o mintérminos
  - cubrimos todos los mintérminos para conseguir el menor número términos producto **(implicantes**) con menor número de variables

- **Expresión mínima como producto de sumas**

  - Nos fijamos en los 0's del K-mapa, o maxtérminos.
  - cubrimos los maxtérminos para conseguir términos suma **(implicadas)** con menor número de variables

$si\ x = 0 \xrightarrow{"da\ lugar\ a..."} \cdot \bar{x}$

$si\ x = 1 \xrightarrow{"da\ lugar\ a..."} \cdot x$

Definición: mintérminos adyacentes

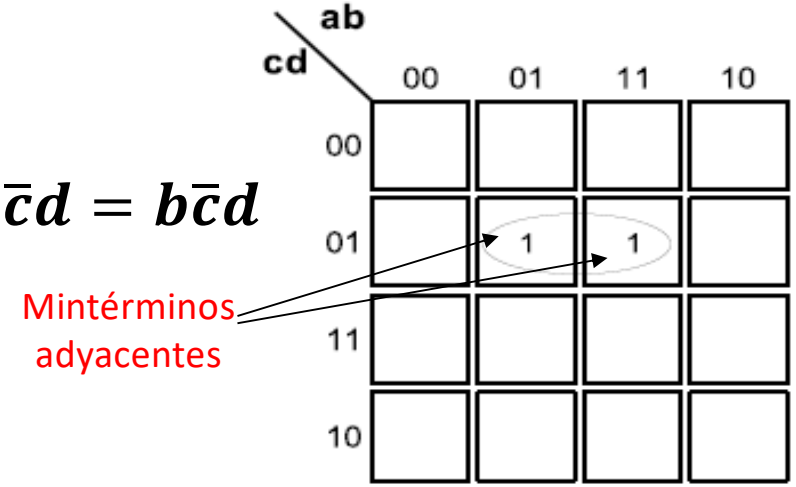
- Dos términos (mintérminos o maxtérminos) son adyacentes, si se diferencia solo en el valor de una variable.

- Se identifican bien en el Kmapa porque son “vecinos” (arriba, abajo, izquierda o derecha) o “reflejados”

- *La suma de dos mintérminos adyacentes genera un único término producto en el que se ha eliminado la variable que cambia*

- Ejemplo:

$$\boldsymbol{m_5 + m_{15} = \bar{a}b\bar{c}d + ab\bar{c}d = b\bar{c}d}$$



$si\ x = 0 \xrightarrow{"da\ lugar\ a..."} +x$

$si\ x = 1 \xrightarrow{"da\ lugar\ a..."} +\bar{x}$

Definición: Maxtérminos adyacentes

- Análogamente, el producto de dos Maxtérminos adyacentes genera un único término suma el que se ha eliminado la variable que cambia

- Ejemplo:

$$\boldsymbol{M_0 M_1 = (a + b + c + d)\left(a + b + c + \bar{d}\right) = (a + b + c)}$$

c d \ a b	00	01	11	10
00	0 0	4	12	8
01	1 0	5	13	9
11	3	7	15	11
10	2	6	14	10

$si\ x = 0 \xrightarrow{"da\ lugar\ a..."} \cdot \bar{x}$

$si\ x = 1 \xrightarrow{"da\ lugar\ a..."} \cdot x$

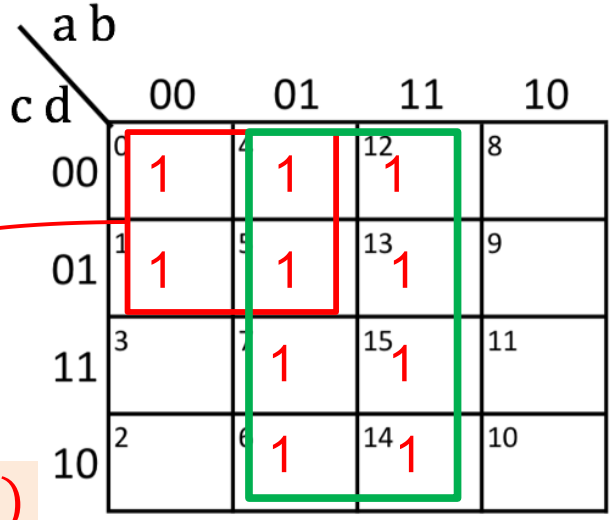
Definición: Implicante de orden n

Es una agrupación formada por 2^n mintérminos adyacentes.

Genera un término producto en el que se han eliminado n variables

Por ejemplo:

- Implicante de orden 2: 4 mintérminos adyacentes (elimina 2 var.)
- Implicante de orden 3: 8 mintérminos adyacentes (elimina 3 var.)



$m_0 + m_1 + m_4 + m_5 = \bar{a}\bar{c}$

$m_4 + m_5 + m_7 + m_6 + m_{12} + m_{13} + m_{15} + m_{14} = b$

$$\boldsymbol{f(a,b,c,d) = \bar{a}\bar{c} + b \quad (SP)}$$

Definición: Implicada de orden n

$si\ x = 0 \xrightarrow{"da\ lugar\ a..."} +\ x$

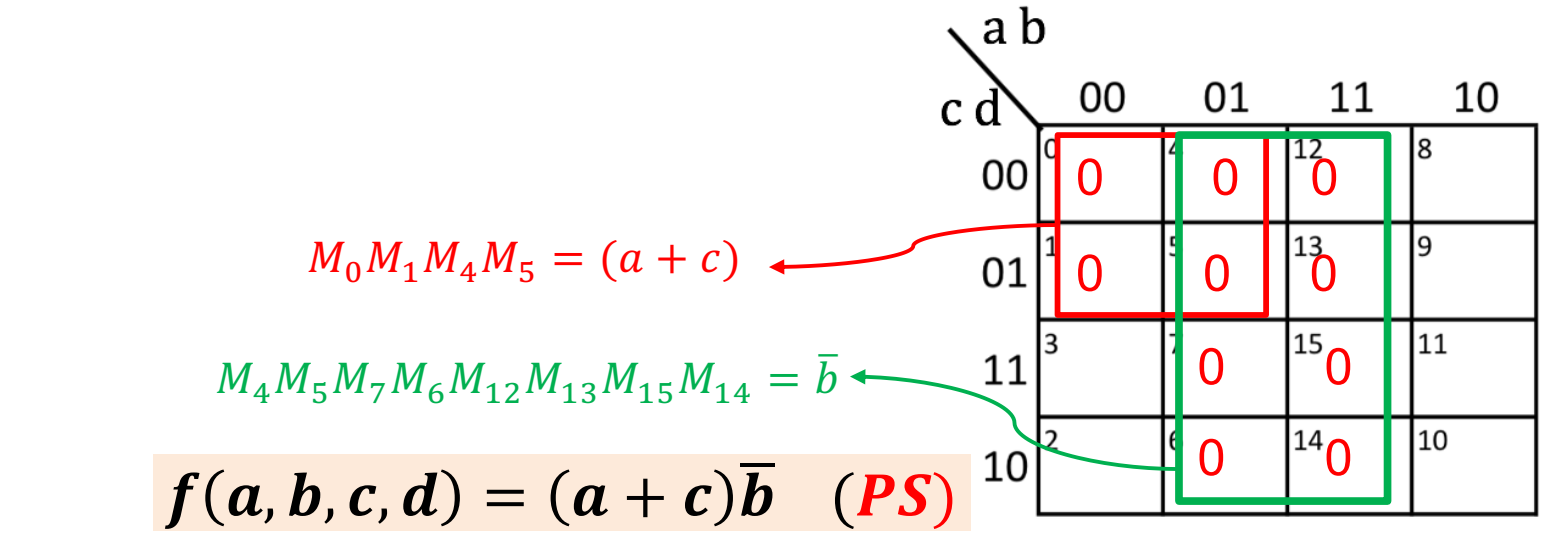
$si\ x = 1 \xrightarrow{"da\ lugar\ a..."} +\ \bar{x}$

Es una agrupación formada por 2^n Maxtérminos adyacentes.

Genera un término suma en el que se han eliminado n variables

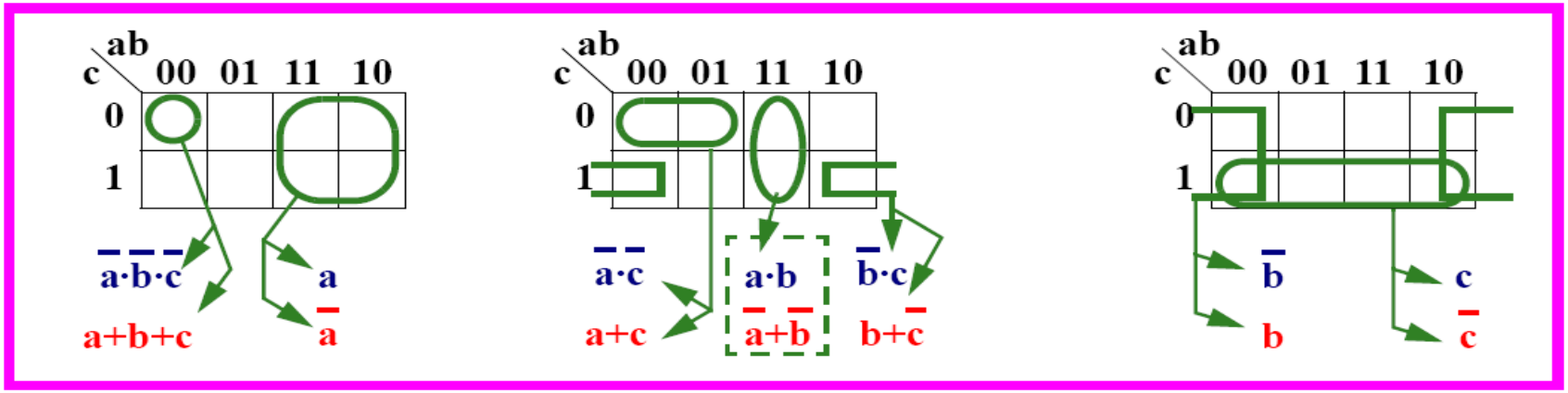
Por ejemplo:

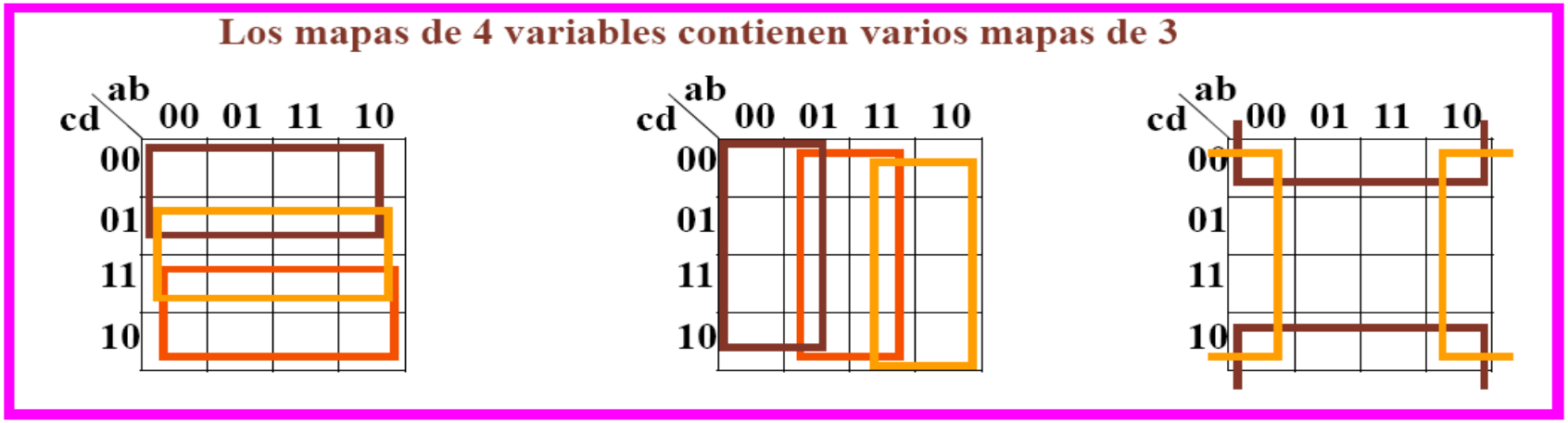
- Implicada de orden 2: 4 Maxtérminos adyacentes (elimina 2 var.)
- Implicada de orden 3: 8 Maxtérminos adyacentes (elimina 3 var.)



Simplificación de funciones usando k-mapa

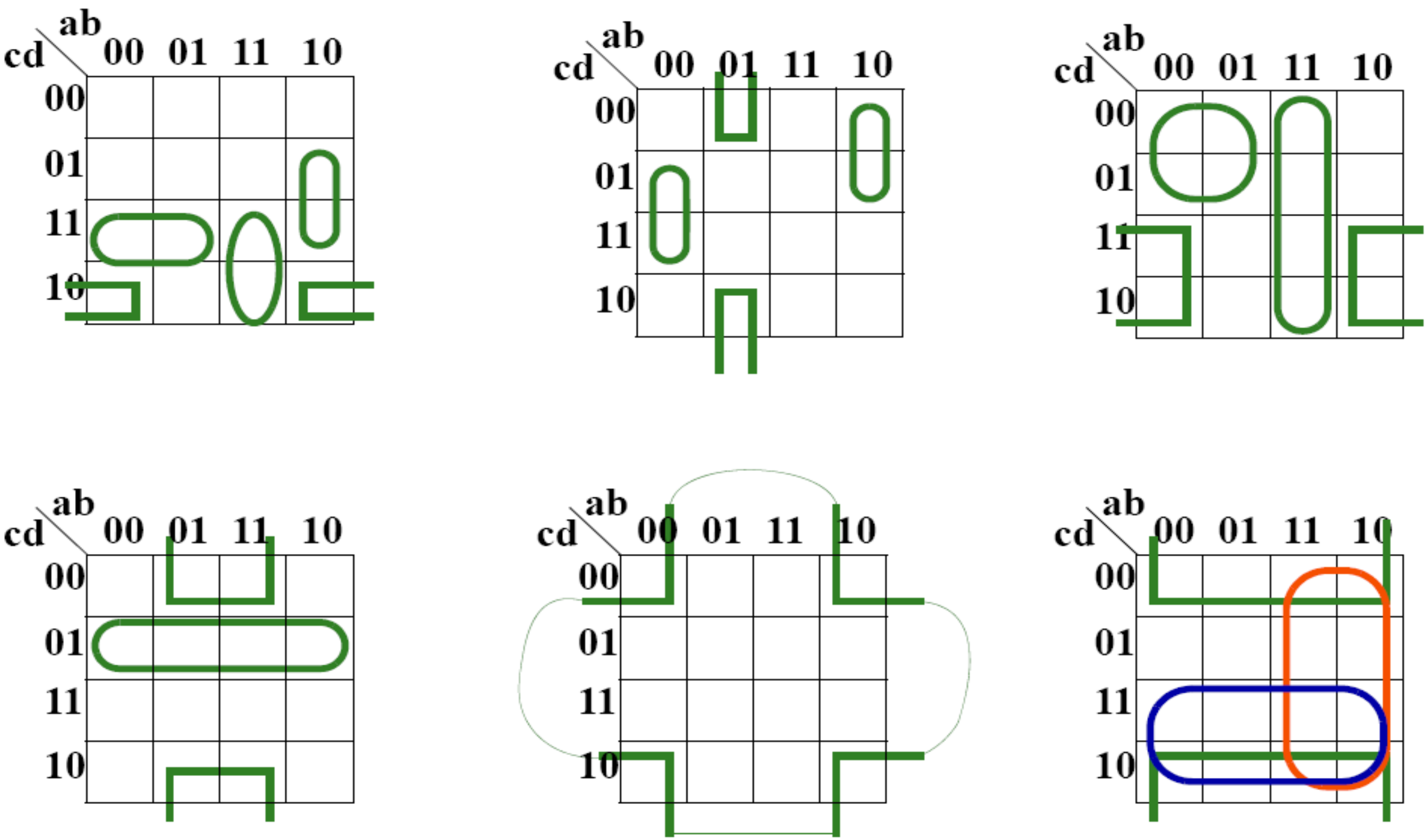
Agrupaciones posibles



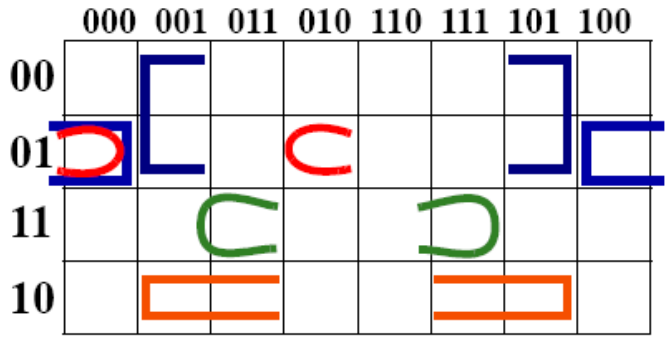


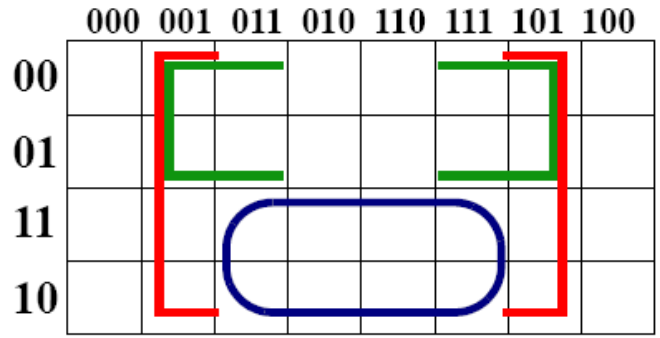
Simplificación de funciones usando k-mapa

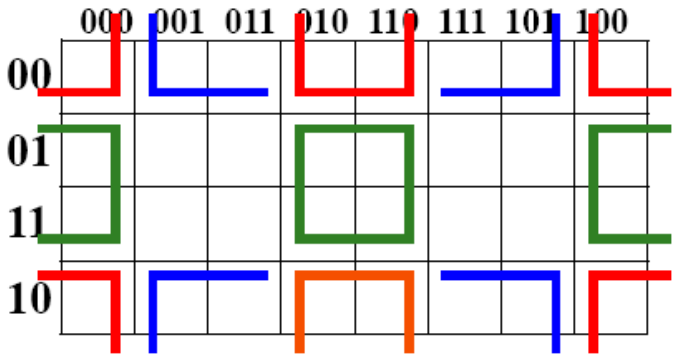
Agrupaciones posibles

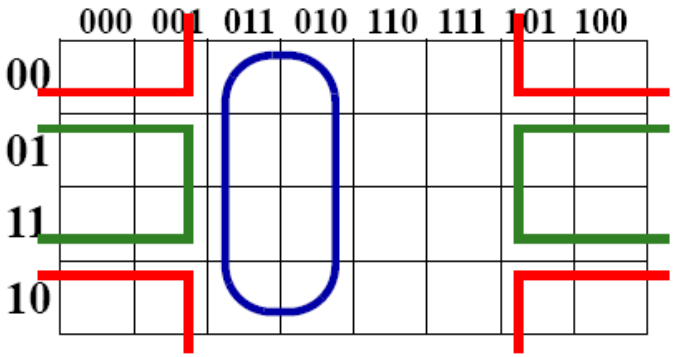


Simplificación de funciones usando k-mapa



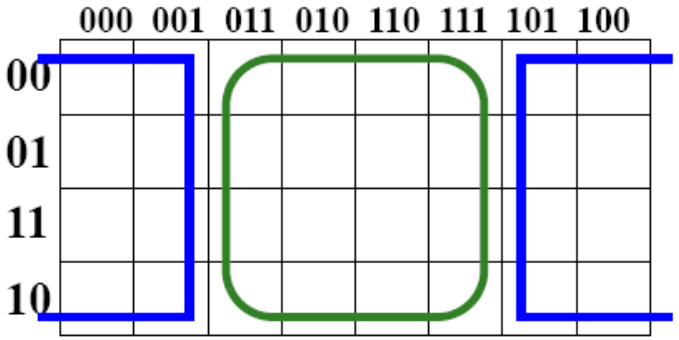


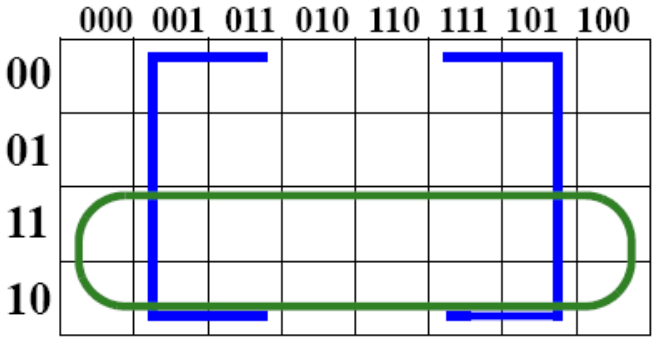


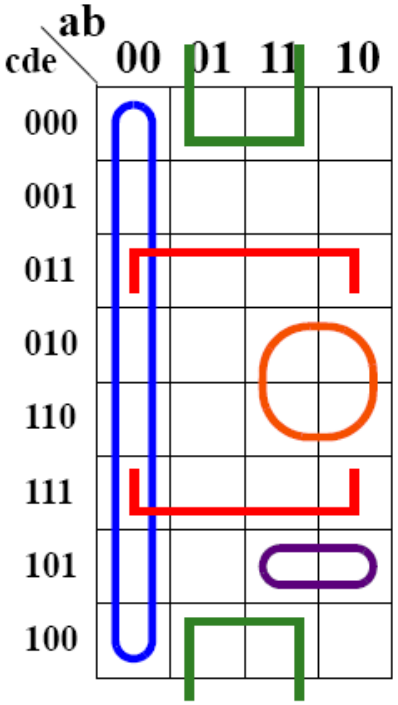


Simplificación de funciones usando k-mapa

Agrupaciones posibles







$$si\ x = 0 \xrightarrow{\text{"da lugar a..."}} \cdot \bar{x}$$
$$si\ x = 1 \xrightarrow{\text{"da lugar a..."}} \cdot x$$

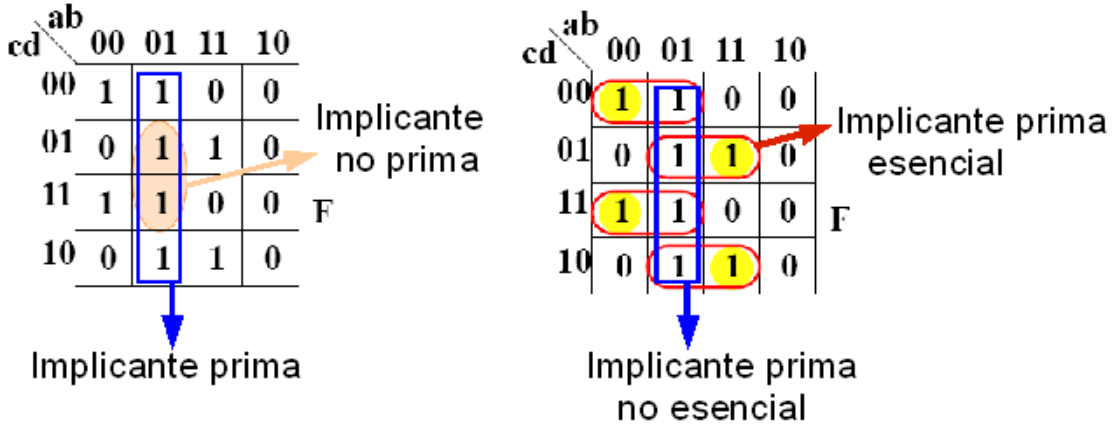
Definiciones:
Implicante Prima e Implicante Prima esencial

Una Implicante se dice que es **prima** si no está cubierta por ninguna otra implicante más grande

La suma de productos mínima de la función se obtiene sumando el menor número de implicantes primas

El menor número de implicantes y los más grandes posible

Una implicante prima se dice que es **esencial** si cubre algún mintérmino no incluido en ninguna otra implicante prima. Al mintérmino se le denomina **distinguido**.



$$si\ x = 0 \xrightarrow{\text{"da lugar a..."}} \cdot \bar{x}$$
$$si\ x = 1 \xrightarrow{\text{"da lugar a..."}} \cdot x$$

Ejemplos de obtención de la expresión sp mín.

$$f(a,b,c,d) = \sum(0,3,4,5,6,7,13,14)$$

c d \ a b	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$$si\ x = 0 \xrightarrow{"da\ lugar\ a..."} +x$$
$$si\ x = 1 \xrightarrow{"da\ lugar\ a..."} +\bar{x}$$

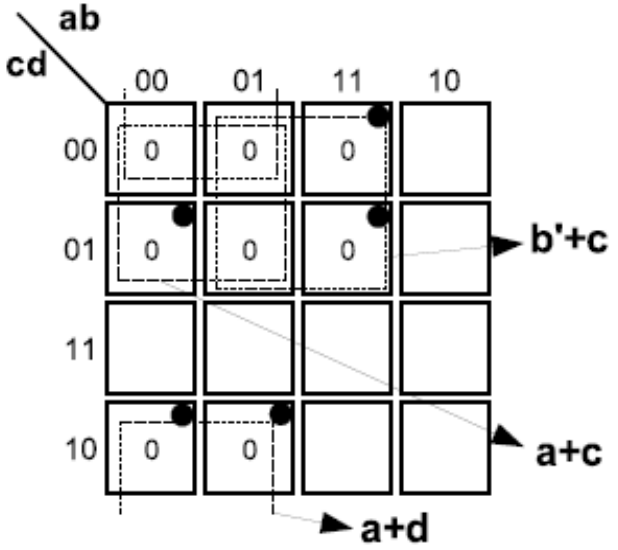
Definiciones:
Implicada Prima e Implicada Prima esencial

Una **Implicada** (o término suma) se dice que es **prima** si no está cubierta por ninguna otra implicada de la función.

El producto de sumas mínimo de la función se obtiene multiplicando el menor número de implicadas primas

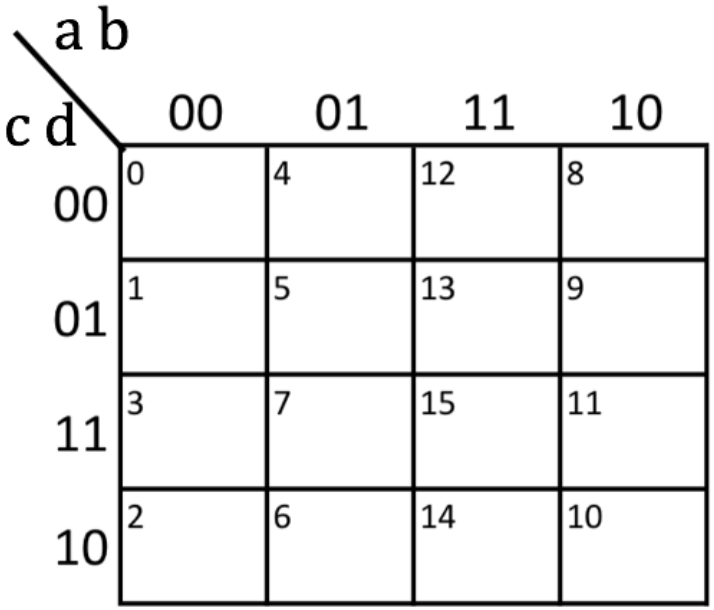
Una implicada prima se dice que es **esencial** si cubre algún maxtérmino no incluido en ninguna otra implicada prima. Al maxtérmino se le denomina **distinguido**. (Se denomina "esencial" porque se escoge siempre en el cubrimiento mínimo.

$$F = \Pi(0,1,2,4,5,6,12,13)$$



Ejemplos de obtención de la expresión ps mín.

$f(a,b,c,d) = \Pi(0,1,2,4,5,6,12,13)$



Método para la obtención del cubrimiento mínimo usando K-mapa **(paso 3 del proceso de diseño)**

OBJETIVO: cubrir todos los términos con el menor número de implicantes, lo más grandes posible

1. Elementos aislados (que no se puedan agrupar con otros)

2. Elementos distinguidos (que sólo pertezcan a una implicante/implicada prima), seleccionando las implicantes/implicadas primas esenciales correspondientes

3. Si no quedan elementos distinguidos, el resto de elementos se cubren "a ojo" con el criterio de "el menor número de agrupaciones y lo más grandes posible

4. La expresión algebraica reducida se obtendrá:

- **SP mínima: sumando las implicantes seleccionadas en el cubrimiento mínimo**
- **PS mínimo: multiplicando las implicadas seleccionadas en el cubrimiento mínimo**

$$si\ x = 0 \xrightarrow{\text{"da lugar a..."}} \cdot \bar{x}$$
$$si\ x = 1 \xrightarrow{\text{"da lugar a..."}} \cdot x$$

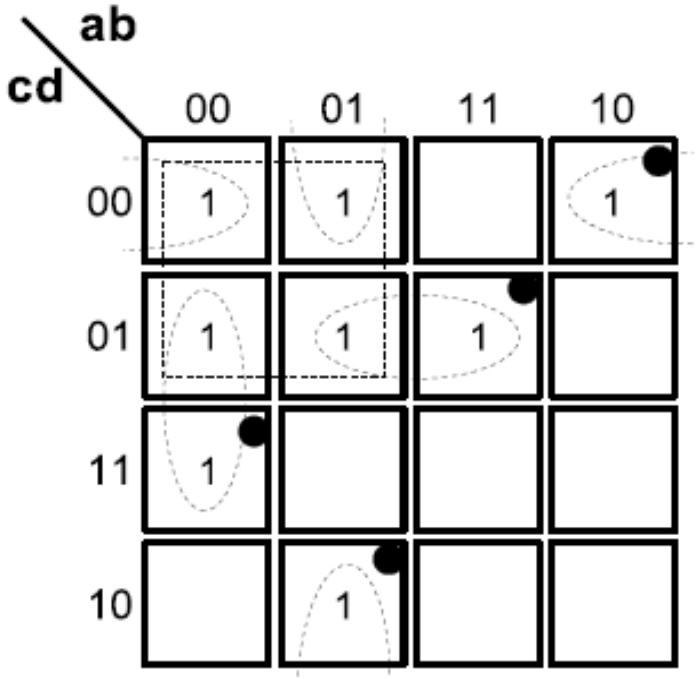
Ejemplo de obtención de la expresión sp mín.

$$f(a,b,c,d) = \sum(0,1,3,4,5,6,8,13)$$

c d \ a b	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Ejemplos de obtención de la expresión mínima

Ejercicio 1.- f = ∑(0,1,3,4,5,6,8,13)



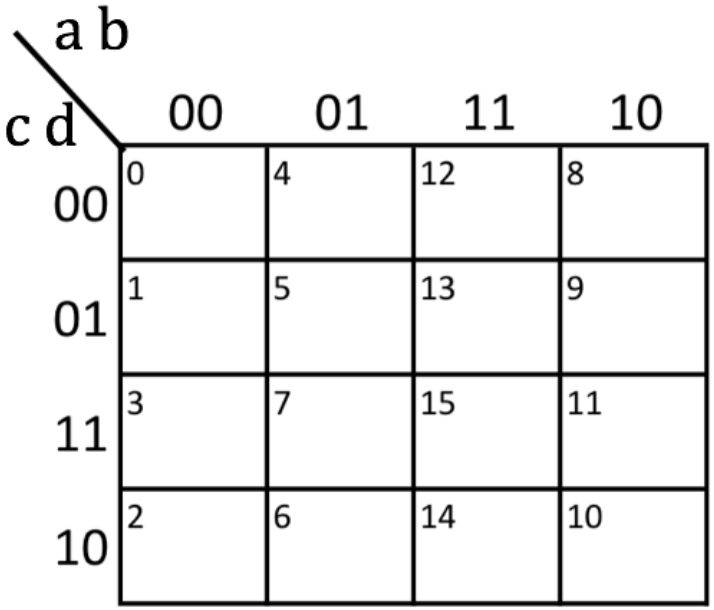
Buscamos las Ip esenciales

F(a,b,c,d) = a'b'd + b'c'd'+a'bd'+bc'd

Fin. Se han cubierto todos los mintérminos sólo con las IP esenciales

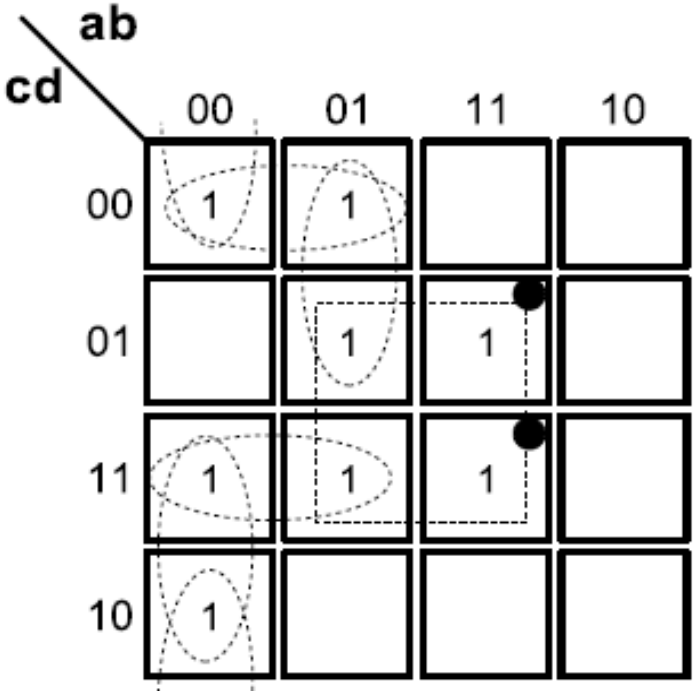
Ejemplos de obtención de la expresión mínima

$f(a,b,c,d) = \sum(0,2,3,4,5,7,13,15)$



Ejemplos de obtención de la expresión mínima

Ejercicio 2.- f = ∑(0,2,3,4,5,7,13,15)



1)Buscamos las Ip esenciales : **bd** con la que no se cubren todos los mintérminos

2)Nos fijamos en el mintérmino 4. Está cubierto por las Ip **a'c'd'** y **a'bd**. Ambas del mismo coste pero la primera cubre también al mintérmino 0 que no estaba cubierto por **bd**. Escogemos **a'c'd'**

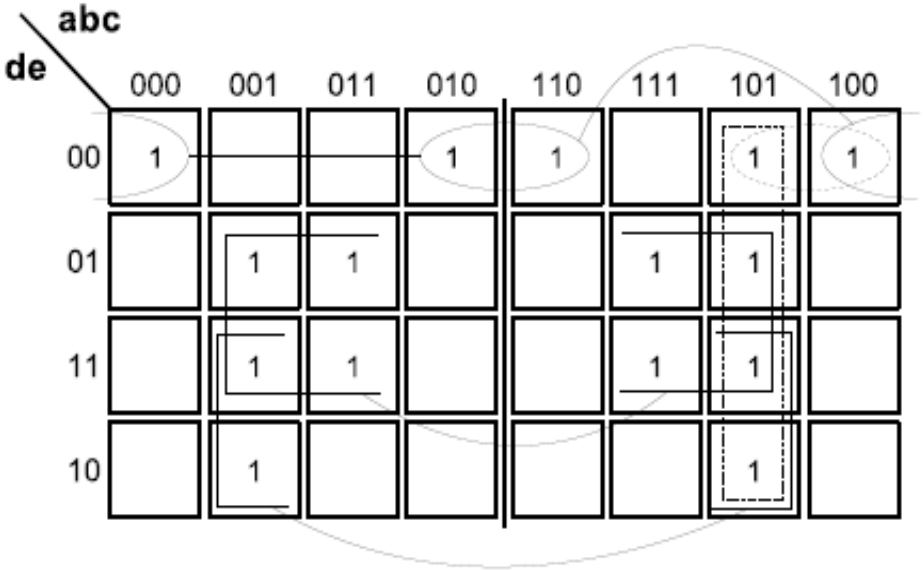
3) Ahora sólo quedan por cubrir los mintérminos 2 y3. Obviamente escogemos **a'b'c**

f = bd + a'c'd' + a'b'c

Fin. Se han cubierto todos los mintérminos

Ejemplos de obtención de la expresión mínima

Ejercicio 5.- f = ∑(0,5,6,7,8,13,15,16,20,21,22,23,24,29,31)



1) La obtención de las implicantes primas en un K-mapa de 5 variables requiere analizar las simetrías entre los sub K-mapa de 4 para cuando la variable más significativa (en este ejemplo es **a**) vale 0 y cuando vale 1.

2) Siga los pasos presentados en las transparencias anteriores.

f = c'd'e' + c e + b' c d + a b'c

NOVEDAD 24/25: NO VAMOS A EXIGIR cubrimiento mínimo de KMAPAS de 5 variables (Aunque sí saber sacar el término suma o producto que corresponde a una implicante)

Índice

Diseño con K-mapa

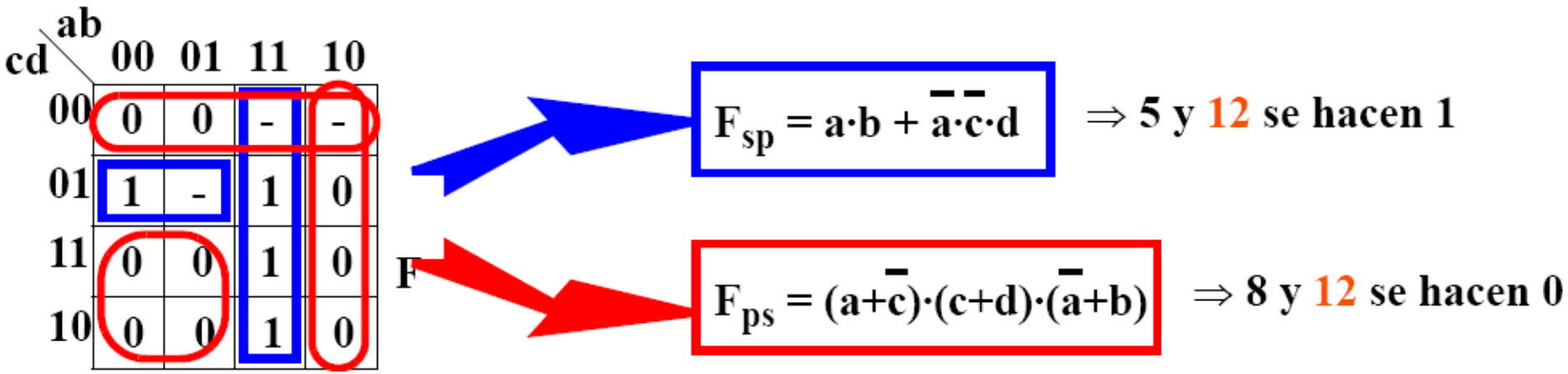
Funciones incompletamente especificadas

Las casillas con inespecificación se usan como mejor nos convenga:

Se pueden incluir para formar grupos mayores.

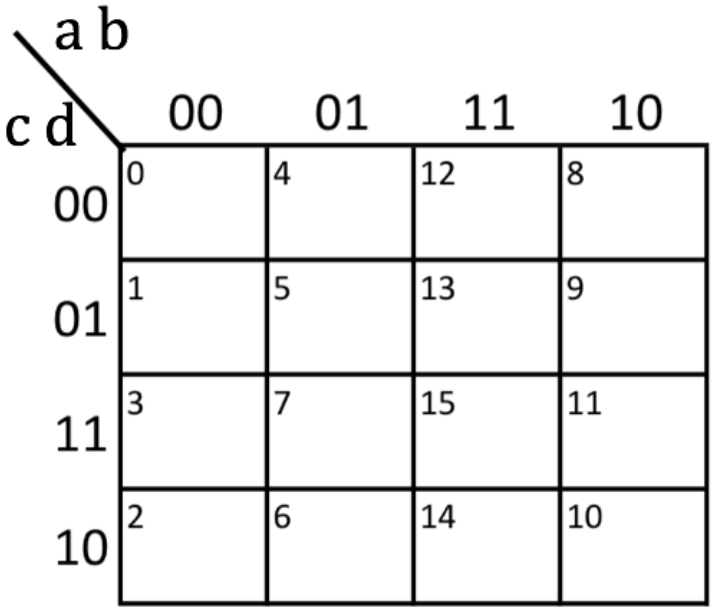
No es necesario cubrirlas todas.

Ejemplo: $F = \Sigma\,(1, 13, 14, 15) + d(5, 8, 12)$



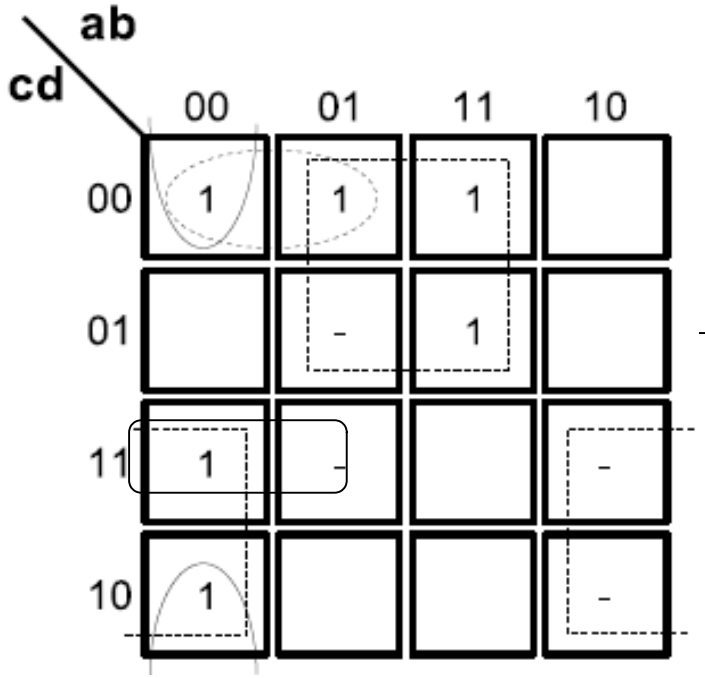
Ejemplos de obtención de la expresión mínima

$$f(a,b,c,d) = \sum(1,13,14,15) + d(5,8,12)$$



Ejemplos de obtención de la expresión mínima

Ejercicio 4.- f = ∑(0,2,3,4,12,13) + d(5,7,10,11)



1) Buscamos las Ip esenciales usando inespecificaciones como mintérminos y rechazando aquellas formadas sólo por inespecificaciones: **bc'**

2) Nos fijamos en el mintérmino 3, cubierto por **a'cd** y **b'c**. La mejor opción es **b'c.**

3) Sólo queda por cubrir el mintérmino 0. Hay dos posibilidades: las implicantes **a'b'd'** y **a'c'd'**. Cualquier opción es válida.

4) No se busca cubrimiento de inespecificaciones

f = bc' + a'b'd' + b'c

Fin. Se han cubierto todos los mintérminos

Ejemplos de obtención de la expresión mínima

$$f(a,b,c,d) = \sum(0,2\ 3,4,12,13) + d(5,7,10,11)$$

c d \ a b	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

NEW 24/25

Indice

1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

NEW 24/25

¿Qué es Verilog?

- **Verilog** es un lenguaje de descripción de hardware (HDL) que, como su nombre indica, permite describir formalmente el funcionamiento de un circuito
- Con ayuda de herramientas como **ISE Design Suite (Xilinx) o Vivado,** podemos **describi**r circuitos en Verilog, **simular** su funcionamiento e incluso **implementar** el circuito hardware
- Es similar a un lenguaje de programación imperativo: formado por un conjunto de sentencias que indican como realizar una tarea.
- Algunas diferencias:

- La mayoría de las sentencias se ejecutan concurrentemente
- Cada sentencia corresponde a un bloque de circuito

NEW 24/25

Estructura general de una descripción Verilog

Un diseño Verilog consta de uno o varios módulos interconectados

La descripción del módulo Verilog (module) consiste en:

```
module mi_circuito (
   input x, y,
   input z,
   output f1, f2
   );

   wire cable_interno;
   reg  variable_a;

   ...
   ...
   ...

endmodule
```

Declaración del módulo con sus entradas y salidas

Declaración de señales y variables que se utilizarán internamente en la descripción

Descripción del comportamiento módulo. Hay varias alternativas para realizarla

NEW 24/25

Tipos de descripciones

- Descripción **funcional (assign)**:
  - a partir de las expresiones algebraicas de las funciones (operadores AND, OR, XOR…)
  - previamente tenemos que usar kmapa y deducir expresiones algebraicas
  - para cada salida, una asignación (son concurrentes)

- Descripción **estructural**:
  - Interconectando de los componentes de un circuito previamente diseñado.
  - Por ejemplo, a partir del diseño a nivel de puertas lógicas que previamente hemos diseñado ”a mano”

- Descripción **procedimental**:
  - Descripción algorítmica utilizando estructuras de control (if… else, case…)
  - Es la opción más potente (tanto para circuitos combinacionales como secuenciales)

NEW 24/25

Descripción funcional con operadores algebraicos (assign)

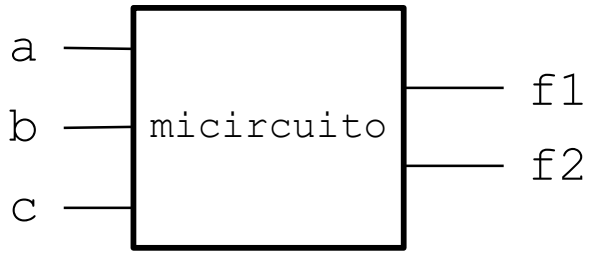
Modela circuitos combinaciones.

Asignamos (**assign**) funciones lógicas a las salidas de cada módulo

Las salidas dependen de los valores de las entradas en cada instante (combinacional).

Necesitamos una sentencia **assign** para cada función de salida

Todas las sentencias assign se ejecutan al mismo tiempo, por lo que da igual el orden en el que aparezcan en la especificación.



f1= ab + ac
f2= a'b + bc

```
module micircuito(
    input a,b,c,
    output f1,f2);

  assign f1 = (a&b) | (a&c);
  assign f2 = (~a&b) | (b&c);

endmodule
```

NEW 24/25

Algunos operadores a nivel de bits (bitwise) (assign)

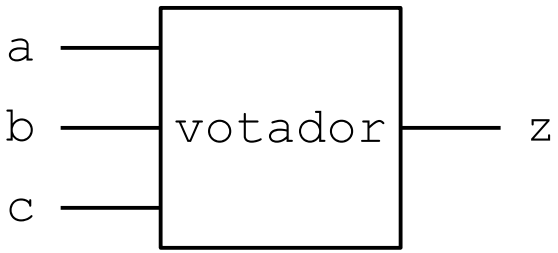
Operador	Ejemplo de código Verilog
&	`c = a&b; // Operación AND de todos los bits`
|	`c = a|b; // Operación OR de todos los bits`
^	`c = a^b; // Operación XOR de todos los bits`
~	`b = ~a; // NOT Inversión de todos los bits`
~&	`d = a ~& b; // Operador NAND a nivel de bits`
~|	`d = a ~| b; // Operador NOR a nivel de bits`
~^	`d = a ~^ b; // Operador EXNOR a nivel de bits`

- Estos operadores trabajan con todos los bits de a y todos los bits de b

- Si las variables son de un bit operan como los operadores del álgebra de conmutación.

NEW 24/25

Ejemplo: circuito votador



- Expresión lógica:
  - z=ab+ac+bc

```
module votador (input a,b,c,output z);

  assign z= (a & b) | (a & c) | (b & c);

endmodule
```

NEW 24/25

Descripción estructural

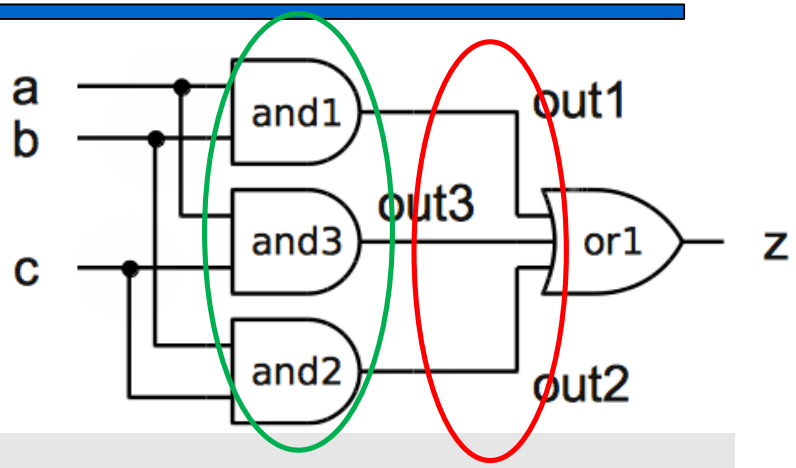
- Permite conectar módulos que ya están definidos previamente

- Las puertas lógicas básicas ya están predefinidas en Verilog

and, or, nand, nor, xor, xnor, not

- Las conexiones internas hay que declararlas tipo "**wire**" (cable)

Es muy útil para la interconexión de los módulos más complejos ya creados




```
module votador(
  input a,b,c,
  output z);

  wire out1,out2,out3;

  and and1(out1,a,b);
  and and2(out2,b,c);
  and and3(out3,a,c);
  or or1(z,out1,out2,out3);

endmodule
```

"Puertas tipo"

nombre de la puerta (instancia de una puerta tipo)

"cables de conexión"

NEW 24/25

Indice

1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

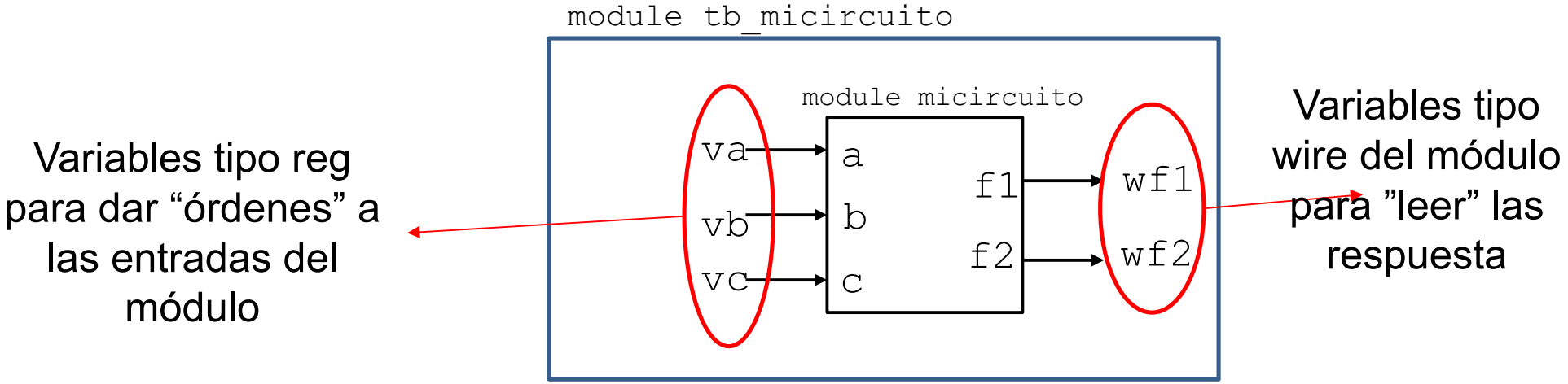
NEW 24/25

Simulación de módulos combinacionales

Vivado incluye la herramienta **ISim**, para simular el funcionamiento de nuestros diseños antes de implementarlo físicamente

Crearemos un fichero de simulación (**PROYECT MANAGER-> Add Sources -> Add or Create Simulation Sources)** lo asociamos al módulo diseñado e incluimos las órdenes para comprobar que el diseño funciona como hemos pensado

Proceso: **PROYECT MANAGER-> SIMULATION -> Run Simulation**



NEW 24/25

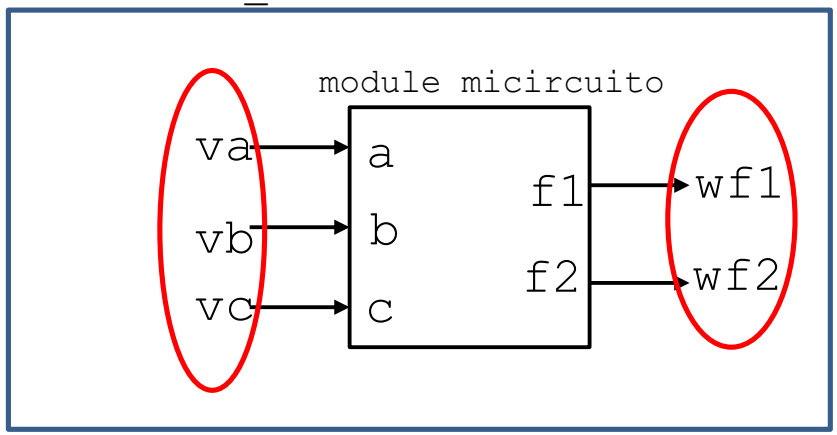
Ejemplo: fichero testbench

```
module micircuito(
    input a,b,c,
    output f1,f2);

  assign f1 = (a&b) | (a&c;)
  assign f2 = (~a&b) | (b&c);

endmodule
```

module tb_micircuito




```
module tb_micircuito;
  //inputs
  reg va,vb,vc;
  //outputs
  wire wf1,wf2;
micircuito uut(
  .a(va),.b(vb),.c(vc),
  .f1(wf1),.f2(wf2)
);
initial begin
  //Initialize Inputs
  va=0;
  vb=0;
  vc=0;
  #100;

  //Add stimulus here
  //****** COMPLETAR ******

end
endmodule
```

NEW 24/25

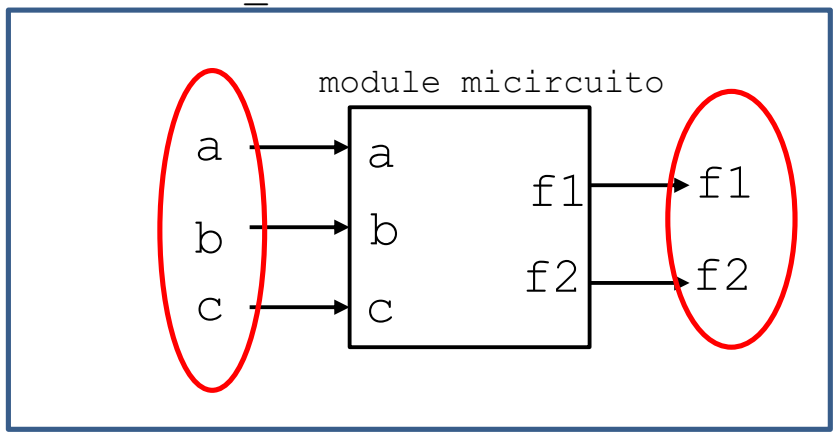
Las variables reg y wire pueden llamarse igual que las entradas y salidas del módulo a probar

```
module micircuito(
    input a,b,c,
    output f1,f2);

  assign f1 = (a&b) | (a&c;)
  assign f2 = (~a&b) | (b&c);

endmodule
```

module tb_micircuito




```
module tb_micircuito;
  //inputs
  reg a,b,c;
  //outputs
  wire f1,f2;
micircuito uut(
  .a(a),.b(b),.c(c),
  .f1(f1),.f2(f2)
);
initial begin
  //Initialize Inputs
  a=0;
  b=0;
  c=0;
  #100;

  //Add stimulus here
  //****** COMPLETAR ******

end
endmodule
```

NEW 24/25

Simulación de módulos combinacionales (ii)

Un fichero de pruebas (testbench) contiene siempre un bloque **initial**, que define los valores iniciales de las variables, y cómo queremos que cambien con el tiempo. Por ejemplo:

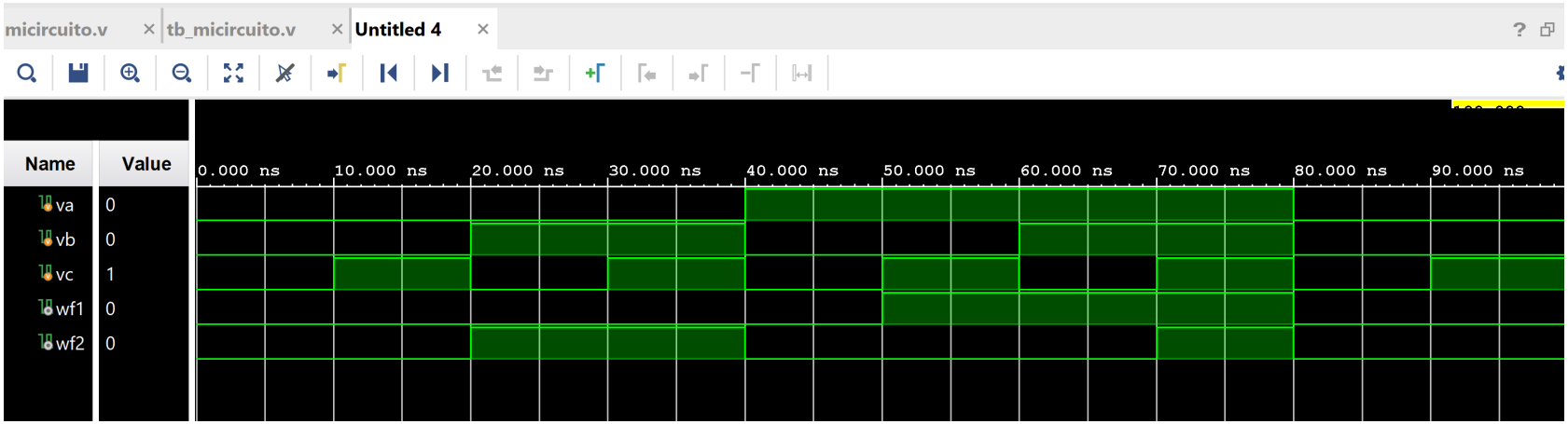
```
//Initial Inputs
initial begin
    va=0; //puedo definir, uno a uno, los valores iniciales
    vb=0; //no importa el orden, son valores iniciales
    vc=0;
    #10; //espero 10 ns
    vc=1; //a y b no cambian
    #10; //espero 10 ns
    {va,vb,vc}=3'b010; //puedo asignar valores a los 3 con {}
    vb=0; //no importa el orden, son valores iniciales
    vc=0;
    #50; //espero 10 ns
    $finish; fin de la simulación
end
```

NEW 24/25

Simulación de módulos combinacionales (iii)

Si quiero probar las 2^n combinaciones, es muy útil el bloque **always** que se ejecuta en paralelo al bloque initial

```
always #10 {va,vb,vc} ={va,vb,vc} + 1; //se generan 000, 001, 010…. 111
                                       //cambiando de valor cada 10ns
//Initial Inputs
initial begin
    va=0; //puedo definir, uno a uno, los valores iniciales
    vb=0; //no importa el orden, son valores iniciales
    vc=0;
    #100; //espero 100 ns
$finish; //fin de la simulación
end
```



Indice

1. Análisis lógico de circuitos combinacionales
2. El proceso de diseño de CC
3. Simplificación de expresiones usando K-mapa
4. Funciones incompletamente especificadas
5. Puertas lógicas integradas (LAB)
6. Descripción de funciones combinacionales con Verilog
7. Simulación de circuitos combinacionales con Verilog
8. Análisis temporal. Azares.

Análisis de Circuitos Combinacionales

Análisis Temporal

Representa la evolución en el tiempo de las entradas y salidas del circuito. A esta representación temporal se la denomina **CRONOGRAMA**.

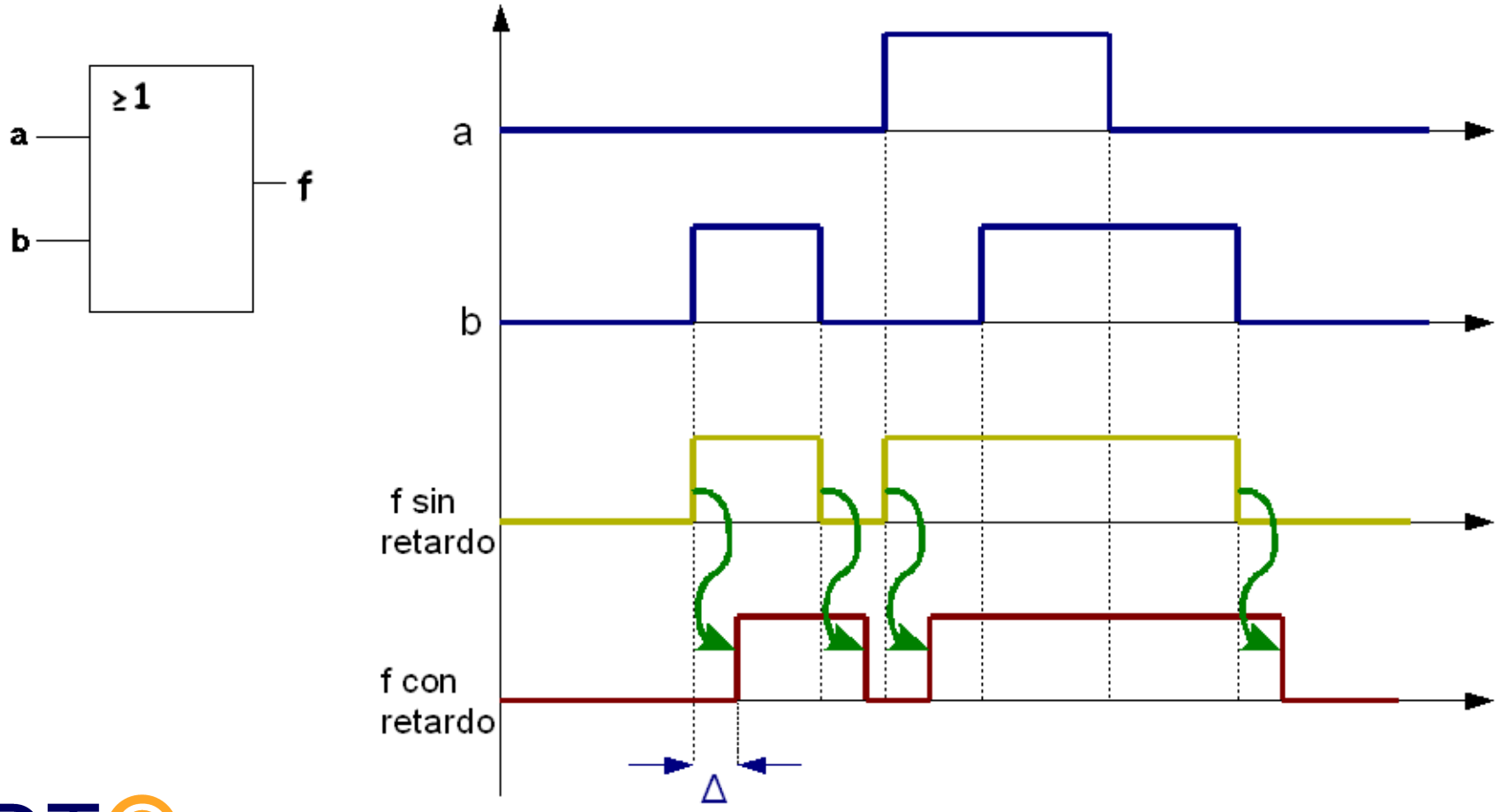
Dicha representación puede ser:

- Suponiendo que las puertas no tienen retrasos.
- Teniendo en cuenta los retrasos propios de las puertas lógicas.

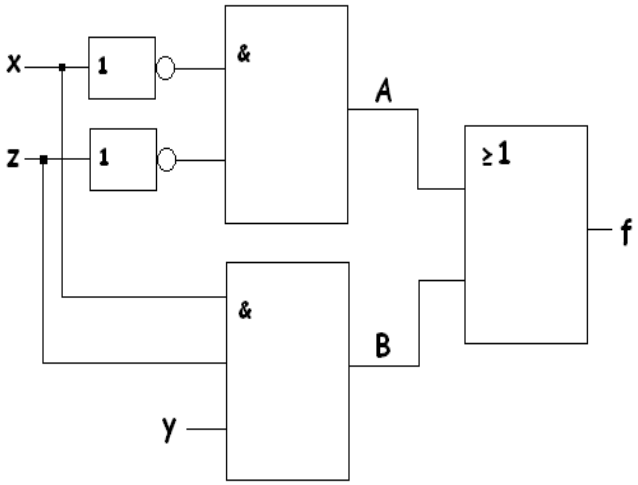
Análisis de Circuitos Combinacionales

Análisis Temporal

Para dibujar el cronograma considerando los retrasos, es necesario desplazar la salida de las puertas tanto como indique el valor del retraso.



Ejemplo de análisis temporal

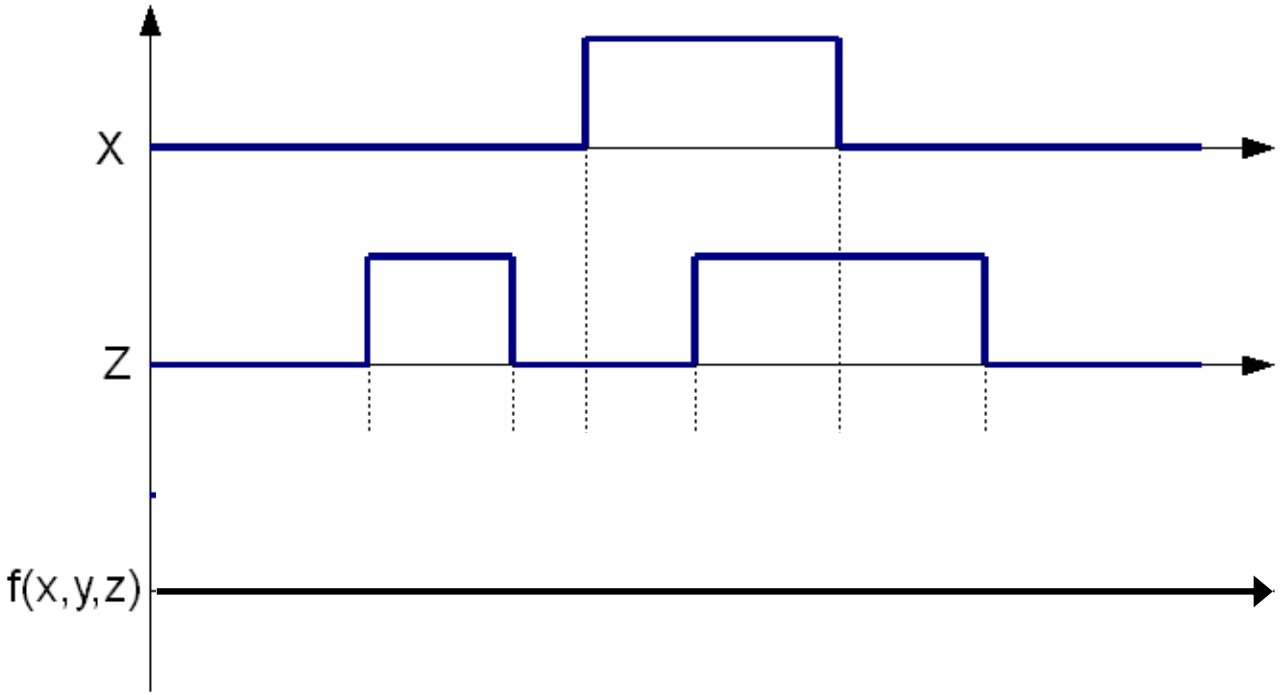


$$f = \bar{X}\bar{Z} + XYZ$$

Ejemplo

Cronograma de f (con y=1, y X,Z cambiando según la figura)

(sin considerar retrasos):



Análisis de Circuitos Combinacionales

Análisis Temporal

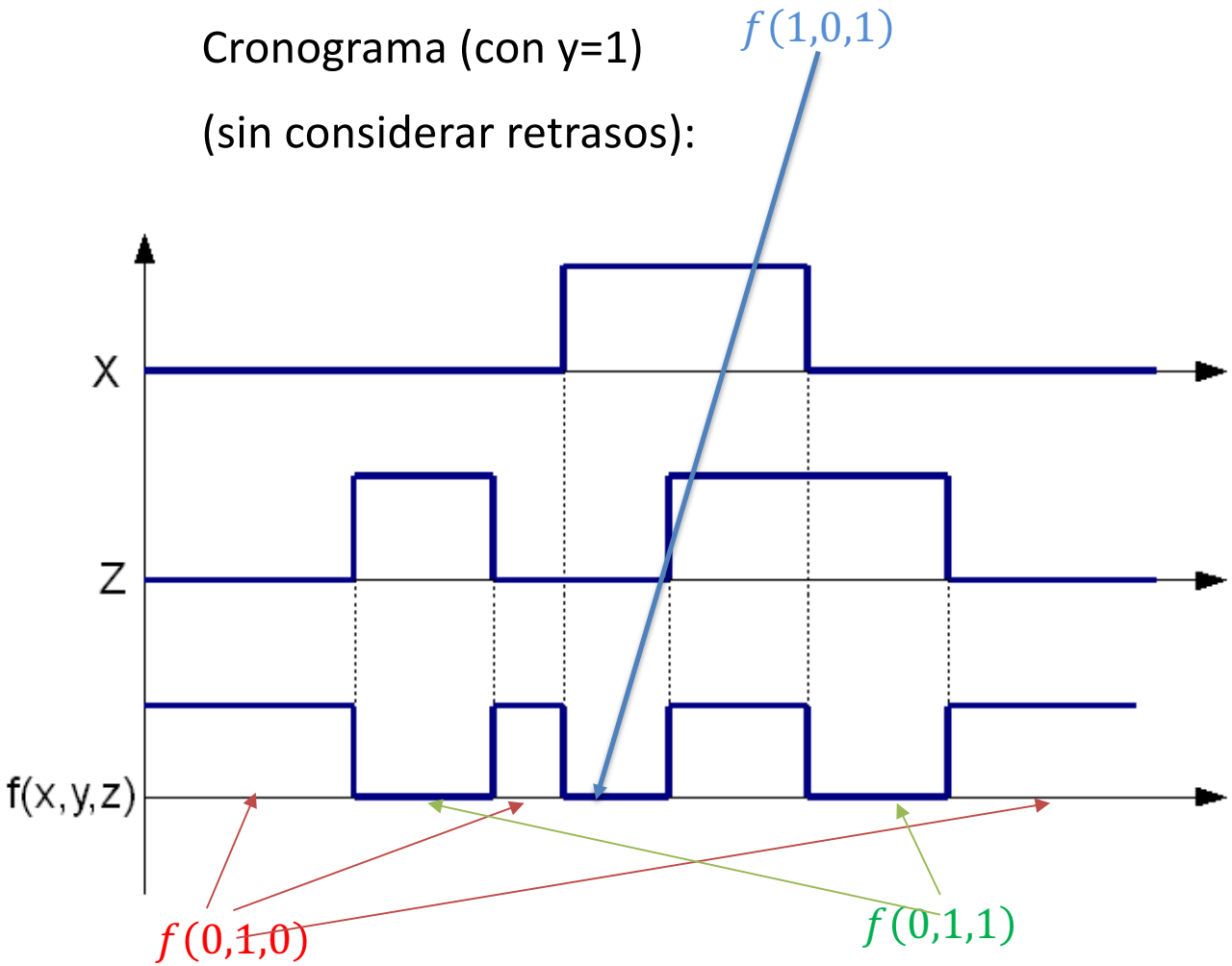
Ejemplo

Tabla:

xyz	f(x,y,z)
000	1
001	0
010	1
011	0
100	0
101	0
110	0
111	1

Cronograma (con y=1)

(sin considerar retrasos):

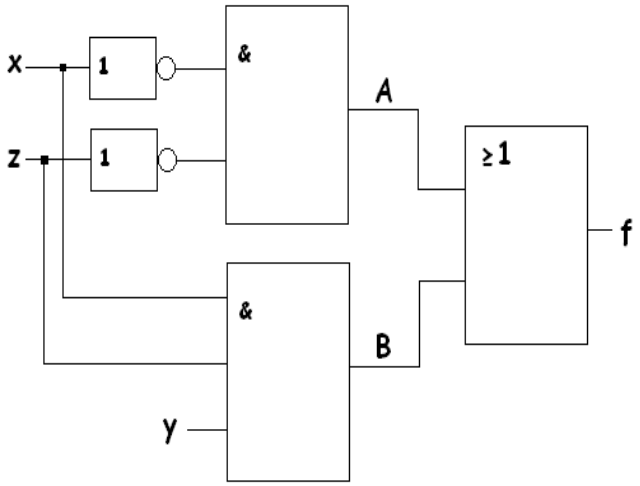


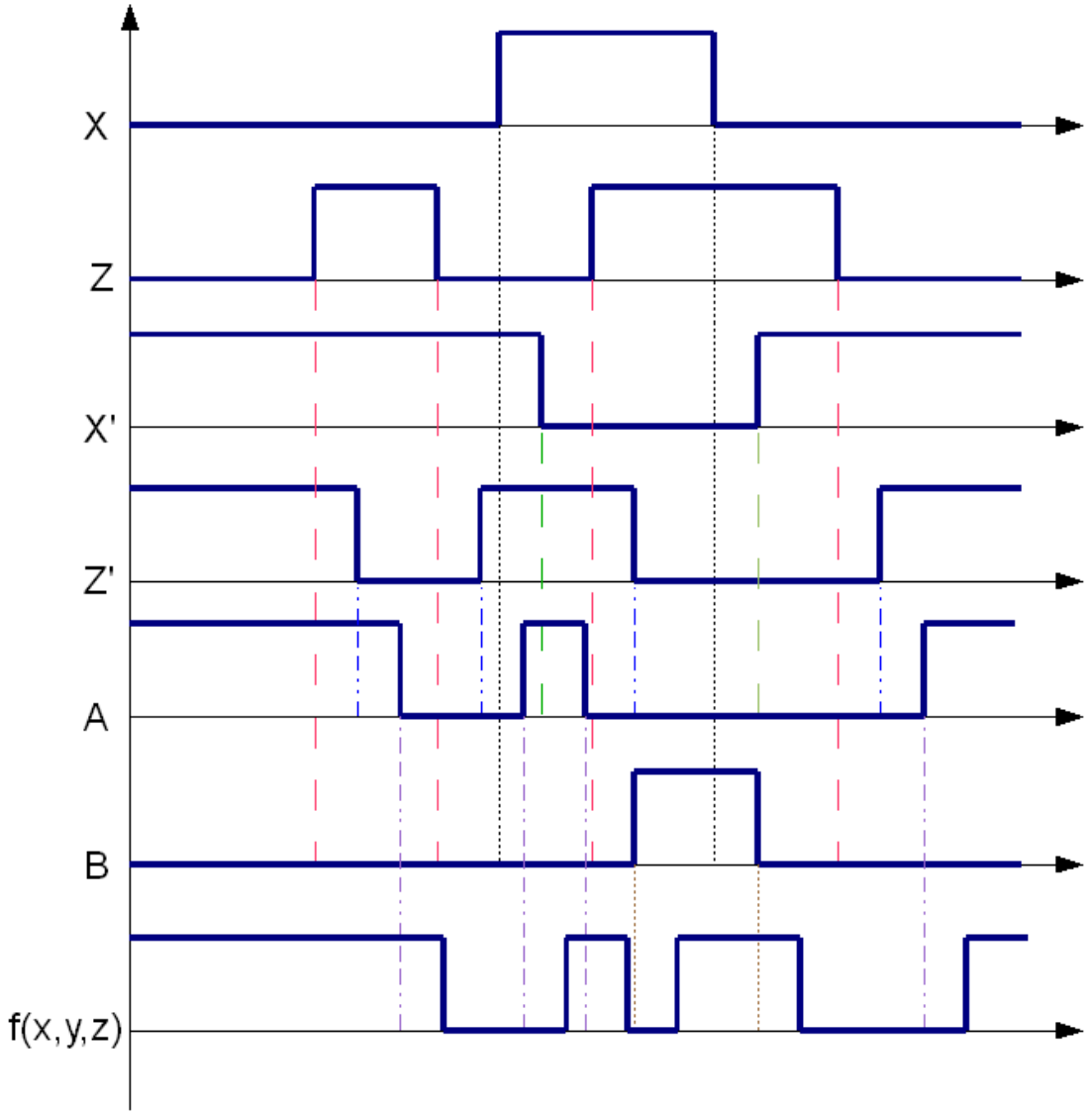
Análisis de Circuitos Combinacionales

Análisis Temporal

Cronograma (con y=1)

(con retrasos igual para todas las puertas)





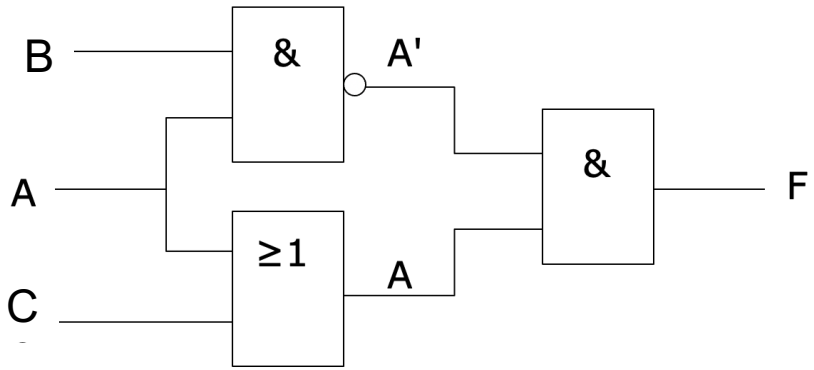
Análisis temporal: azares

Azares (o "glitches"): son pulsos inesperados que pueden aparecer en la salida de un circuito cuando las variables cambian con el tiempo.

Mucho cuidado si hay caminos con retrasos diferentes.

Ejemplo: Encontrar la forma de onda de F, si

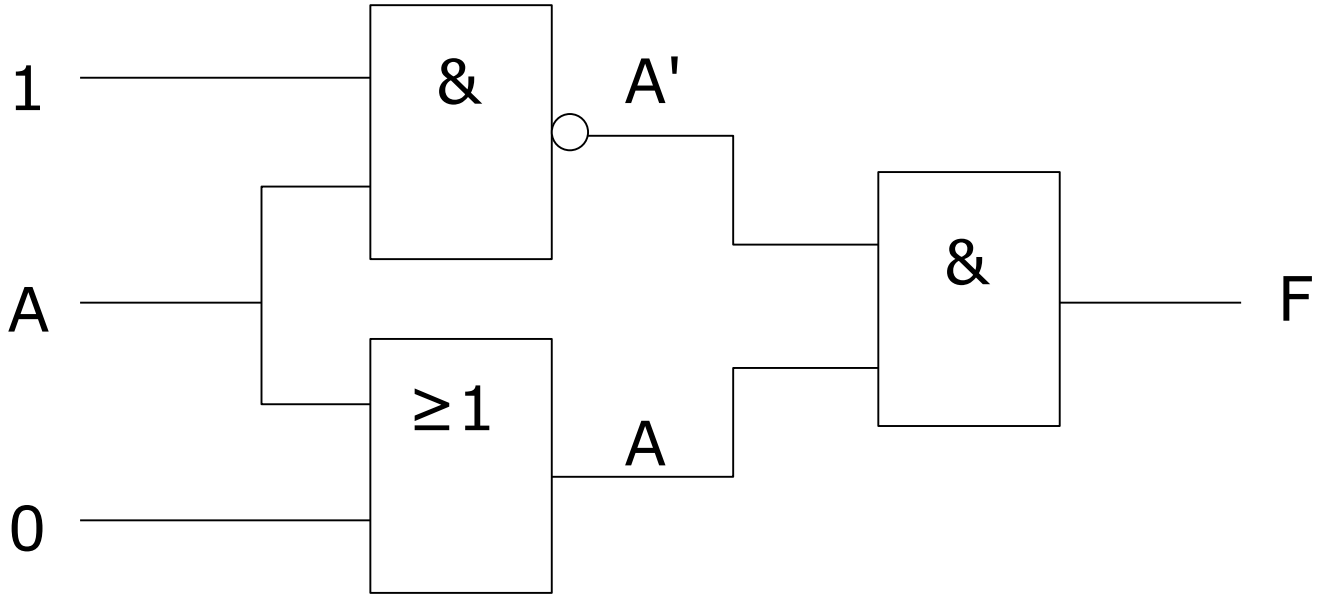
B=1, C=0,
A → cambia de 0 a 1



$$F(A,B,C) = \overline{(AB)}\,(A+C)$$

Análisis de Circuitos Combinacionales

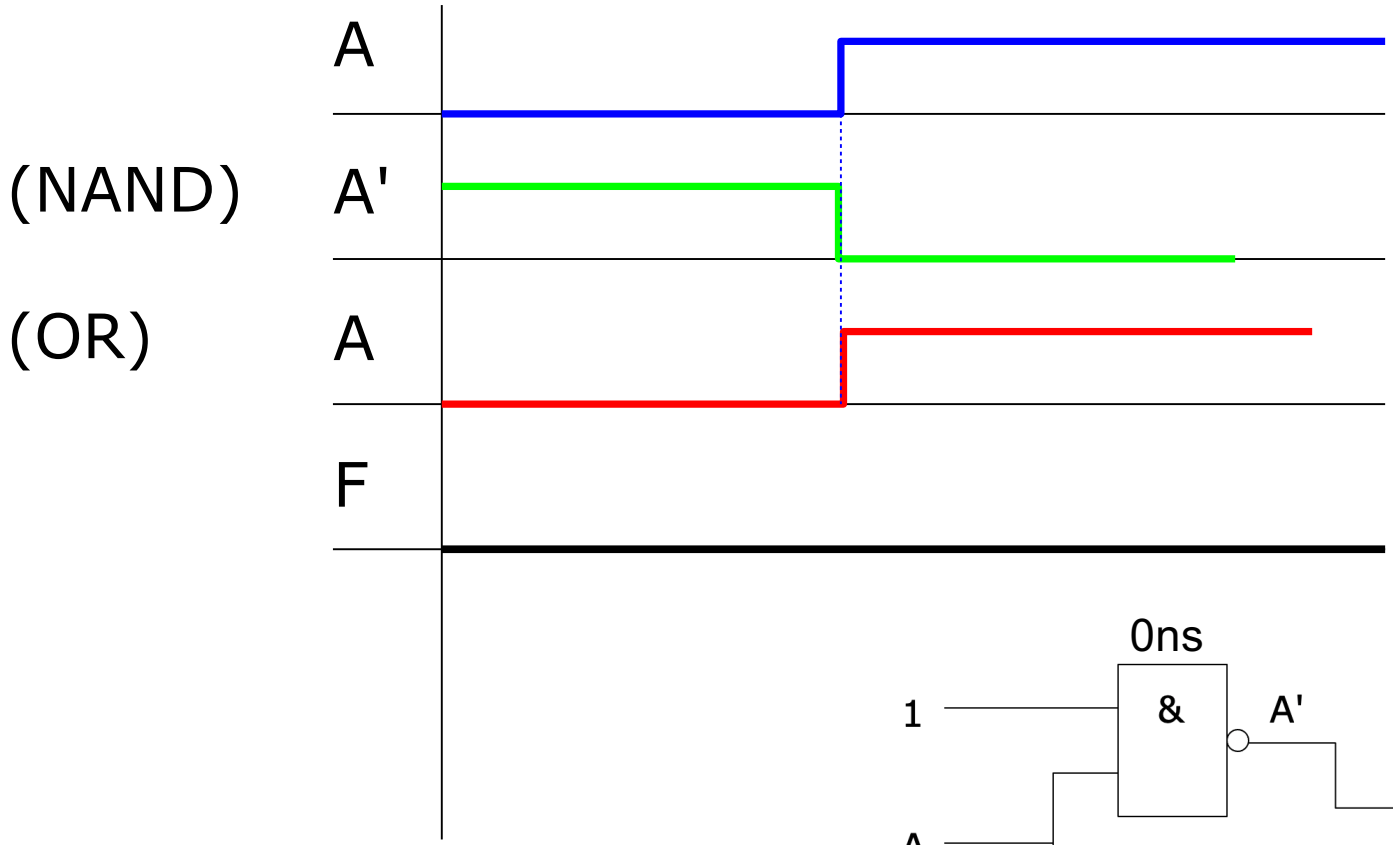
- Análisis lógico

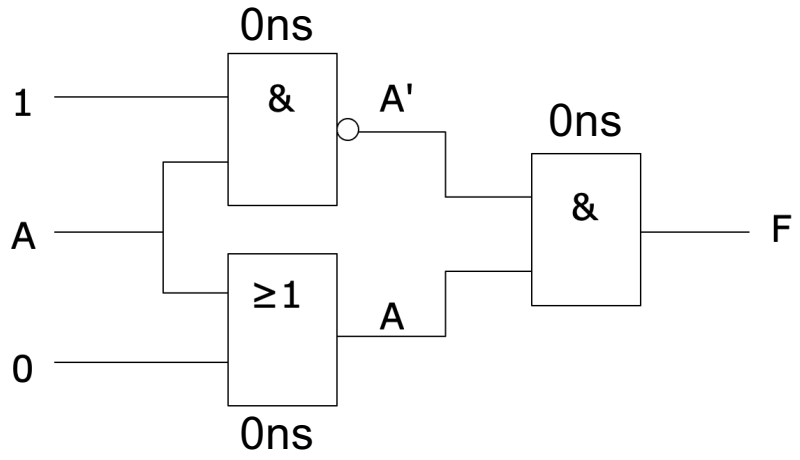


$$F = A' A = 0$$

Análisis temporal

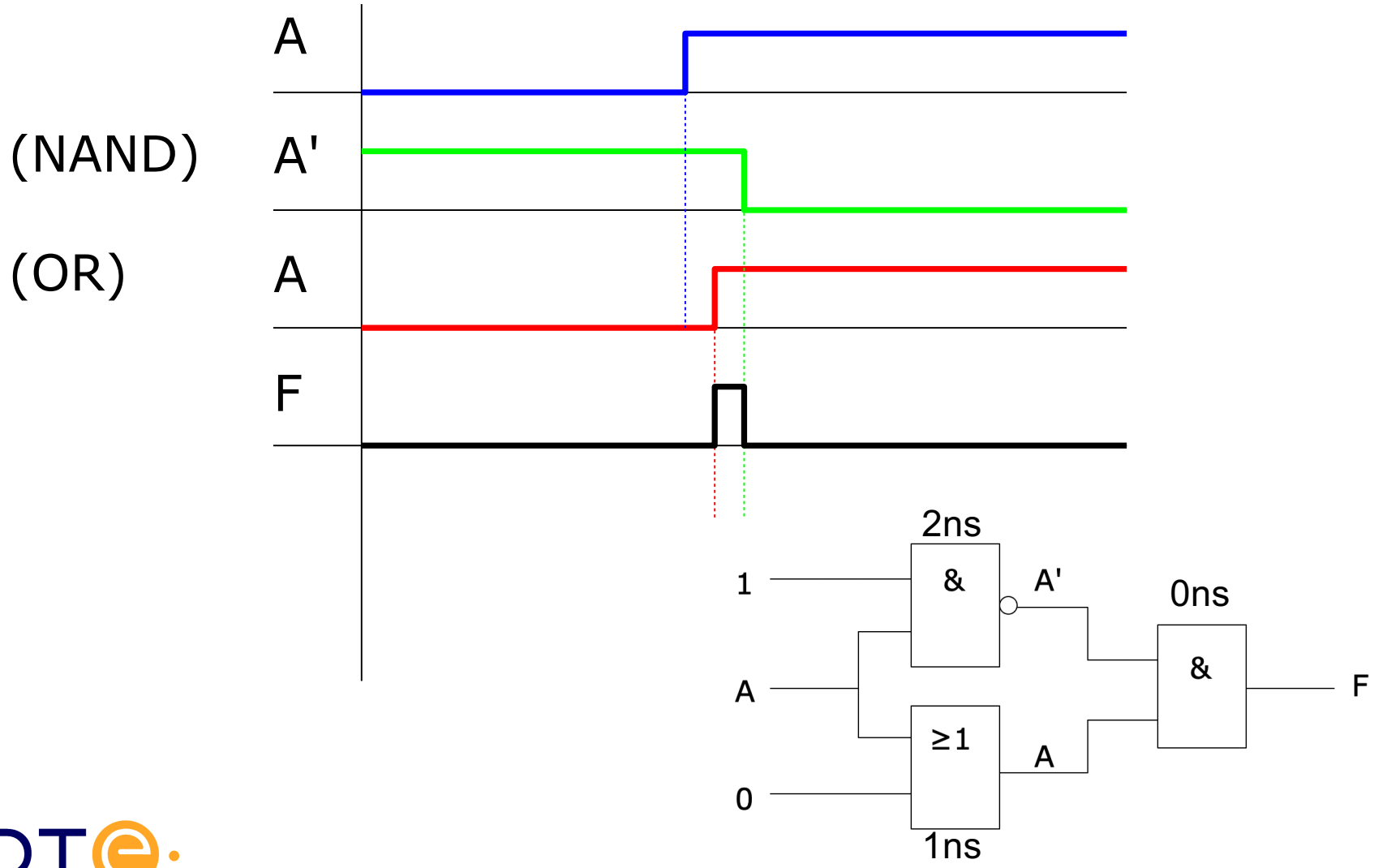
- Análisis temporal (ideal, sin considerar retrasos)





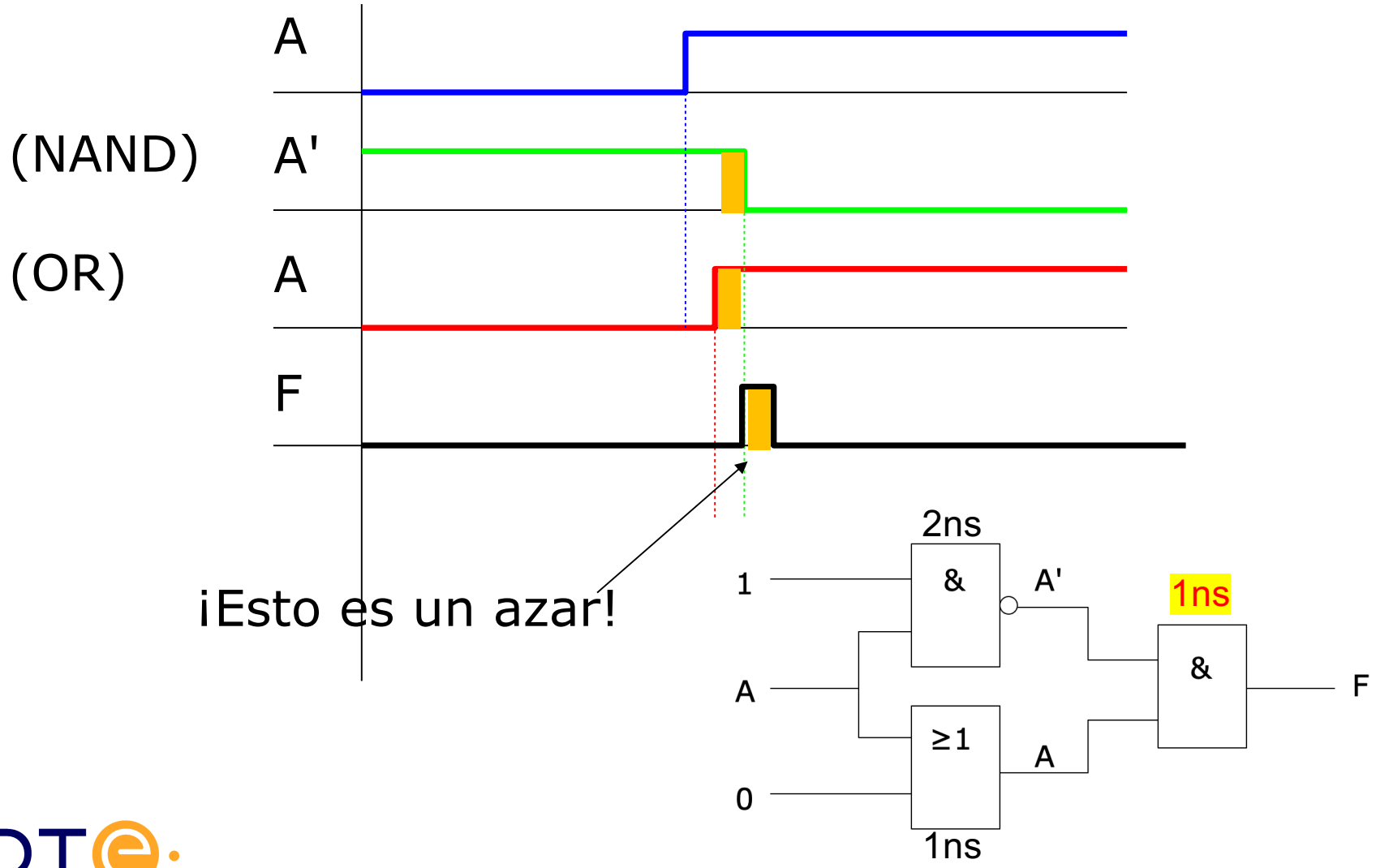
Análisis temporal: azares

- Azar en análisis temporal (considerando retrasos)



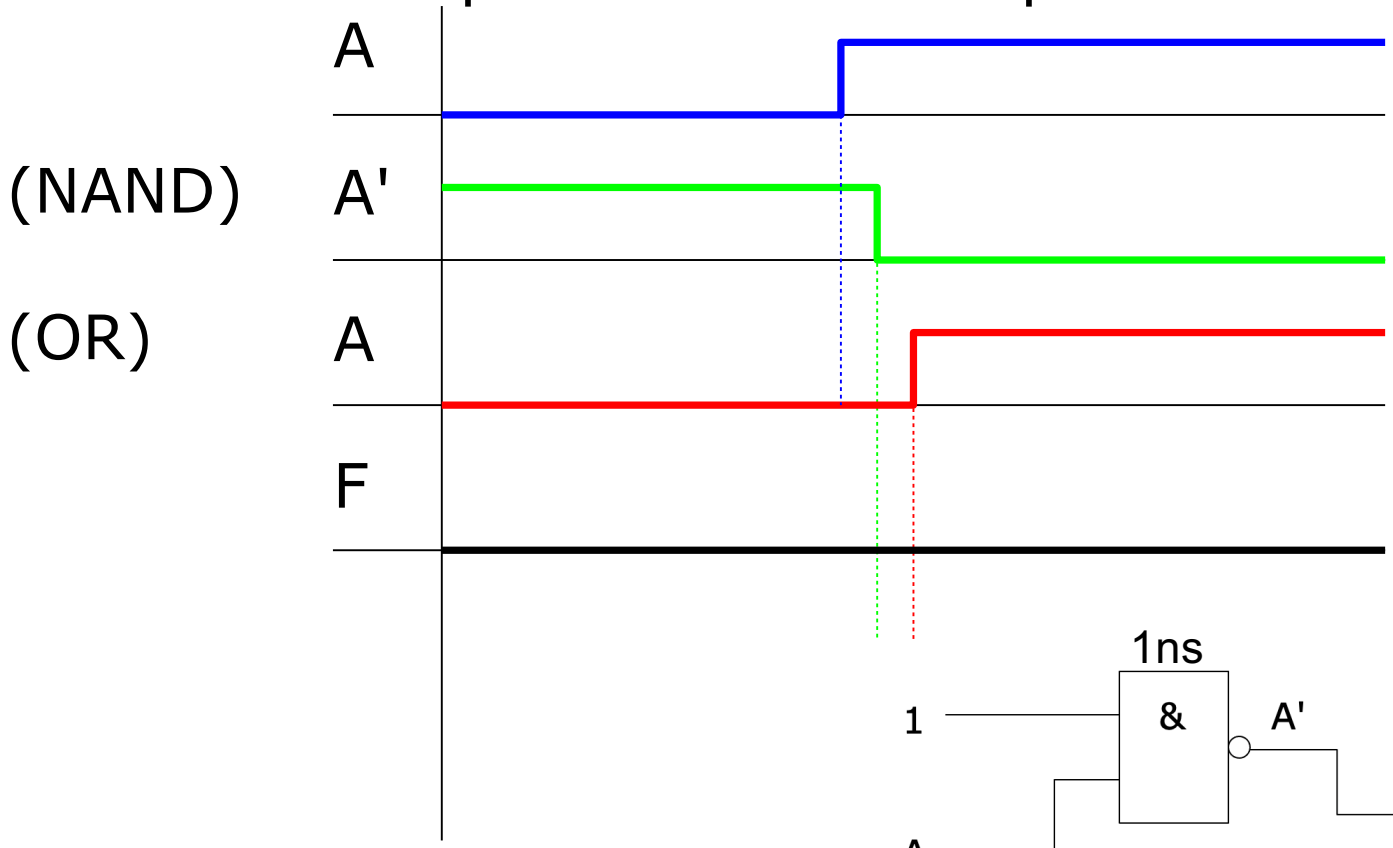
Análisis temporal: azares

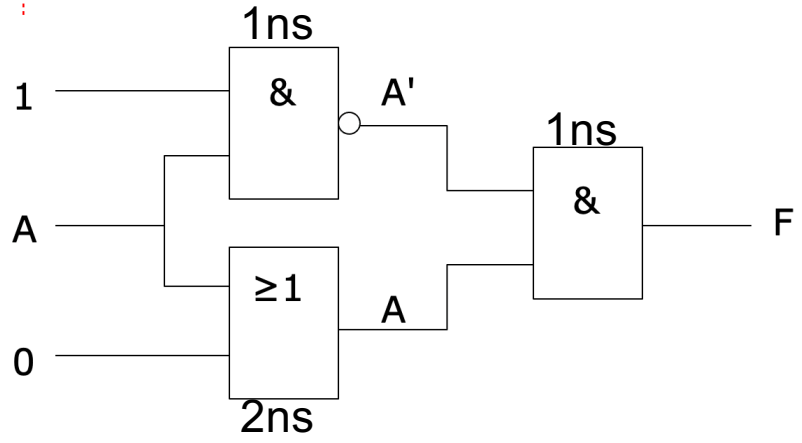
- Azar en análisis temporal (considerando retrasos)



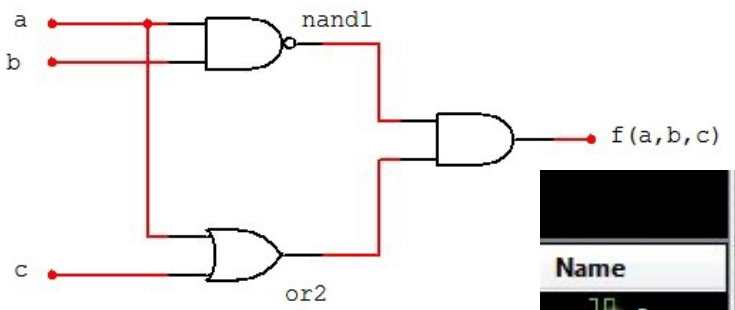
Análisis temporal sin azares

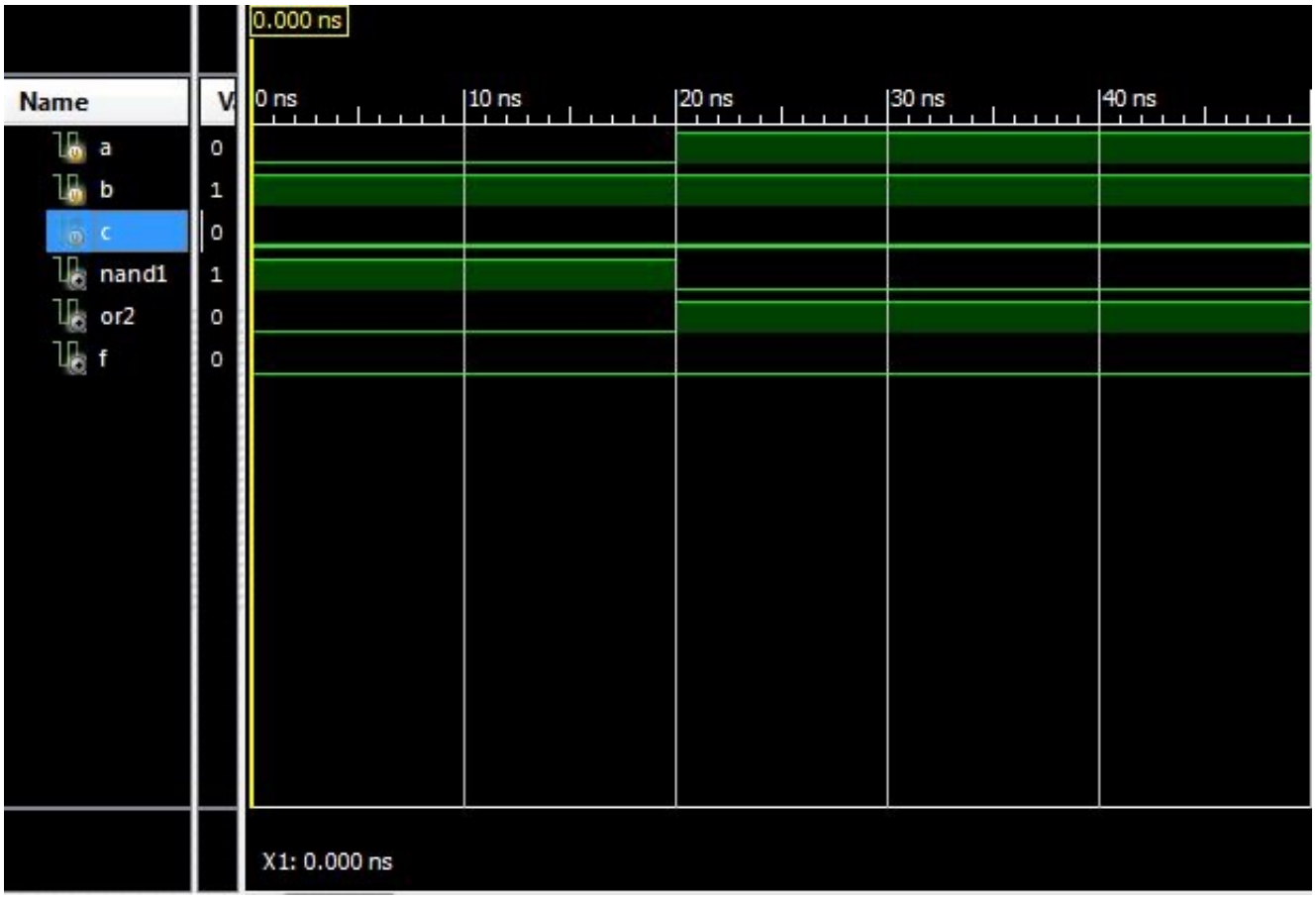
Ahora si el retraso de la puerta OR fuera mayor que el retraso de la puerta NAND no aparecería el AZAR



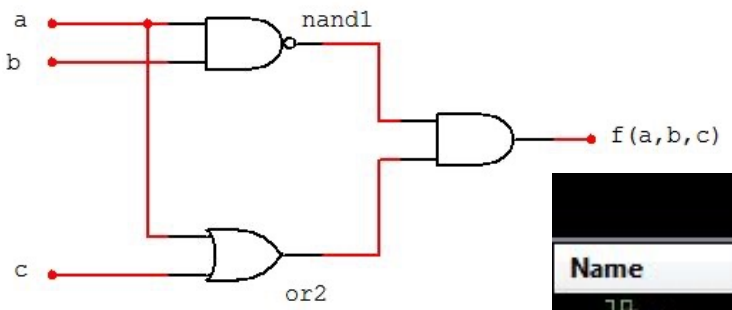


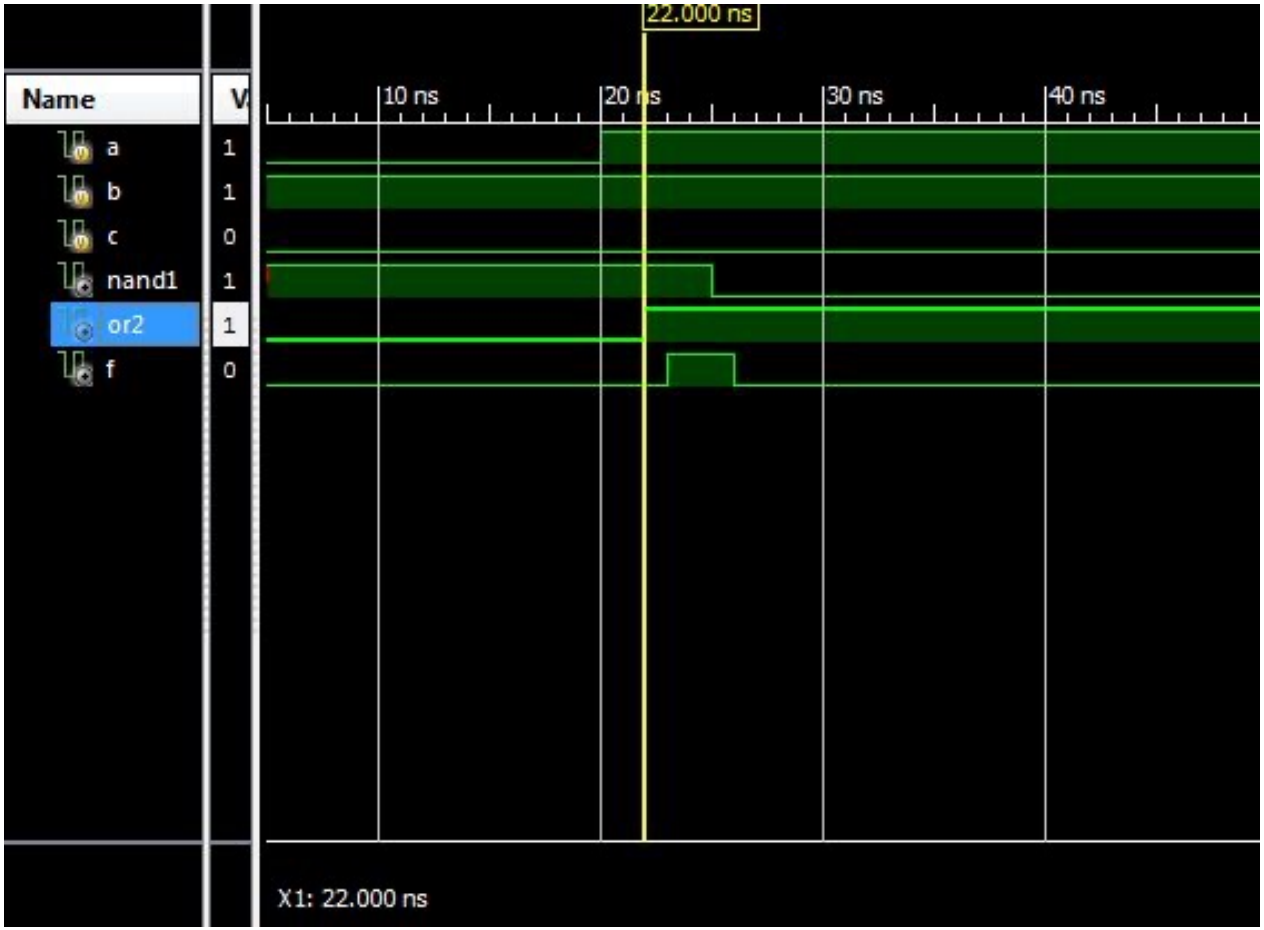
Ejemplo de análisis temporal usando un simulador lógico (XILINX Isim)





Ejemplo de análisis temporal usando un simulador lógico (XILINX Isim)





Ejemplo de análisis temporal usando un simulador lógico (XILINX Isim)