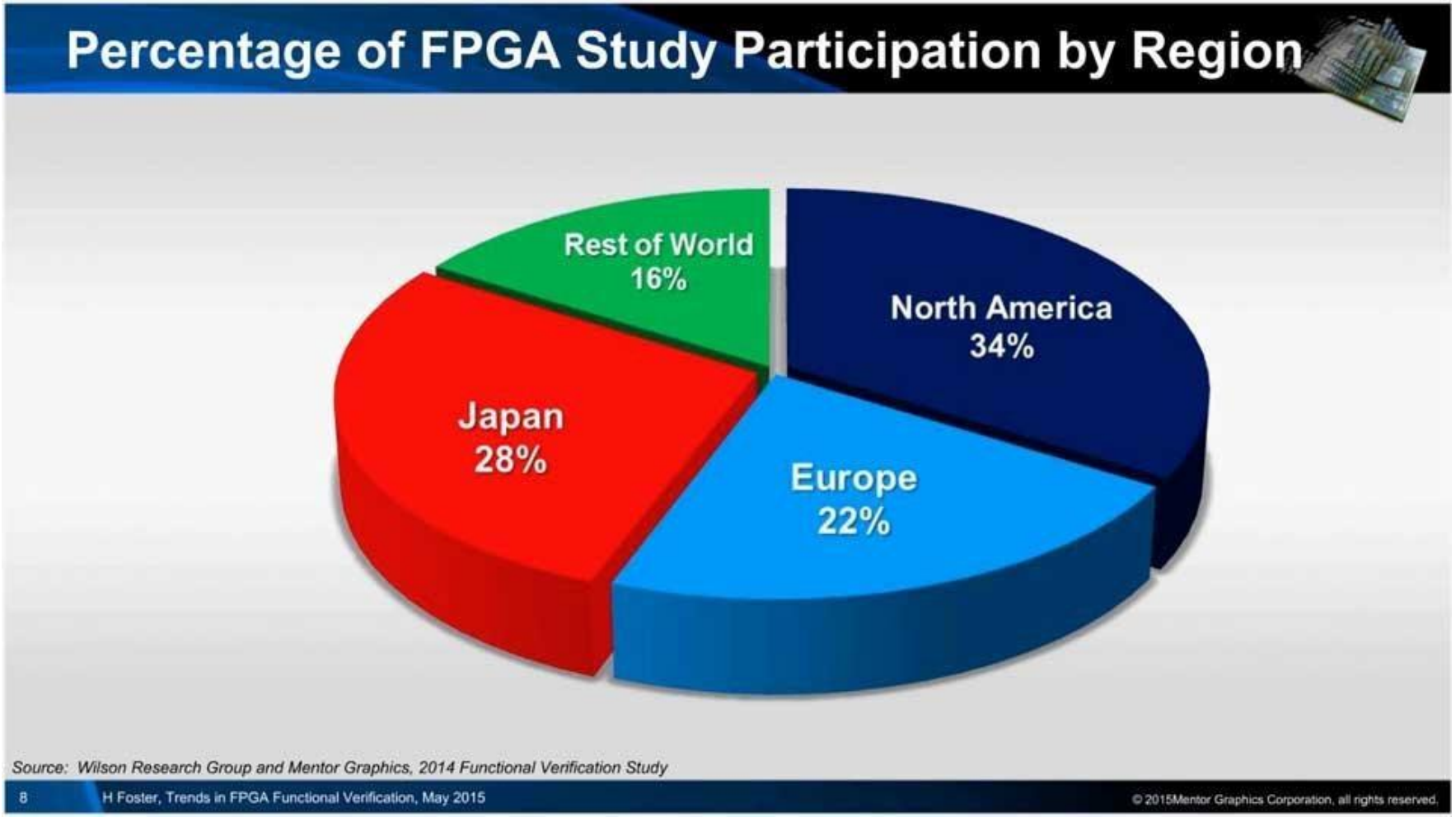
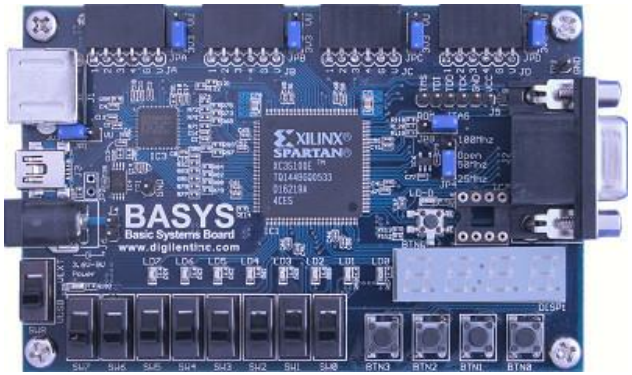
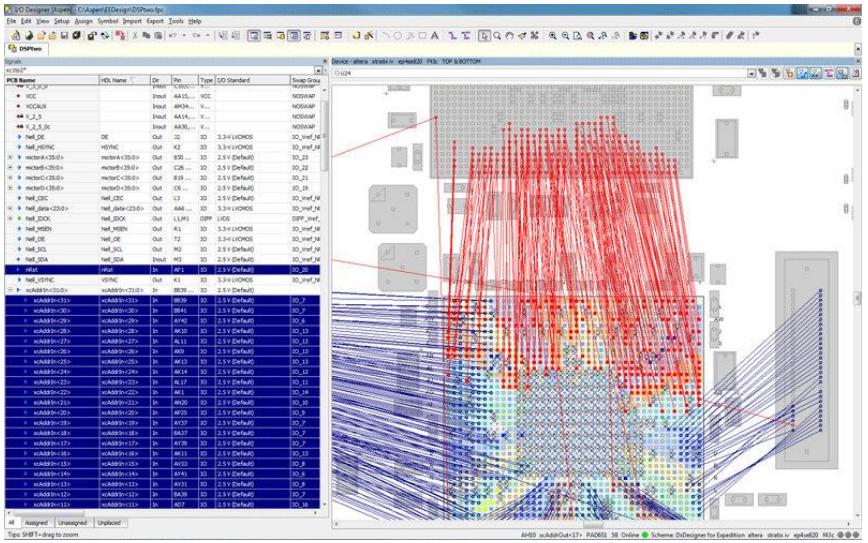
LENGUAJE DE DESCRIPCION DE HARDWARE

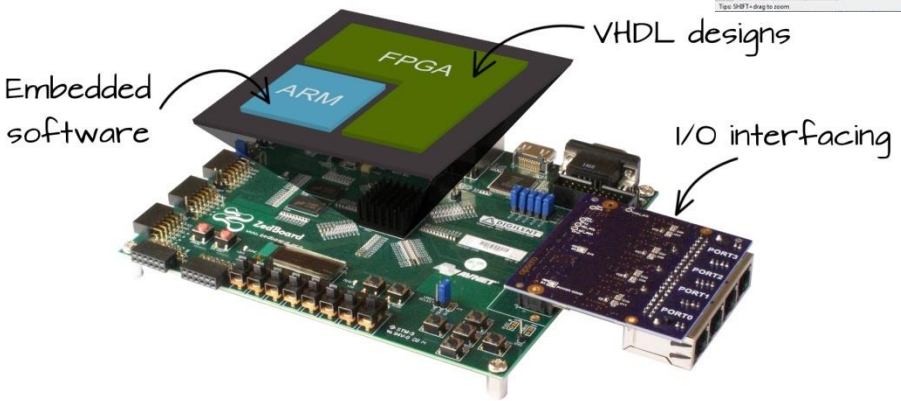
LENGUAJE DE DESCRIPCION DE HARDWARE



LENGUAJE DE DESCRIPCION DE HARDWARE







Dispositivos Programables

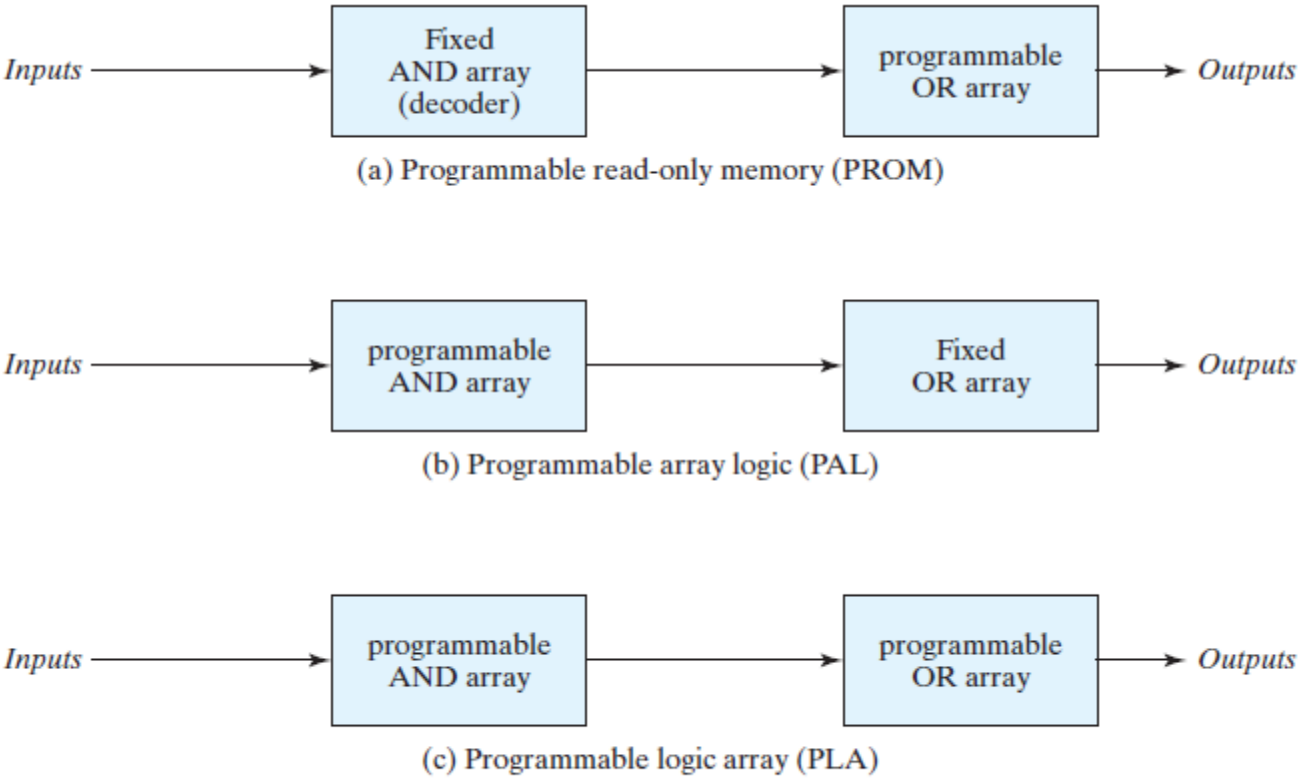


FIGURE 7.13
Basic configuration of three PLDs

PLA

$$F_1 = AB' + AC + A'BC'$$
$$F_2 = (AC + BC)'$$

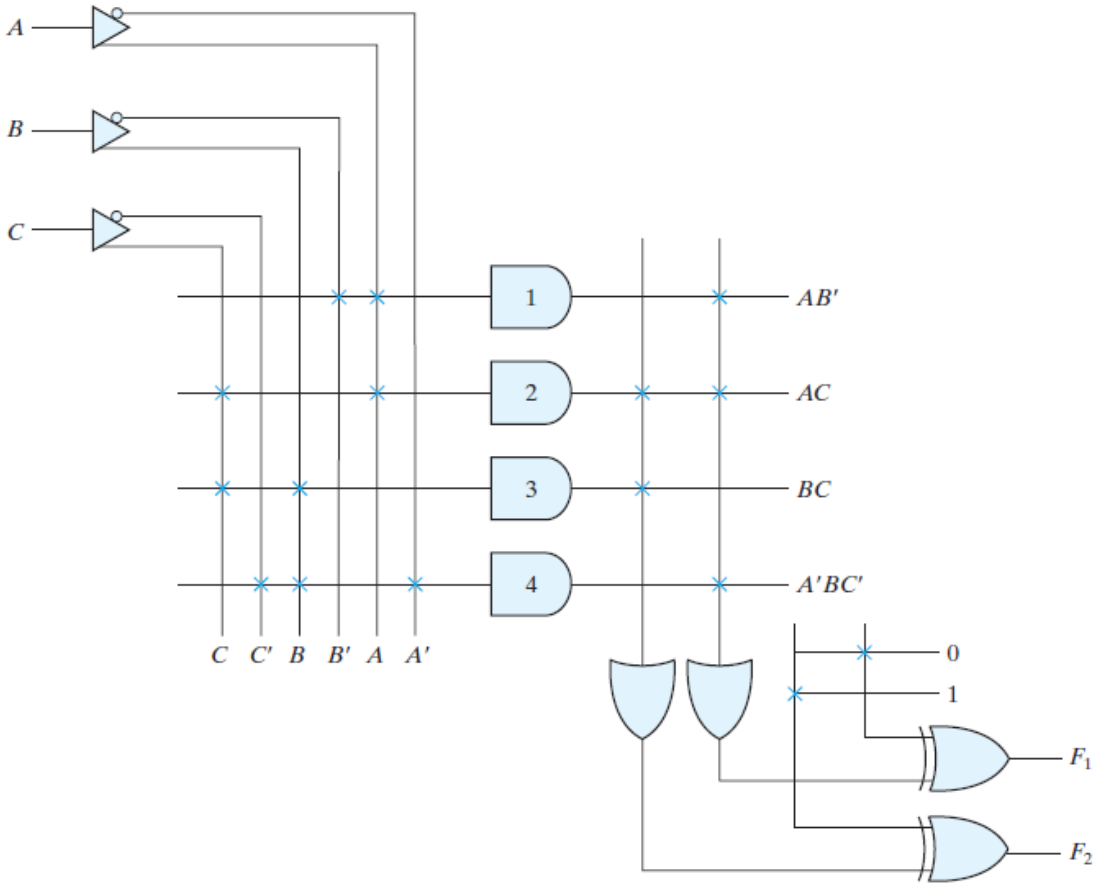


FIGURE 7.14
PLA with three inputs, four product terms, and two outputs

PAL

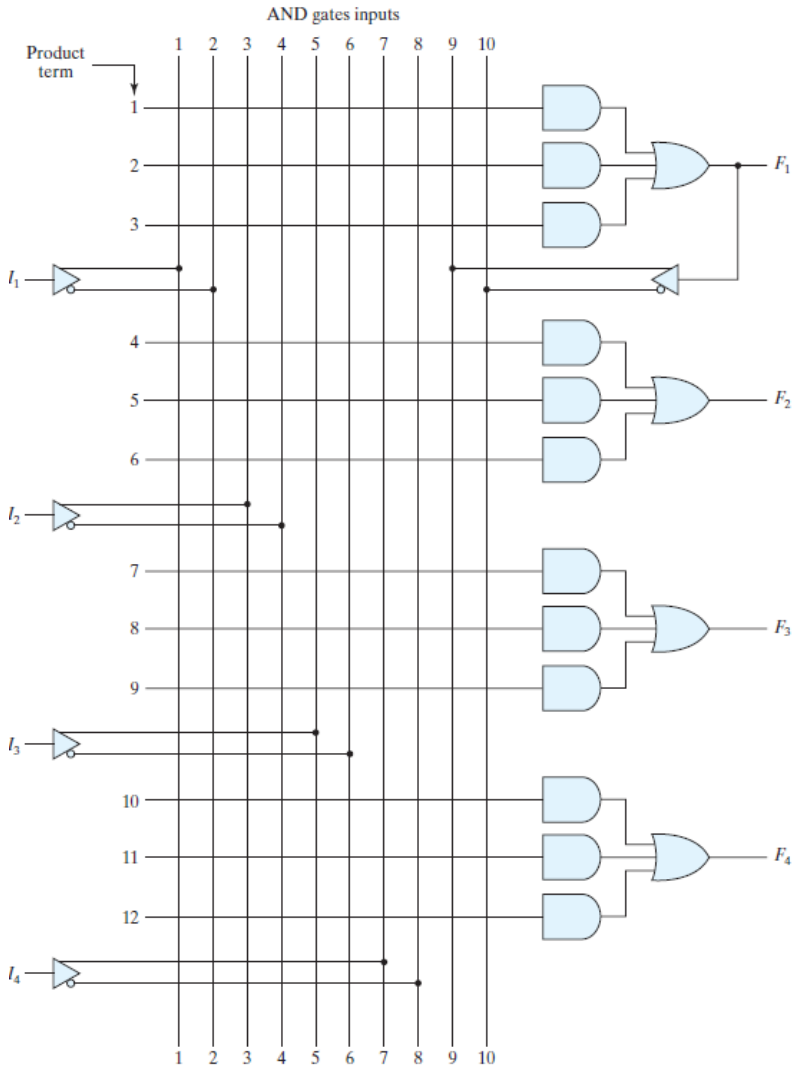


FIGURE 7.16
PAL with four inputs, four outputs, and a three-wide AND–OR structure

PAL

Table 7.6
PAL Programming Table

Product Term	AND Inputs A	B	C	D	w	Outputs
1	1	1	0	—	—	$w = ABC' + A'B'CD'$
2	0	0	1	0	—	
3	—	—	—	—	—	
4	1	—	—	—	—	$x = A + BCD$
5	—	1	1	1	—	
6	—	—	—	—	—	
7	0	1	—	—	—	$y = A'B + CD + B'D'$
8	—	—	1	1	—	
9	—	0	—	0	—	
10	—	—	—	—	1	$z = w + AC'D' + A'B'C'D$
11	1	—	0	0	—	
12	0	0	0	1	—	

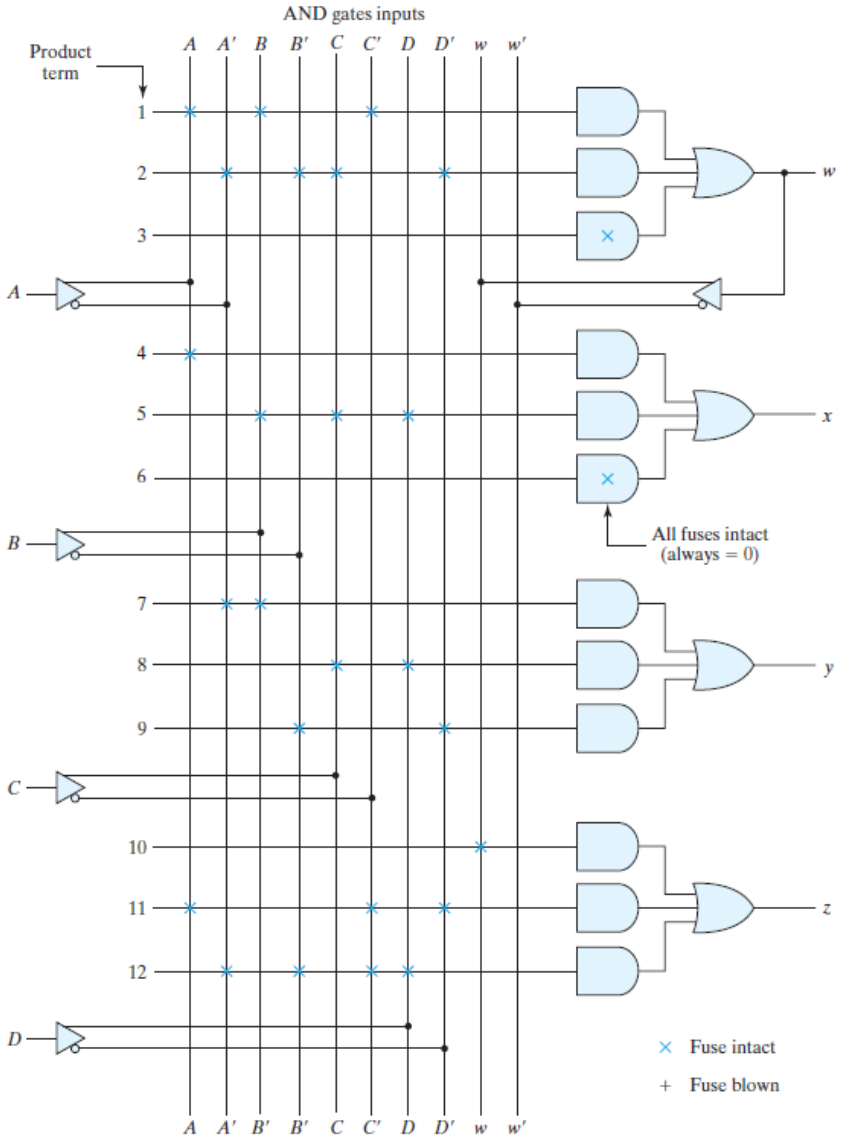


FIGURE 7.17
Fuse map for PAL as specified in Table 7.6

SPLD - CPLD- FPGA

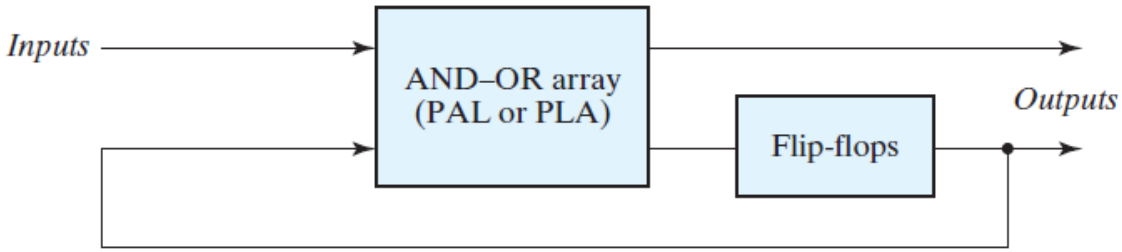


FIGURE 7.18
Sequential programmable logic device

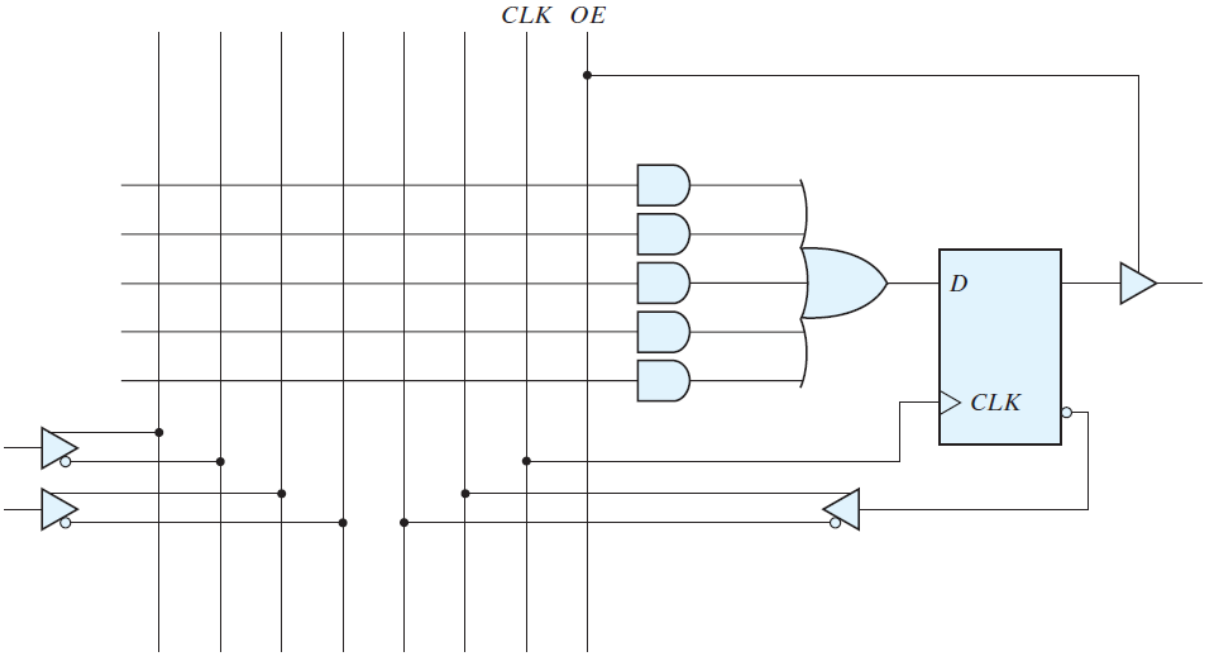


FIGURE 7.19
Basic macrocell logic

SPLD - CPLD- FPGA

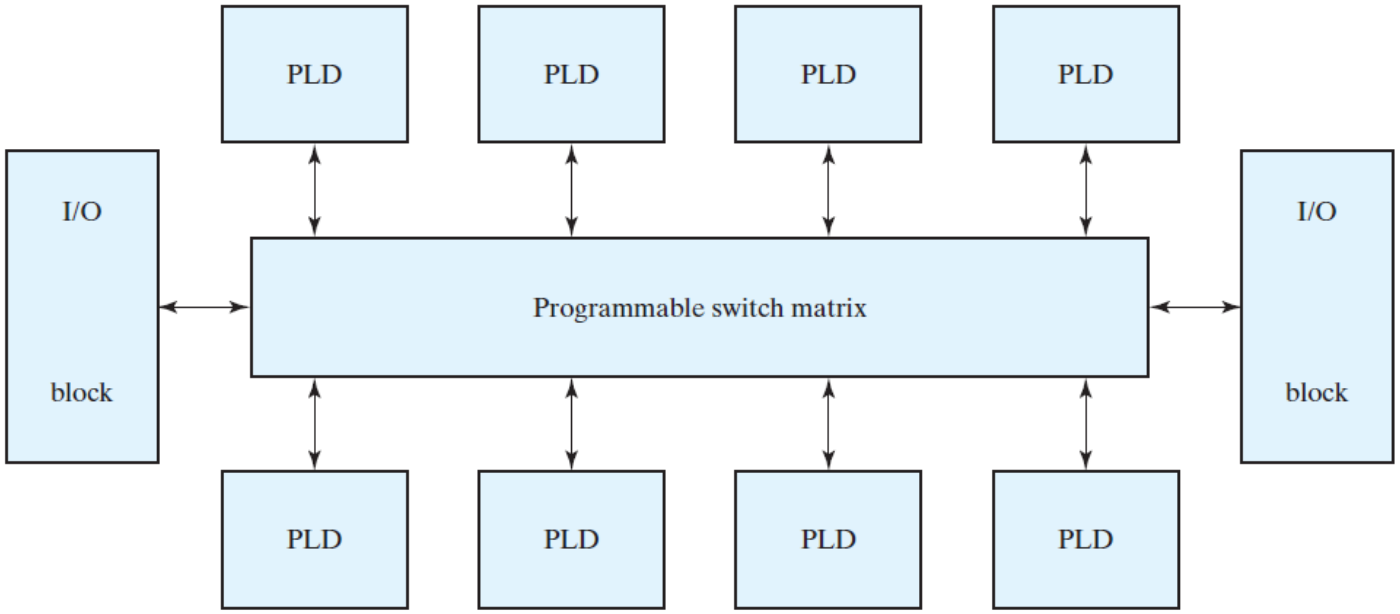
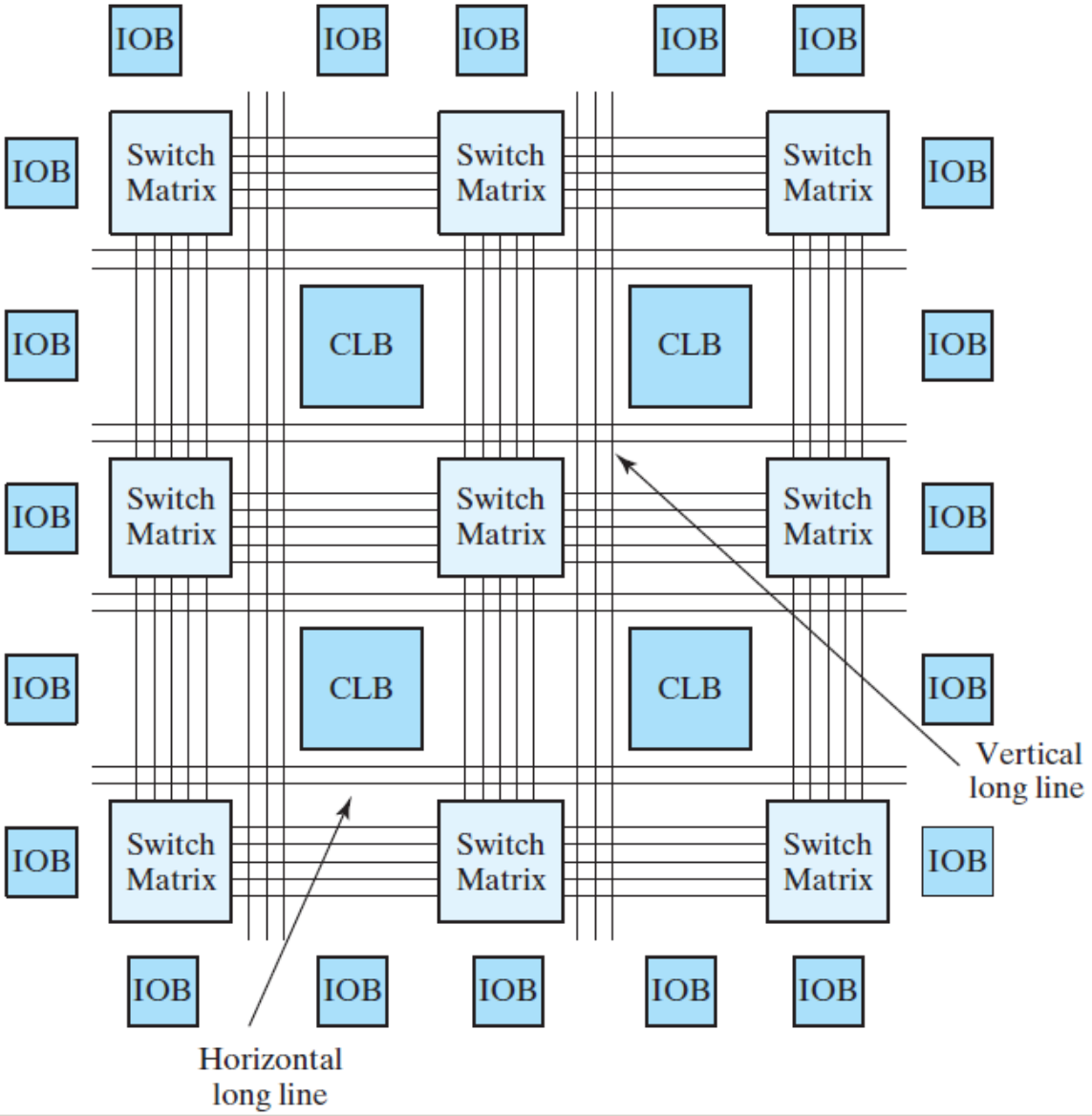


FIGURE 7.20
General CPLD configuration

FPGA


IOB
Switch Matrix
CLB
Vertical long line
Horizontal long line

FPGA - CLB

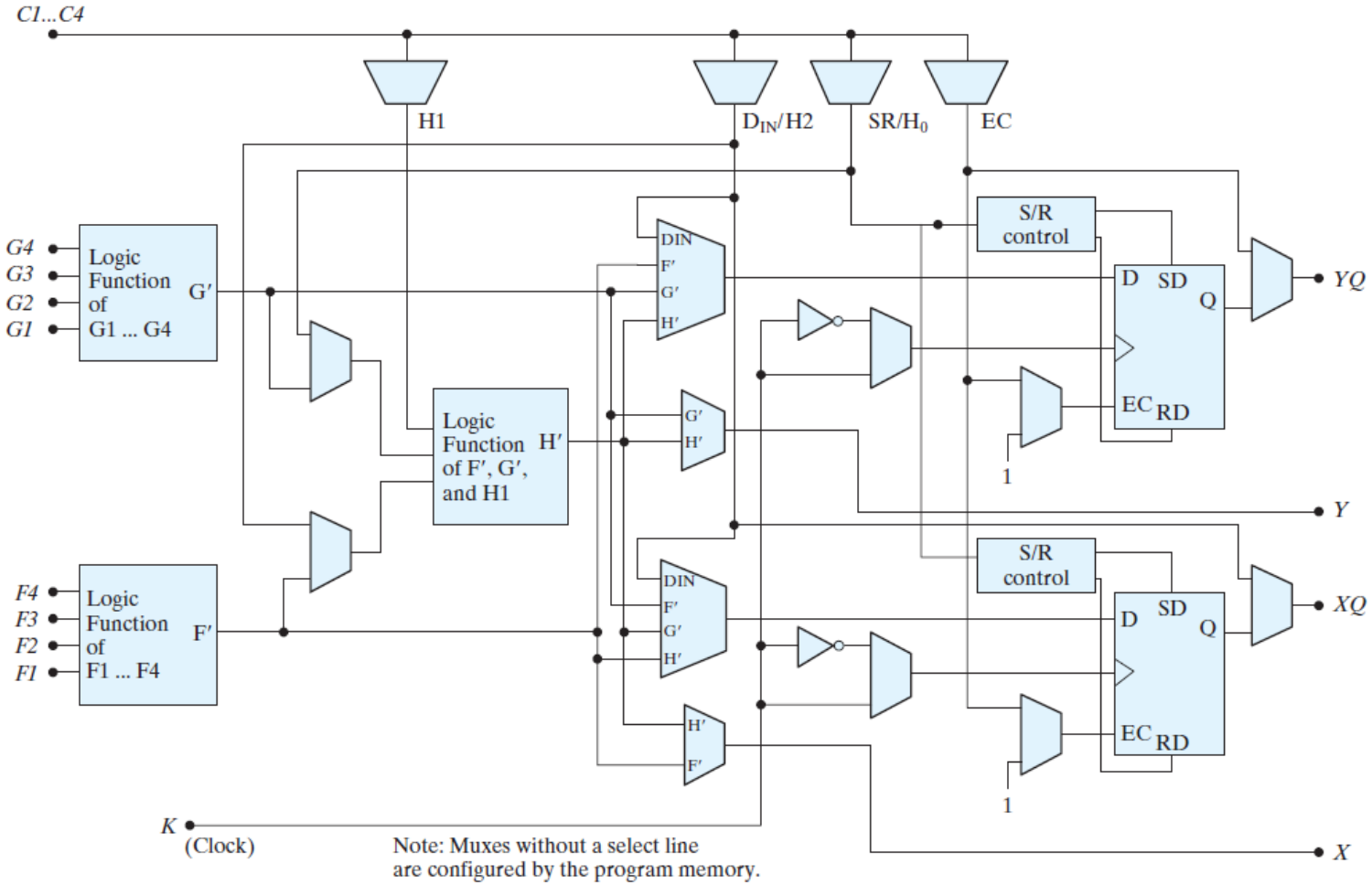


FIGURE 7.22
CLB architecture

FPGA – Block RAM

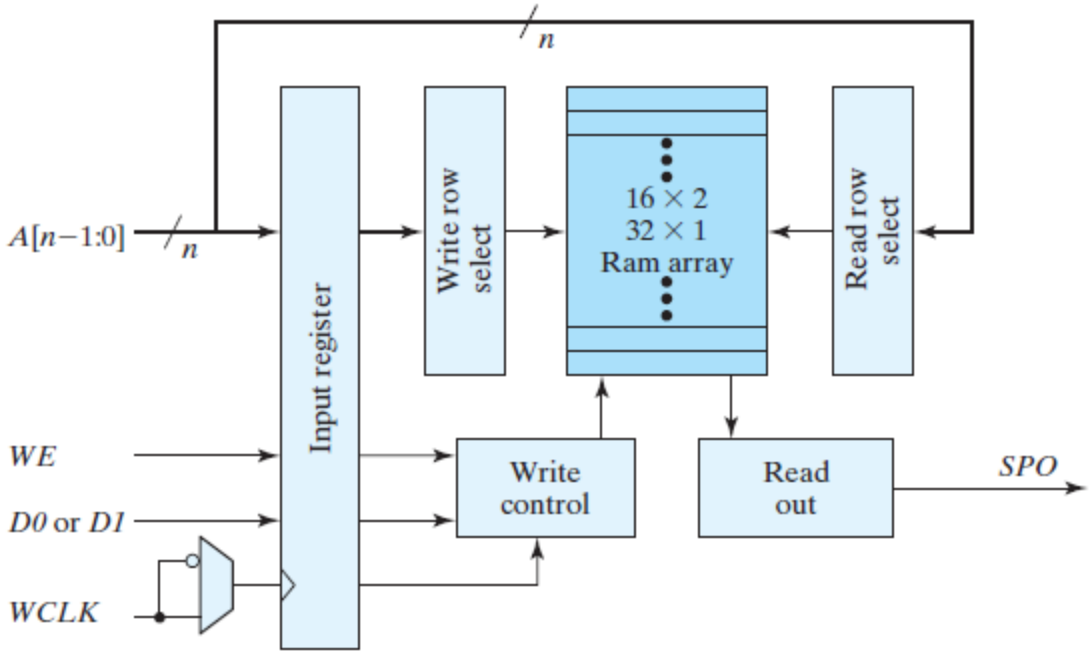


FIGURE 7.26
Distributed RAM cell formed from a lookup table

FPGA – Block RAM Dual Port

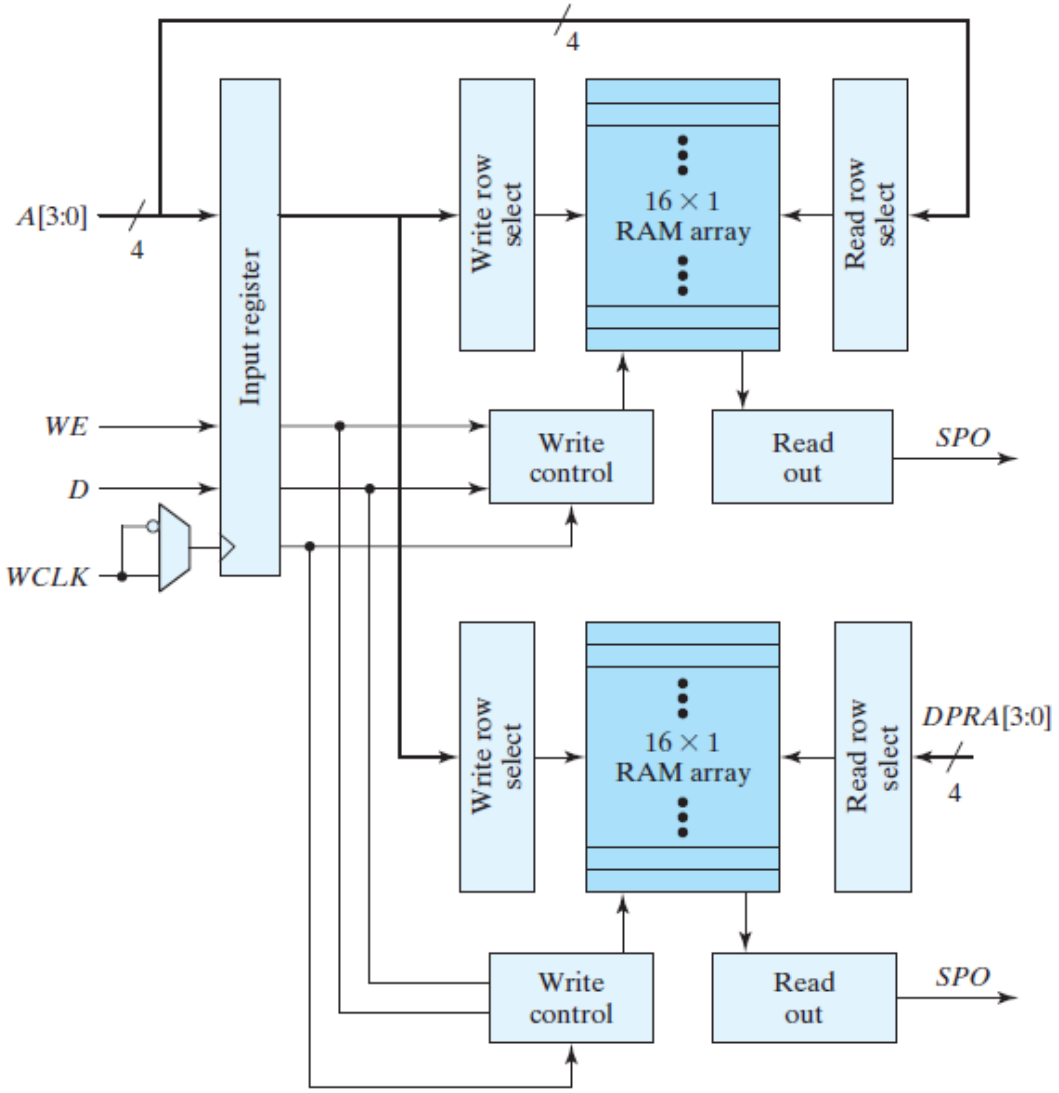


FIGURE 7.27
Spartan dual-port RAM

FPGA

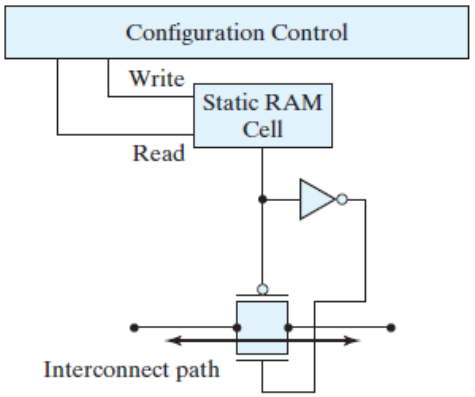


FIGURE 7.23
RAM cell controlling a PIP transmission gate

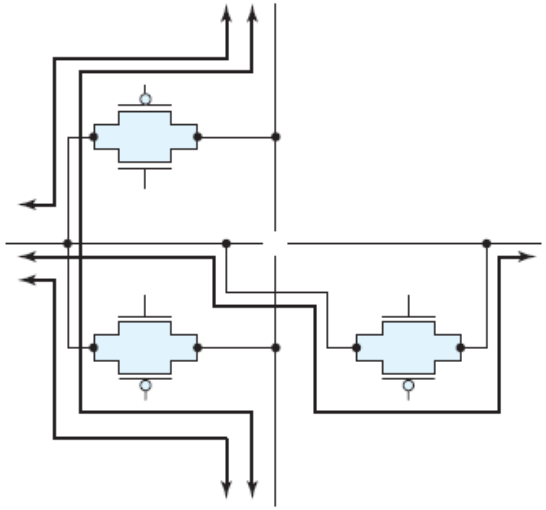


FIGURE 7.24
Circuit for a programmable PIP

FPGA - IOB

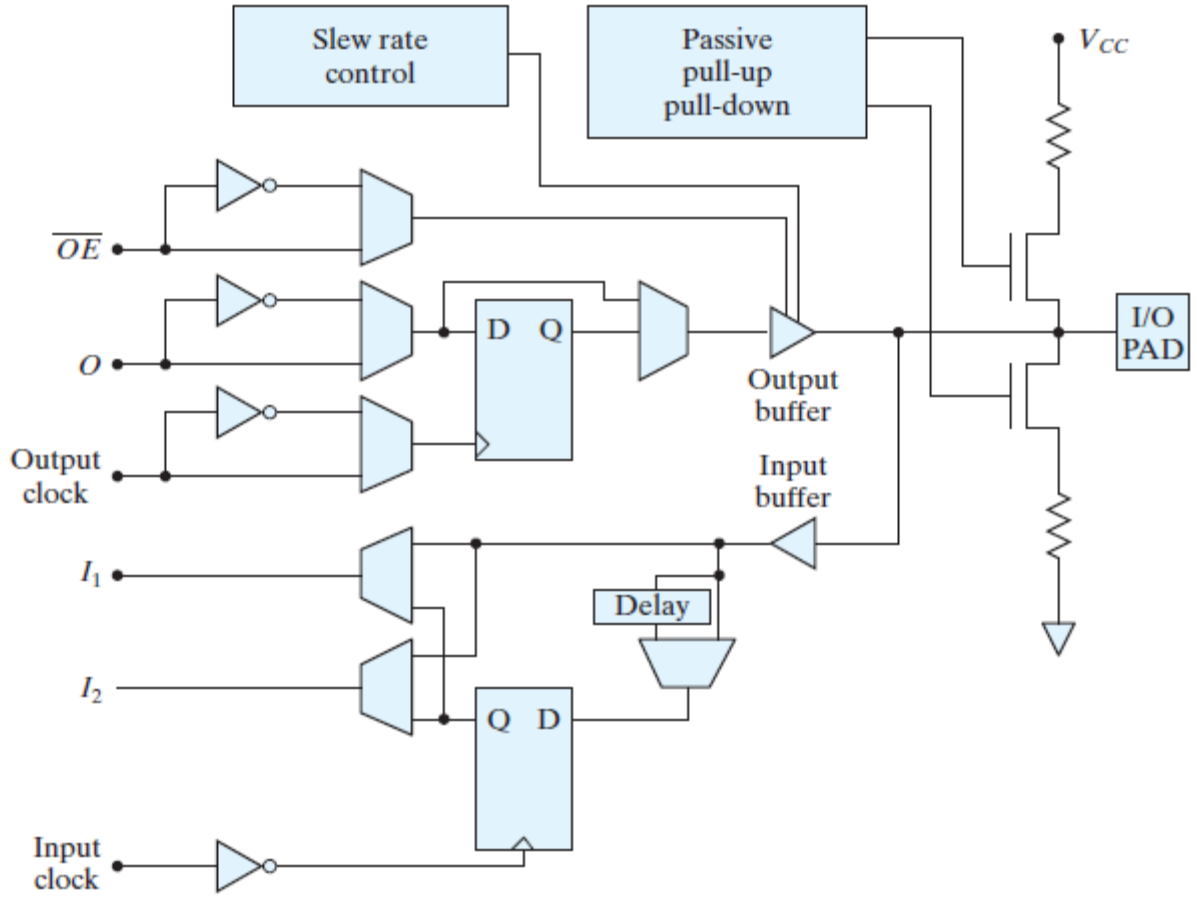


FIGURE 7.25
XC4000 series IOB

FPGA - IOB

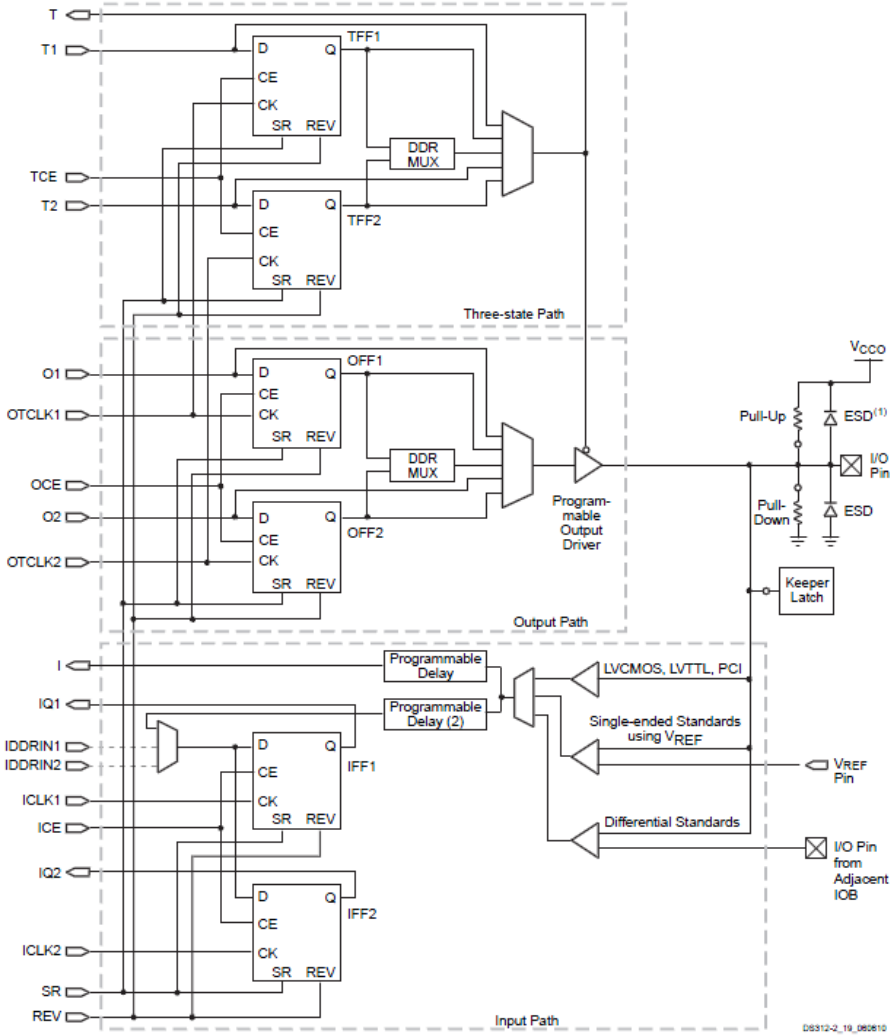


Figure 10-1: **Simplified IOB Diagram**

FPGA - IOB

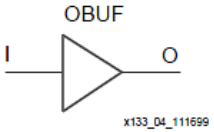


Figure 10-5: **Output Buffer (OBUF) Symbol**

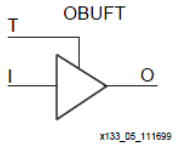


Figure 10-6: **3-State Output Buffer (OBUFT) Symbol**

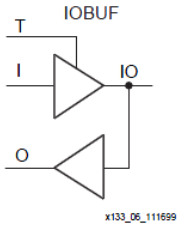


Figure 10-7: **Input/Output Buffer (IOBUF) Symbol**

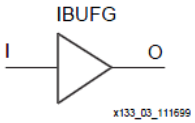


Figure 10-3: **Global Clock Input Buffer (IBUFG) Symbol**

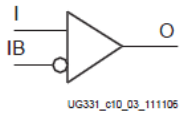


Figure 10-4: **Differential Input Buffer (IBUFDS) Symbol**

```
-- INOUT_PORT : inout STD_LOGIC;
--**Insert the following between the
-- 'architecture' and 'begin' keywords**
  signal IN_SIG, OUT_SIG, T_ENABLE: std_logic;
component IOBUF
      port (I, T: in std_logic;
            O: out std_logic;
           IO: inout std_logic);
end component;
   --**Insert the following after the 'begin' keyword**
U1: IOBUF port map (I => OUT_SIG, T => T_ENABLE,
                    O => IN_SIG, IO => INOUT_PORT);
```

FPGA

Table 7.8
Spartan II Device Attributes

Spartan II FPGAs	XC2S15	XC2S30	XC2S50	XC2S100	XC2S150	XC2S200
System Gates[1]	6K–15K	13K–30K	23K–50K	37K–100K	52K–150K	71K–200K
Logic Cells[2]	432	972	1,728	2,700	3,888	5,292
Block RAM Bits	16,384	24,576	32,768	40,960	49,152	57,344
Max Avail I/O	86	132	176	196	260	284

[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

FPGA

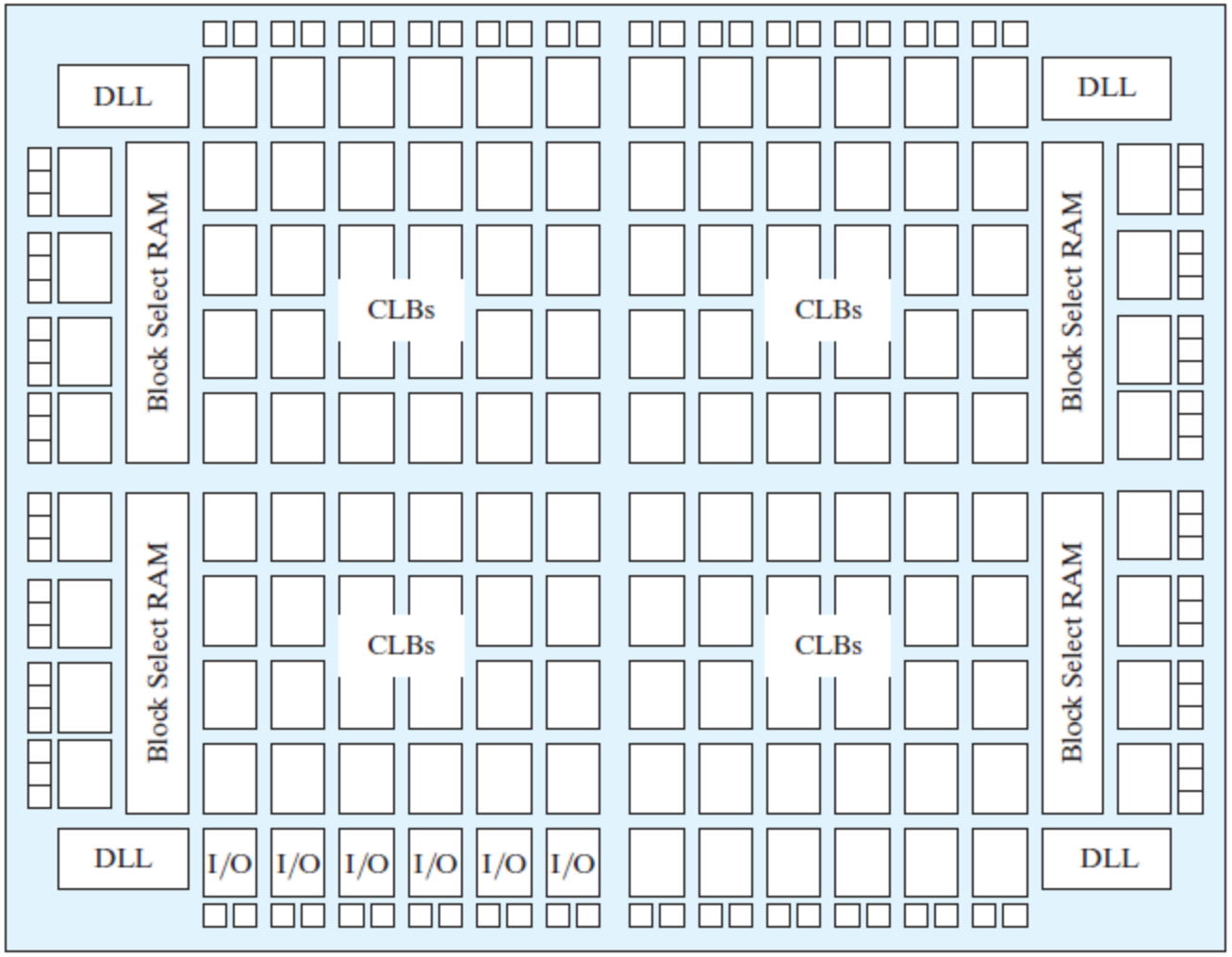


FIGURE 7.28
Spartan II architecture

FPGA

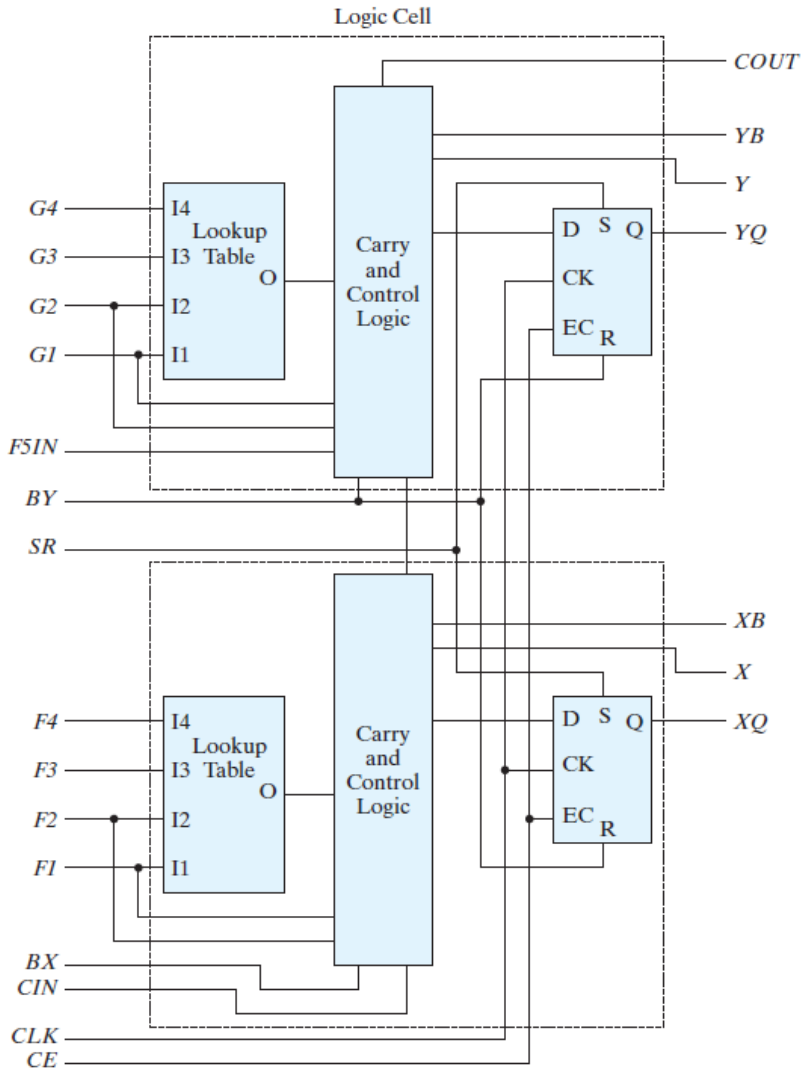


FIGURE 7.29
Spartan II CLB slice

FPGA

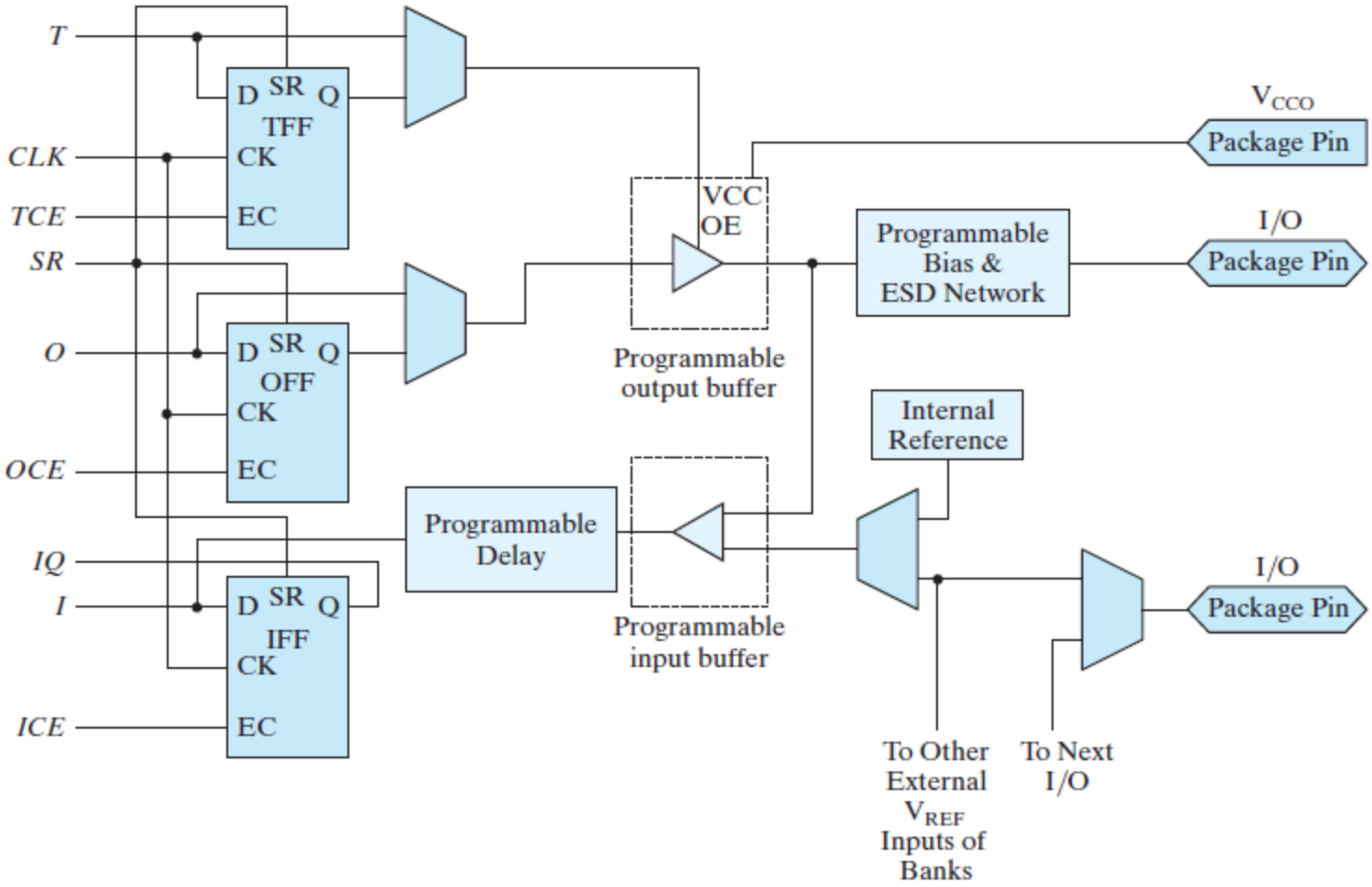


FIGURE 7.30
Spartan II IOB

VHDL

El lenguaje de programación VHDL (Very High Speed Integrated Circuit Hardware *D*escription Language) es un lenguaje que describe el comportamiento del circuito, es decir describe el hardware

En la Fig 1 se observan los tres estilos de descripción

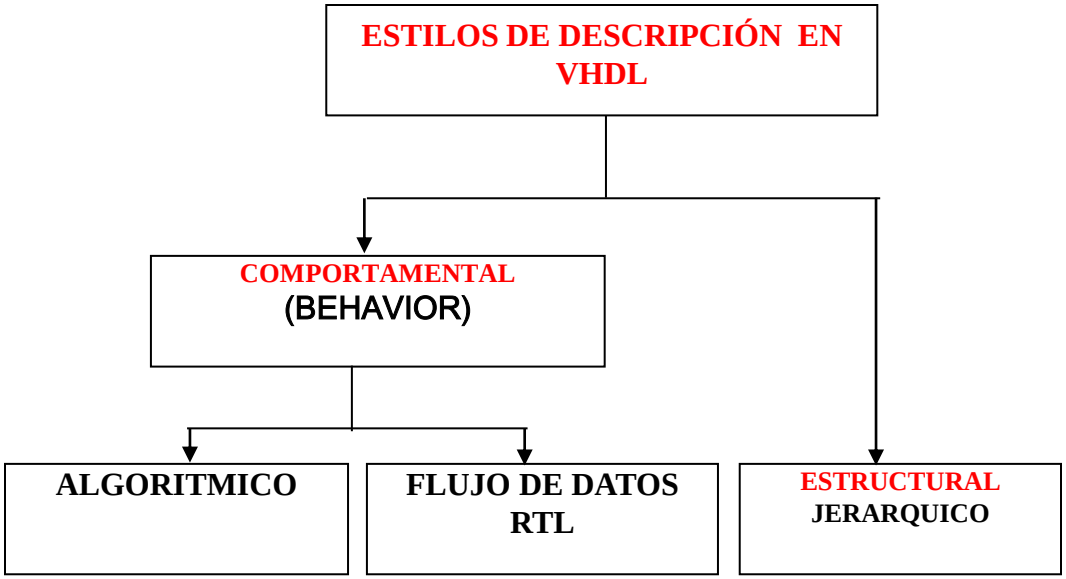


Fig. 1 Estilos de descripción VHDL

VHDL – Lenguaje para síntesis y modelado de circuitos – Fernado Pardo y Jose Boluda

Editorial Alfaomega

VHDL - David Maxinez - Editorial C.E.C.S.A

Diseño de sistemas con VHDL – Editorial Paraninfo

VHDL

ESTRUCTURA BASICA DE UN ARCHIVO FUENTE VHDL

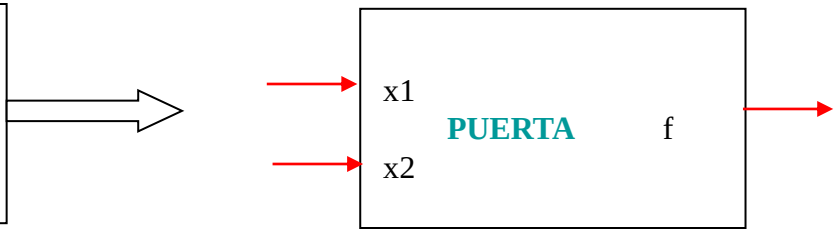
ENCABEZAMIENTO	**Library** <nombre_libreria> **Use**<nombre_librería>.<nombre_paquete>. **all**
ENTIDAD	**Entity**<nombre_entidad>**is** <listado de puertos> --Declaración de pines **end** <nombre_entidad>;
ARQUITECTURA	**Architecture**<nombre_arquitectura> **of**<nombre_entidad> **is** --Declaracion de señales internas --Declaracion de tipos de datos definidos por el usuario --Declaracion de componentes en caso de instanciación **begin** --Cuerpo de la arquitectura --Se define la funcionalidad del diseño con: --Asignaciones concurrentes --Procesos --Instanciación de componentes **end**<nombre_arquitectura>;

VHDL

ENTIDAD Y ARQUITECTURA

```
entity puerta is
port (x1, x2: IN BIT;
          f:OUT BIT);

end  puerta;
```




```
architecture AND of puerta is
begin
      f <= x1 AND x2
end  AND;
```

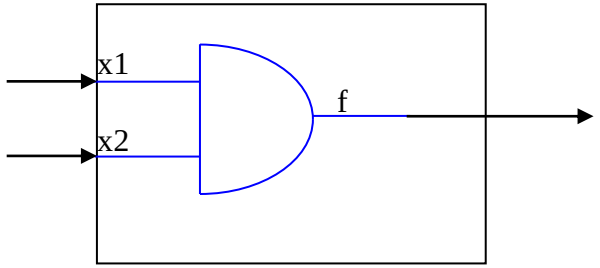


Fig. 4 Entidad y Arquitectura

PUERTOS:

IN: ESTE PUERTO (DE ENTRADA) SE PUEDE SOLO LEER NUNCA ESCRIBIR

OUT: ESTE PUERTO (DE SALIDA) SE PUEDE SOLO ESCRIBIR NUNCA LEER, EL VALOR DE LA SEÑAL NO PUEDE USARSE DENTRO DE LA ENTIDAD

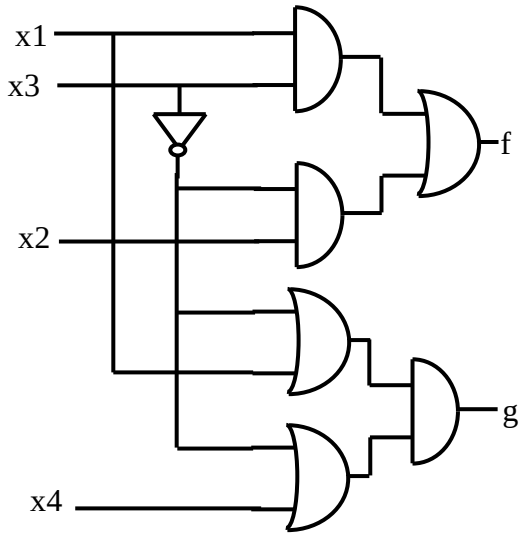
INOUT: (ENTRADA-SALIDA) PERMITE LECTURA Y ESCRITURA

BUFFER: (SALIDA) PERMITE LECTURA Y ESCRITURA, EL VALOR DE LA SEÑAL PUEDE USARSE DENTRO DE LA ENTIDAD

ANÁLISIS DE CÓDIGO VHDL

```
ENTITY ejemplo2 IS
        PORT (x1, x2, x3, x4 :  IN     BIT;
              f,g       :  OUT  BIT);
END ejemplo2;

ARCHITECTURE LogicFunc  OF ejemplo2   IS
BEGIN
        f <= (x1 AND x3) OR (NOT x3 AND x2);
        g <= (NOT x3 OR x1) AND (NOT x3 OR x4);
END LogicFunc;
```



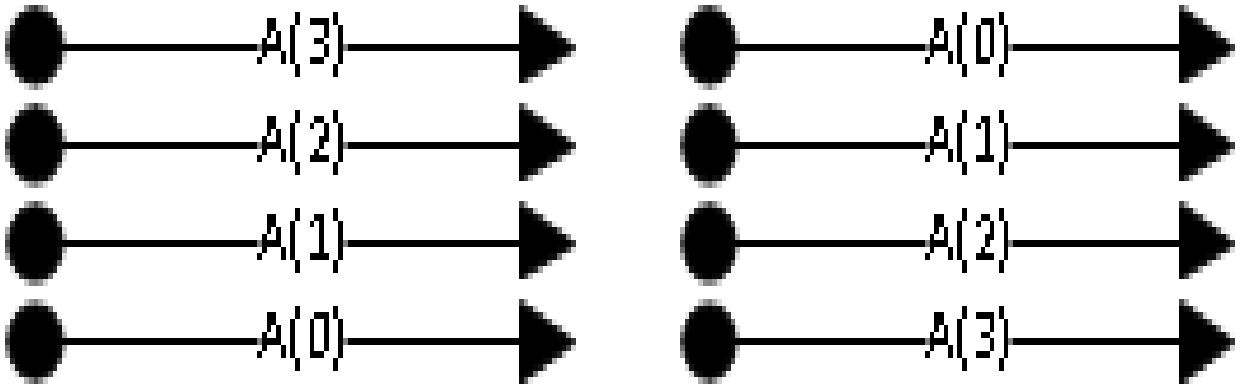
- **STD_LOGIC**
- **STD_LOGIC_VECTOR**
- **SIGNAL**

- IEEE 1164
  library IEEE;
  use IEEE.STD_LOGIC_1164.ALL;
- **STD_LOGIC**

```
PACKAGE std_logic_1164 IS

    ------------------------------------------------------
    -- logic state system  (unresolved)
    ------------------------------------------------------
    TYPE std_ulogic IS ( 'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'H',  -- Weak     1
                         '-'   -- Don't care
                       );
```

- STD_LOGIC_VECTOR
- **A : std_logic_vector(3 downto 0) | A : std_logic_vector(0 to 3)**

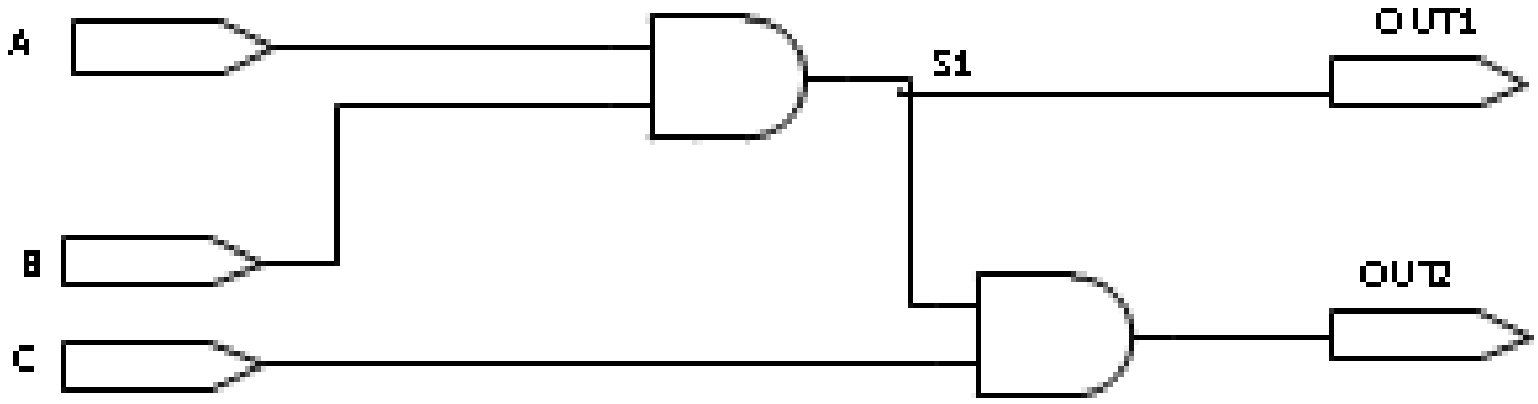


- **A(0) <='1';**
- **dato<=A(0) & A1 (3) & A1(5) ; -- Concatenación**

VHDL SIGNALS

- **Signals:**
  - **Se declaran dentro de la arquitectura.**
  - **Tienen las mismas propiedades que los puertos.**
  - **No salen fuera de la arquitectura.**
  - **Son cables y señales internas del dispositivo.**
  - **Se asignan igual que los puertos.**
  - **Son bidireccionales.**
- **Declaracion**
- **<nombre>: tipo;**
- **D : std_logic;**
- **D2: std_logic_vector(0 to 10);**
- **D3. std_logic_vector(10 downto 0)**

VHDL SIGNALS




```
Entity compuerta is
   Port ( A : in  STD_LOGIC;
      B: STD_LOGIC;
      C: STD_LOGIC;
      OUT1 : OUT std_logic;
      OUT2. OUT std_logic);
end compuerta;
```

VHDL SIGNALS

```
architecture Behavioral of compuerta is
Signal S1: std_logic;
begin
out1<=S1;
S1<= A and B;
OUT2<=  S1 and C;
end Behavioral;
```

VHDL STD_LOGIC

IEEE 1164
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

STD_LOGIC

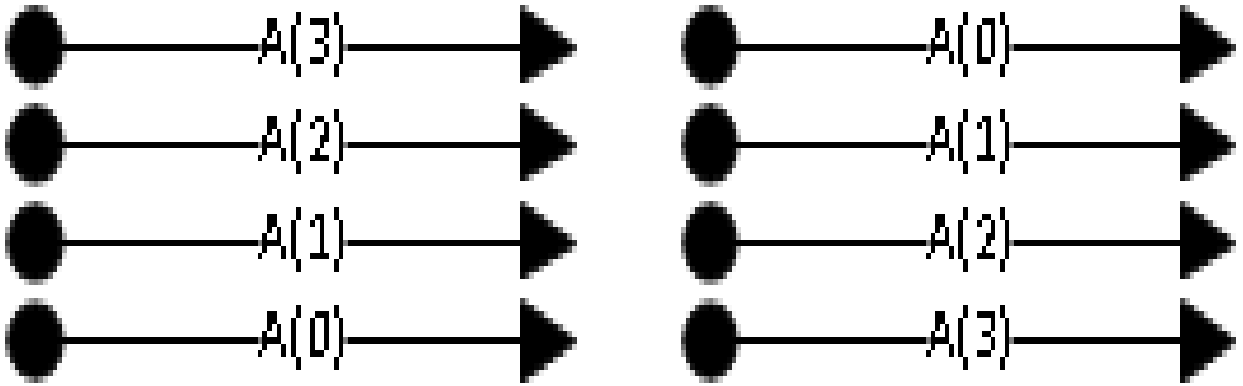
```
PACKAGE std_logic_1164 IS

    ------------------------------------------------------
    -- logic state system  (unresolved)
    ------------------------------------------------------
    TYPE std_ulogic IS ( 'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'H',  -- Weak     1
                         '-'   -- Don't care
                       );
```

VHDL STD_LOGIC

STD_LOGIC_VECTOR

A : STD_LOGIC_VECTOR(3 DOWNTO 0) | A : STD_LOGIC_VECTOR(0 TO 3)



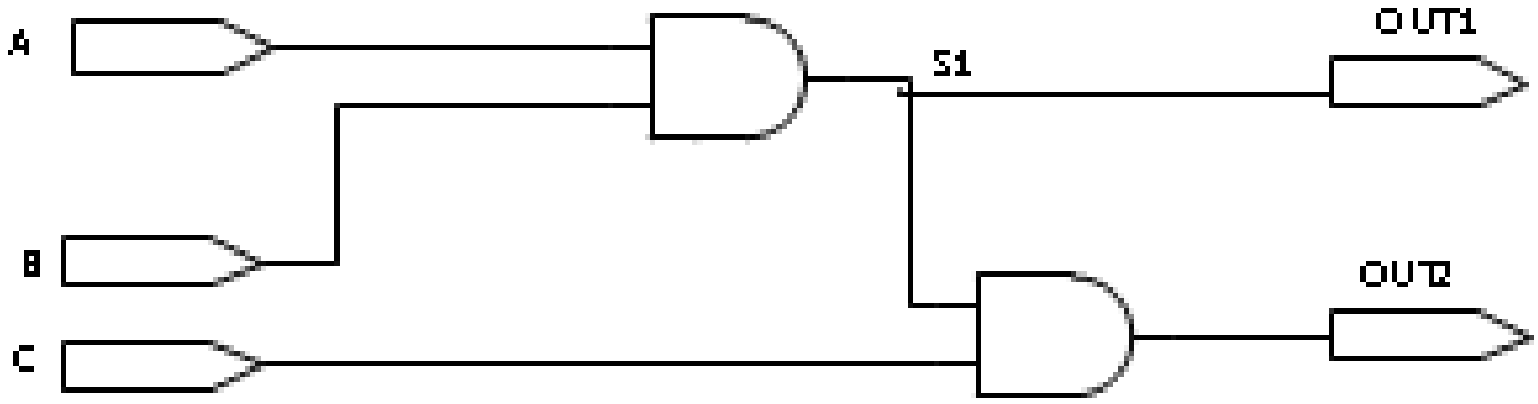

```
A(0) <='1';
 DATO<=A(0) & A1 (3) & A1(5) ; -- CONCATENACIÓN
```

VHDL SIGNALS

- **Signals:**
  - **Se declaran dentro de la arquitectura.**
  - **Tienen las mismas propiedades que los puertos.**
  - **No salen fuera de la arquitectura.**
  - **Son cables y señales internas del dispositivo.**
  - **Se asignan igual que los puertos.**
  - **Son bidireccionales.**
- **Declaración**
- **<nombre>: tipo;**

- **D : std_logic;**
- **D2: std_logic_vector(0 to 10);**
- **D3. std_logic_vector(10 downto 0)**

VHDL SIGNALS

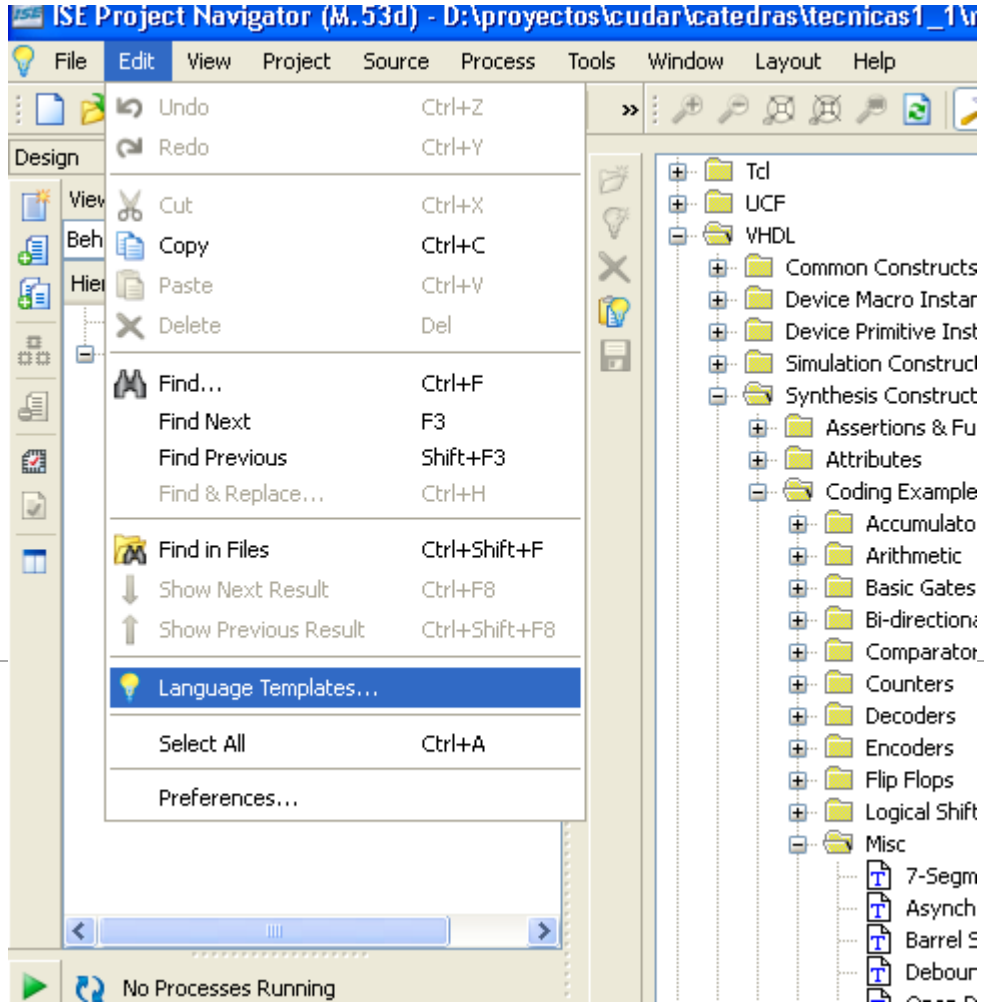



```
Entity compuerta is
   Port ( A : in  STD_LOGIC;
          B: STD_LOGIC;
          C: STD_LOGIC;
          OUT1 : OUT std_logic;
          OUT2. OUT std_logic);
end compuerta;
```

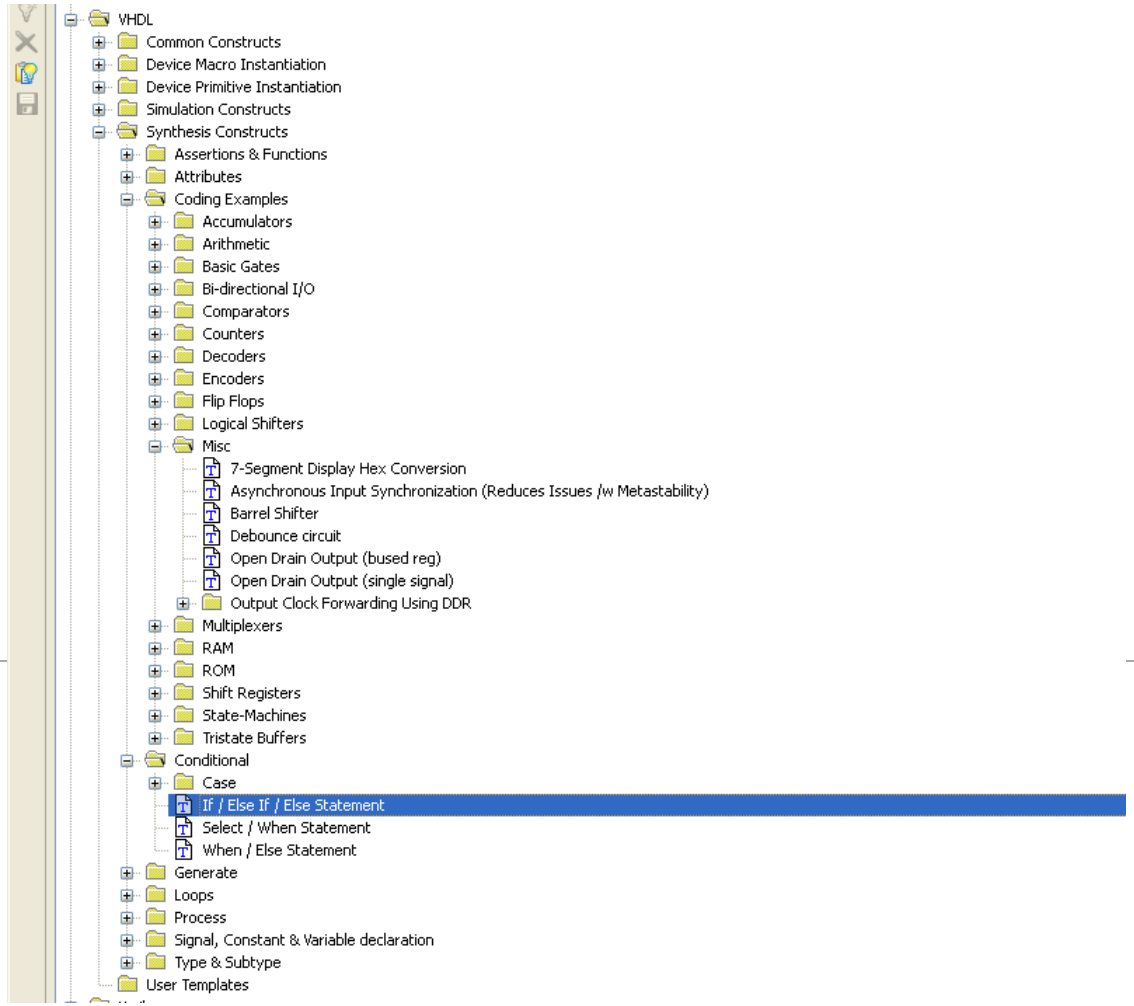
VHDL SIGNALS

```
architecture Behavioral of compuerta is
Signal S1: std_logic;
begin
out1<=S1;
S1<= A and B;
OUT2<=  S1 and C;
end Behavioral;
```

VHDL – ISE TEMPLATES



VHDL – ISE TEMPLATES



TEMPLATE

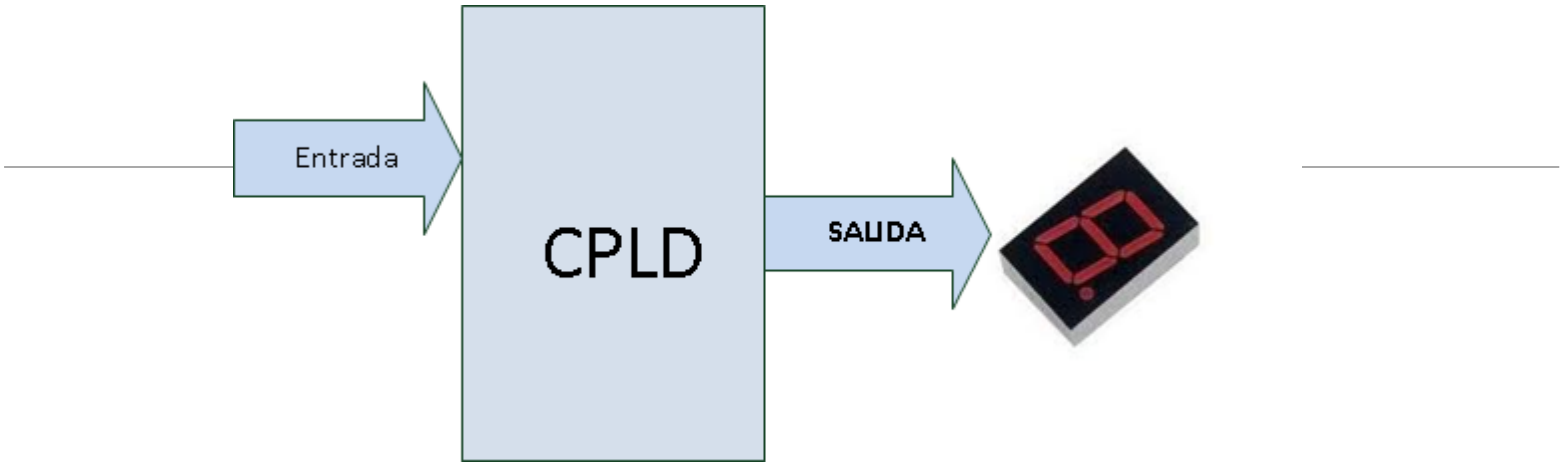
```
WITH <CHOICE_EXPRESSION> SELECT
    <NAME> <= <EXPRESSION> WHEN <CHOICES>,
            <EXPRESSION> WHEN <CHOICES>,
            <EXPRESSION> WHEN OTHERS;
```

TEMPLATE BCD-7SEG

```
WITH HEX SELECT
  LED<= "1111001" WHEN "0001",  --1
      "0100100" WHEN "0010",  --2
      "0110000" WHEN "0011",  --3
      "0011001" WHEN "0100",  --4
      "0010010" WHEN "0101",  --5
      "0000010" WHEN "0110",  --6
      "1111000" WHEN "0111",  --7
      "0000000" WHEN "1000",  --8
      "0010000" WHEN "1001",  --9
      "0001000" when "1010",   --A
            "0000011" when "1011",   --b
            "1000110" when "1100",   --C
            "0100001" when "1101",   --d
            "0000110" when "1110",   --E
            "0001110" when "1111",   --F
            "1000000" when others;   --0
```

Entidad

```
ENTITY BCD_7SEGMENT IS
    PORT ( ENTRADA : IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
           SALIDA : OUT  STD_LOGIC_VECTOR (6 DOWNTO 0));
END BCD_7SEGMENT;
```



Arquitectura

```
ARCHITECTURE BEHAVIORAL OF BCD_7SEGMENT IS

BEGIN

    WITH ENTRADA SELECT

   SALIDA<= "1111001" WHEN "0001",  --1

        "0100100" WHEN "0010",   --2

        "0110000" WHEN "0011",   --3

        "0011001" WHEN "0100",   --4

        "0010010" WHEN "0101",   --5

        "0000010" WHEN "0110",   --6

"1111000" when "0111",   --7
        "0000000" when "1000",   --8
        "0010000" when "1001",   --9
        "0001000" when "1010",   --A
        "0000011" when "1011",   --b
        "1000110" when "1100",   --C
        "0100001" when "1101",   --d
        "0000110" when "1110",   --E
        "0001110" when "1111",   --F
        "1000000" when others;   --0
end Behavioral;
```

when - else

```
<NAME> <= <EXPRESSION> WHEN <CONDITION> ELSE
          <EXPRESSION> WHEN <CONDITION> ELSE
          <EXPRESSION>;
```

Arquitectura

```
ARCHITECTURE BEHAVIORAL OF BCD_7SEGMENT IS

BEGIN

        SALIDA<= "1111001" WHEN ENTRADA="0001" ELSE  --1
              "0100100" WHEN ENTRADA="0010" ELSE   --2
              "0110000" WHEN ENTRADA="0011" ELSE --3
              "0011001" WHEN ENTRADA="0100" ELSE  --4
              "0010010" WHEN ENTRADA="0101" ELSE  --5
              "0000010" WHEN ENTRADA="0110" ELSE  --6
              "1111000" WHEN ENTRADA="0111" ELSE  --7
              "0000000" WHEN ENTRADA="1000" ELSE   --8
              "0010000" WHEN ENTRADA="1001" ELSE   --9
              "1000000" ;   --0

END BEHAVIORAL;
```

Rtl schematic