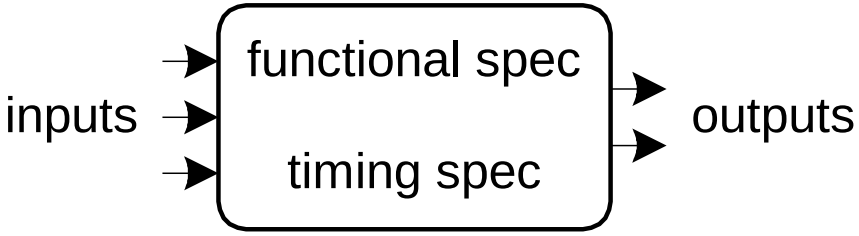
Lógica Combinacional en VHDL (I)

UNIVERSIDAD NEBRIJA

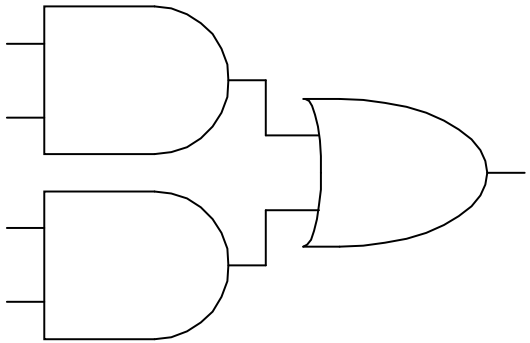
Recordamos: Tipos de Circuitos Lógicos

- **Lógica combinacional**
  - Sin memoria
  - Salidas determinadas por el valor actual de la entrada
- **Lógica Secuencial**
  - Con memoria
  - Salidas determinadas por los valores actuales y anteriores de las entradas.



Regla de composición para lógica combinacional

- Cada elemento es combinacional
- Cada nodo es o una entrada o conecta con exactamente una salida.
- El circuito no contiene rutas cíclicas
- **Ejemplo:**

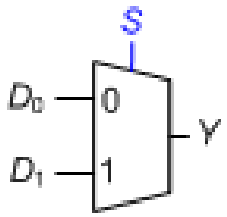


Algunos Bloques Combinacionales

- Multiplexores
- Decodificadores
  - 7-Segmentos

Multiplexor (Mux)

- Selecciona una de *N* entradas para conectarla a una salida
- $\log_2 N$-bit entradas de selección
- **Ejemplo:** **2:1 Mux**



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S	Y
0	D_0
1	D_1

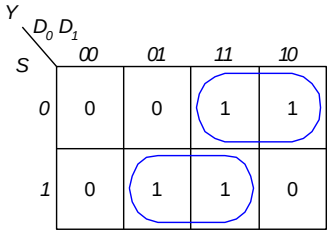
MUX en VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity mux2 is
  port(d0, d1: in  STD_LOGIC;
       s:      in  STD_LOGIC;
       y:      out STD_LOGIC);
end mux2;

architecture synth of mux2 is
begin
 pr: process (s, d0, d1)
  begin
    if s='1' then
       y <= d1;
    else
       y <= d0;
    end if;
  end process;
end synth;
```

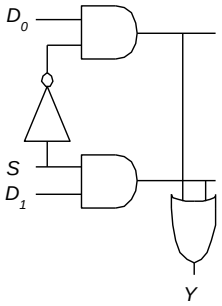
```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity mux4 is
  port(d0, d1,
       d2, d3: in  STD_LOGIC_VECTOR(3 downto 0);
       s:      in  STD_LOGIC_VECTOR(1 downto 0);
       y:      out STD_LOGIC_VECTOR(3 downto 0));
end;

architecture synthl of mux4 is
begin
  y <= d0 when s = "00" else
       d1 when s = "01" else
       d2 when s = "10" else
       d3;
end;
```

Implementación de multiplexores

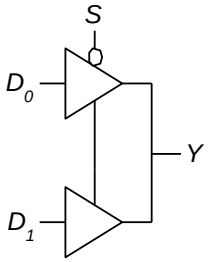
- **Con puertas lógicas**
  - Suma de productos



$Y = D_0\overline{S} + D_1S$



- **Buffers Triestado**
  - Para un mux de N entradas, usa N buffers triestado
  - Activa exactamente uno para seleccionar la entrada apropiada



Implementación VHDL

```
library IEEE; use
IEEE.STD_LOGIC_1164.all;

entity mux2 is
  port(d0, d1: in  STD_LOGIC;
       s:      in  STD_LOGIC;
       y:      out STD_LOGIC);
end mux2;

architecture synth of mux2 is
begin
   y <= (s='0' and d0='1')
         or (s='1' and d1='1');
end synth;
```

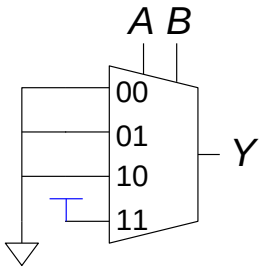
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux2 is
  port(d0, d1: in  STD_LOGIC_VECTOR(3 downto 0);
       s:      in  STD_LOGIC;
       y:      out STD_LOGIC_VECTOR(3 downto 0));
end mux2;

architecture struct of mux2 is
 signal sbar: STD_LOGIC;
begin
  sbar <= not s;
  t0: entity work.tristate
           port map(d0, sbar, y);
  t1: entity work.tristate
           port map(d1, s, y);
end struct;
```

Lógica mediante Multiplexores

- Usar el mux como una tabla de verdad

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

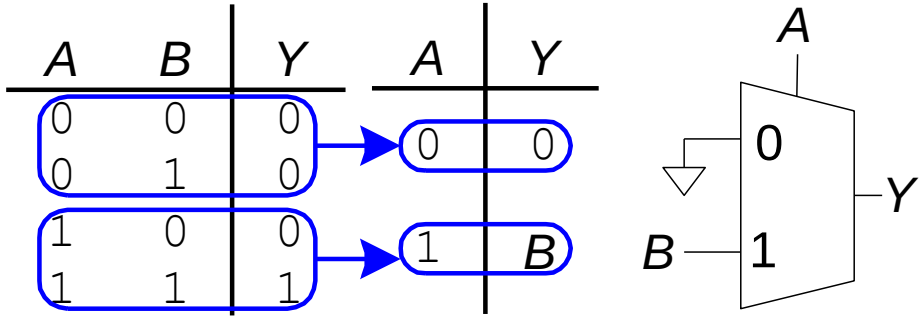
$Y = AB$



Lógica usando multiplexores

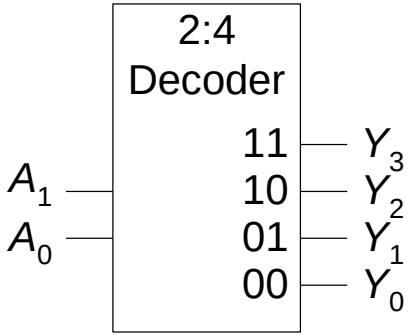
- Reduciendo el tamaño del MUX

$Y = AB$



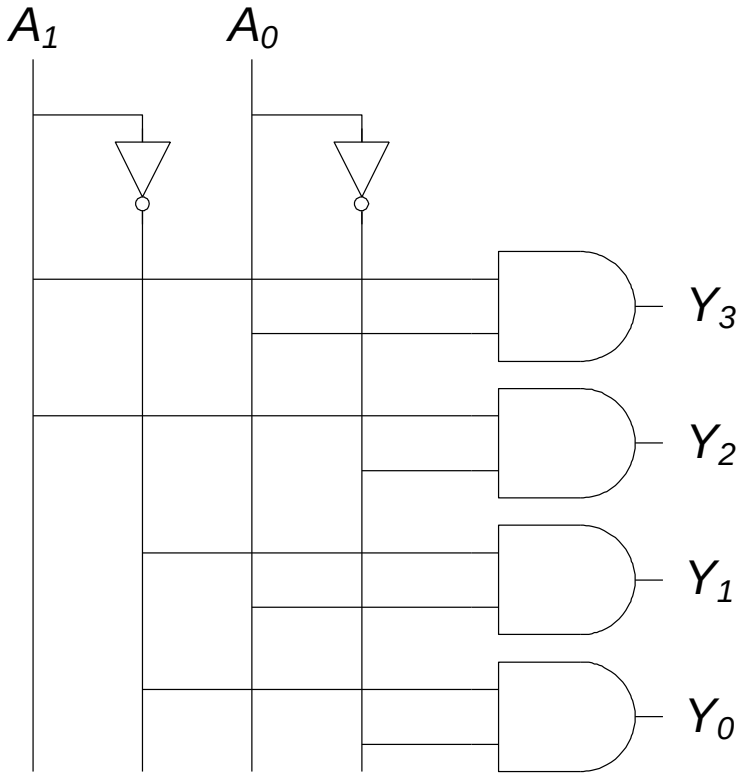
Decodificadores Binarios

- N entradas, 2^N salidas
- Salidas one-hot: solo una salida en alto a la vez



A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Implementación del Decodificador

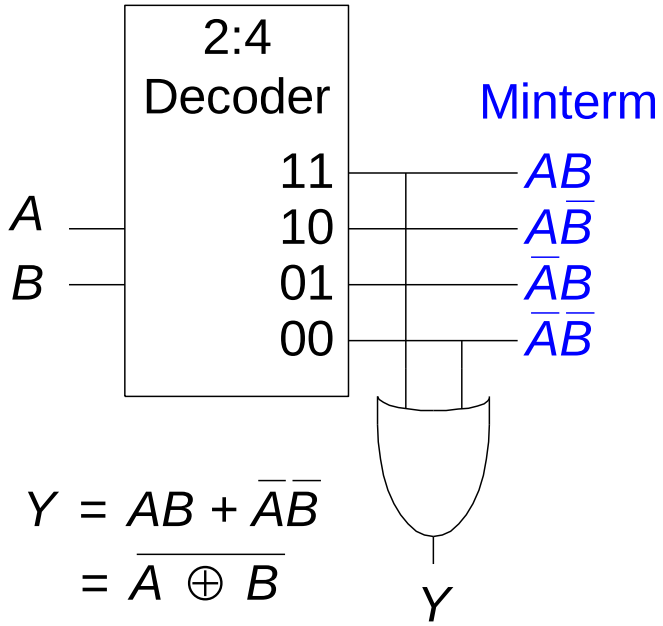


Decodificador Binario en VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decode_2to4_top is
    Port ( A  : in  STD_LOGIC_VECTOR (1 downto 0);  -- 2-bit input
           X  : out STD_LOGIC_VECTOR (3 downto 0);  -- 4-bit output
           EN : in  STD_LOGIC);                     -- enable input
end decode_2to4_top;

architecture Behavioral of decode_2to4_top is
begin
      process (A, EN)
      begin
          X <= "0000";         -- default output value
          if (EN = '1') then  -- active high enable pin
              case A is
                  when "00" => X(0) <= '1';
                  when "01" => X(1) <= '1';
                  when "10" => X(2) <= '1';
                  when "11" => X(3) <= '1';
                  when others => X <= "0000";
              end case;
          end if;
      end process;
end Behavioral;
```

Lógica Usando Decodificadores

- OR minterms

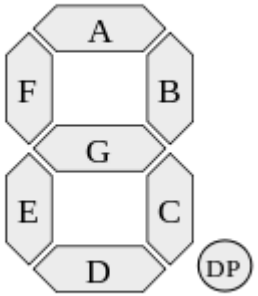


Decodificadores traductores entre códigos

- N entradas, 2^N salidas
- Puede haber varias salidas activadas a la vez.
- Ejemplo: decodificador BCD a 7 segmentos.

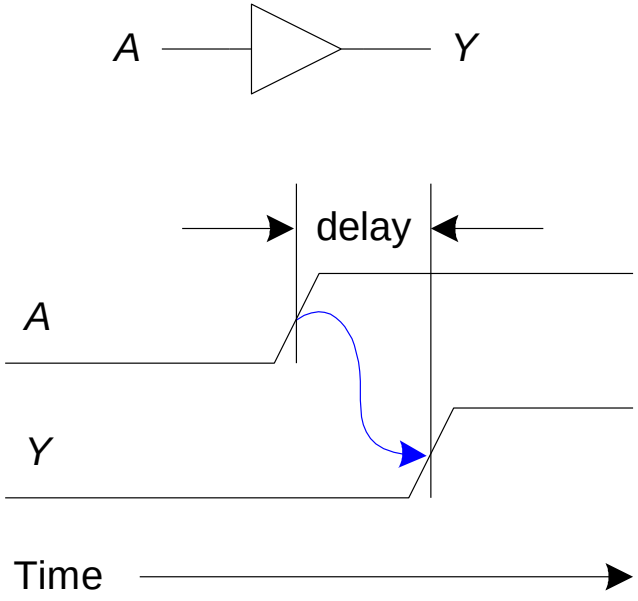
Ejemplo VHDL: bcd2seg.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity bcd2seg is
port (  input: in std_logic_vector(3 downto 0);
      seg: out std_logic_vector (6 downto 0));
end bin2seg;

architecture arch of bin2seg is
begin
  uno:PROCESS (input)
    BEGIN
      CASE input IS
        WHEN "0000"=>seg<="1000000";
        WHEN "0001"=>seg<="1111001";
        WHEN "0010"=>seg<="0100100";
        WHEN "0011"=>seg<="0110000";
        WHEN "0100"=>seg<="0011001";
        WHEN "0101"=>seg<="0010010";
        WHEN "0110"=>seg<="0000010";
        WHEN "0111"=>seg<="1111000";
        WHEN "1000"=>seg<="0000000";
        WHEN "1001"=>seg<="0010000";
        WHEN OTHERS=>seg<="1111111";
      END CASE;
   END PROCESS;
end arch;
```



Temporización (Timing)

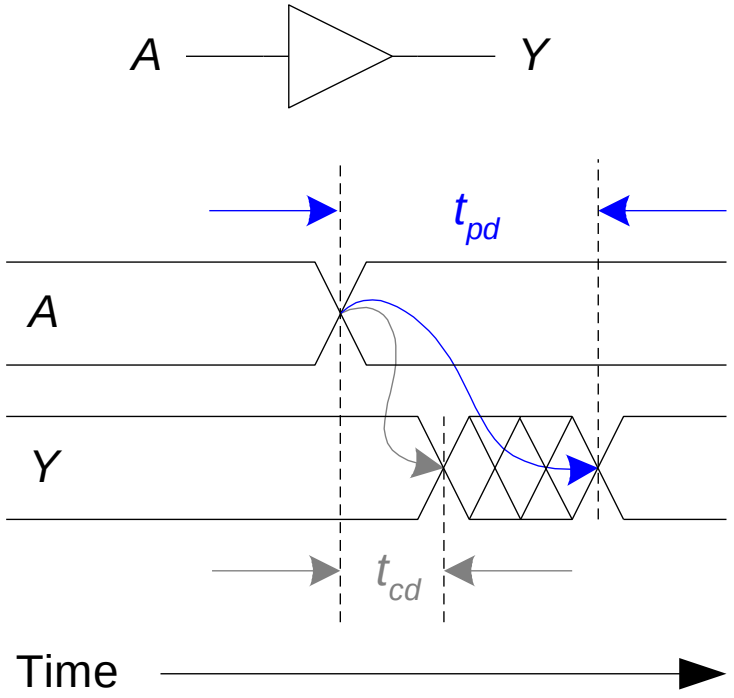
- Retardo entre el cambio en la entrada y el cambio en la salida.
- ¿Cómo construir circuitos más rápidos?



Propagation & Contamination Delay

- **Propagation delay:** t_{pd} = retardo desde que la entrada ha cambiado hasta que la salida se encuentra estable

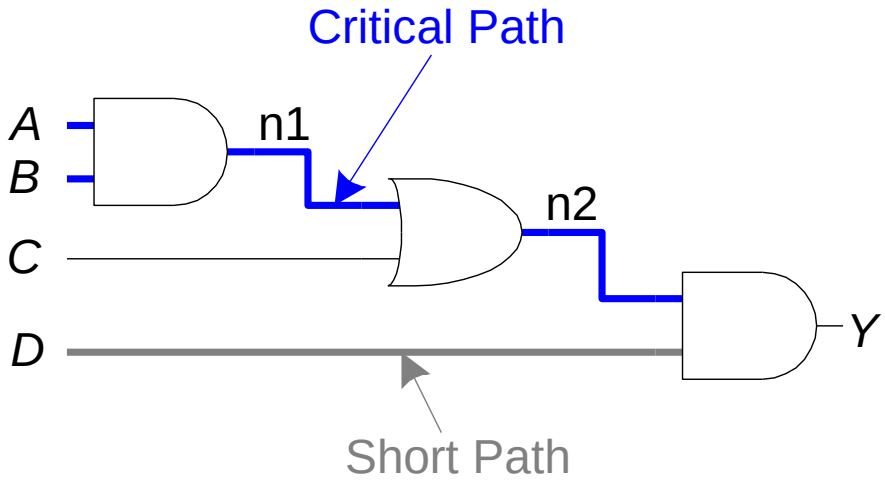
- **Contamination delay:** t_{cd} = tiempo mínimo entre que cambia la entrada y empieza a cambiar la salida (sin estar aún estable)



Propagation & Contamination Delay

- Retardos causados por
  - Capacitancia y resistencia en un circuito
  - Limitación de la velocidad de la luz
- Razones por las que t_{pd} y t_{cd} pueden ser diferentes:
  - Diferentes retardos de subida y bajada
  - Múltiples entradas y salidas, algunas de las cuales pasan más rápido a la salida
  - Los circuitos funcionan más lento cuando se calientan y más rápido al enfriarse.

Ruta Crítica



Ruta crítica: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

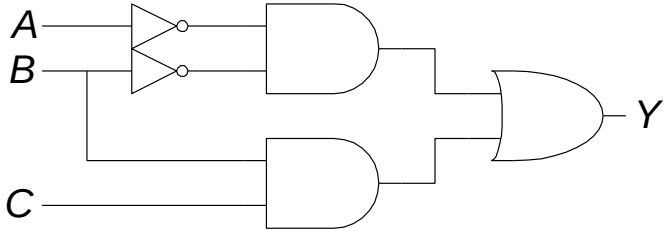
Ruta más corta: $t_{cd} = t_{cd\_AND}$

Glitches

- Cuando el cambio de una entrada produce varios cambios en la salida.

Ejemplo de Glitch

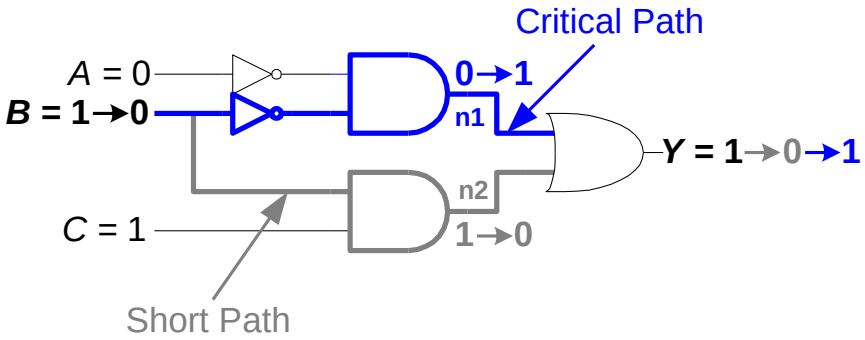
- ¿Qué ocurre cuando A = 0, C = 1, B cae?

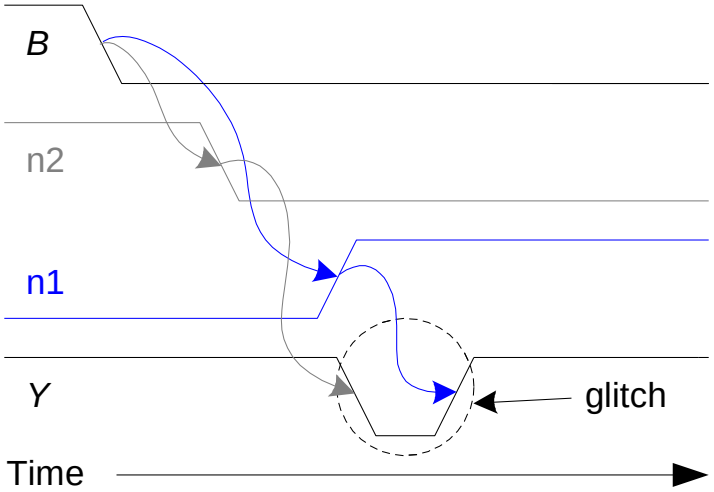


Y \ AB	00	01	11	10
C = 0	1	0	0	0
C = 1	1	1	1	0

$$Y = \bar{A}\bar{B} + BC$$

Ejemplo de Glitch





Solucionando el Glitch