

# Práctica1. Descripción de circuitos en VHDL

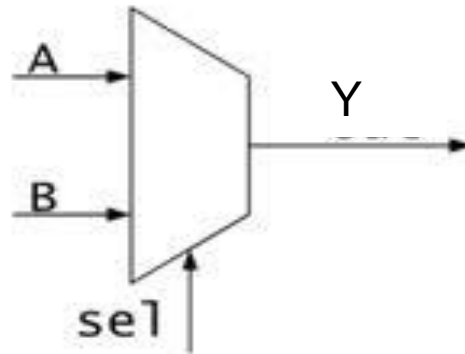
## OBJETIVOS

- Diseñar un módulo VHDL:
  - Entity
  - Architecture
  - Libraries
- Comparar los diferentes estilos de descripción VHDL:
  - Dataflow
  - Behavioral
- Describir circuitos mediante sentencias (Behavioral):
- Declarar y trabajar con el objeto “Signal”
- Trabajar con el tipo de datos compuestos STD\_LOGIC\_VECTOR
- Realizar un diseño complejo: Descripción estructural
- Operaciones aritméticas con los tipos Signed y Unsigned

# 1. Estilo de descripción “Dataflow”

# 1. Estilo de descripción “Dataflow”

**Proyecto03.** Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida. El multiplexor selecciona entre dos posibles entradas, de 1 bit cada una, “A” y “B” en función del valor de la línea de selección “sel”. El nombre de la salida es “Y”.



Si	sel = “0”	entonces	Y = A
Si	sel = “1”	entonces	Y = B

# 1. Estilo de descripción “Dataflow”

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida.

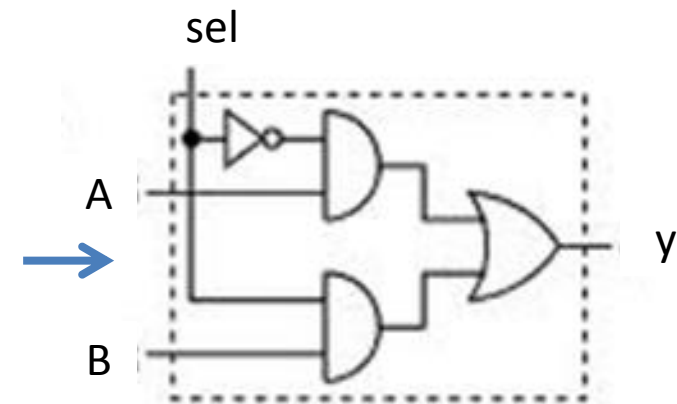
***Diseño digital tradicional:***

Entradas      Salida

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



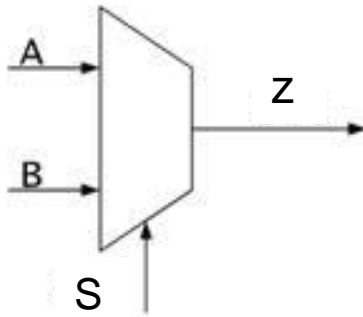
$$Y = A \cdot \overline{\text{sel}} + B \cdot \text{sel}$$



# 1. Estilo de descripción “Dataflow”

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida.

Entidad



New Source Wizard

Define Module

Specify ports for module.

Entity name: mux2\_df\_1bit

Architecture name: Dataflow

Port Name	Direction	Bus	MSB	LSB
A	in	<input type="checkbox"/>		
B	in	<input type="checkbox"/>		
Sel	in	<input type="checkbox"/>		
Y	out	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		

Next Cancel

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

entity mux2_df_1 is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          Sel : in  STD_LOGIC;
          Y : out  STD_LOGIC);
end mux2_df_1 ;
```

# 1. Estilo de descripción *Dataflow*

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1”, de dos entradas y una salida.

## Arquitectura dataflow 1ª

```
architecture Dataflow of mux2_df_1bit is
begin
    -- Descripción del MUX mediante
    -- ecuación boolena (DATAFLOW)

    Y <= (A and (not Sel)) or (B and Sel);
end Dataflow ;
```

Comentarios!!

Asignación simple.

() Precedencia de  
operadores.  
Claridad

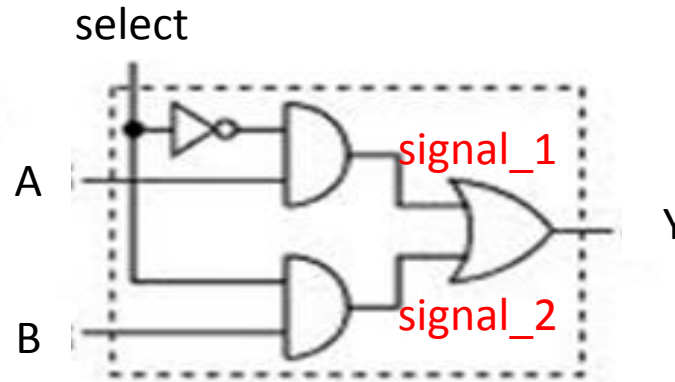
Palabras  
clave en  
minúscula!!

# 1. Estilo de descripción “Dataflow”

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida.

## Arquitectura dataflow 2ª

Comentar la línea de código anterior



```
architecture Dataflow of mux2_df_1bit is
    -- Declaraciones de señales
    signal signal_1, signal_2: std_logic;
Begin
    -- y <= (A and (not sel)) or (B and sel);
    -- Descripción del MUX separando en
    -- varias la ecuación booleana (DATAFLOW)
    signal_1 <= A and (not Sel);
    signal_2 <= B and Sel;
    Y <= signal_1 or signal_2;
end Dataflow;
```

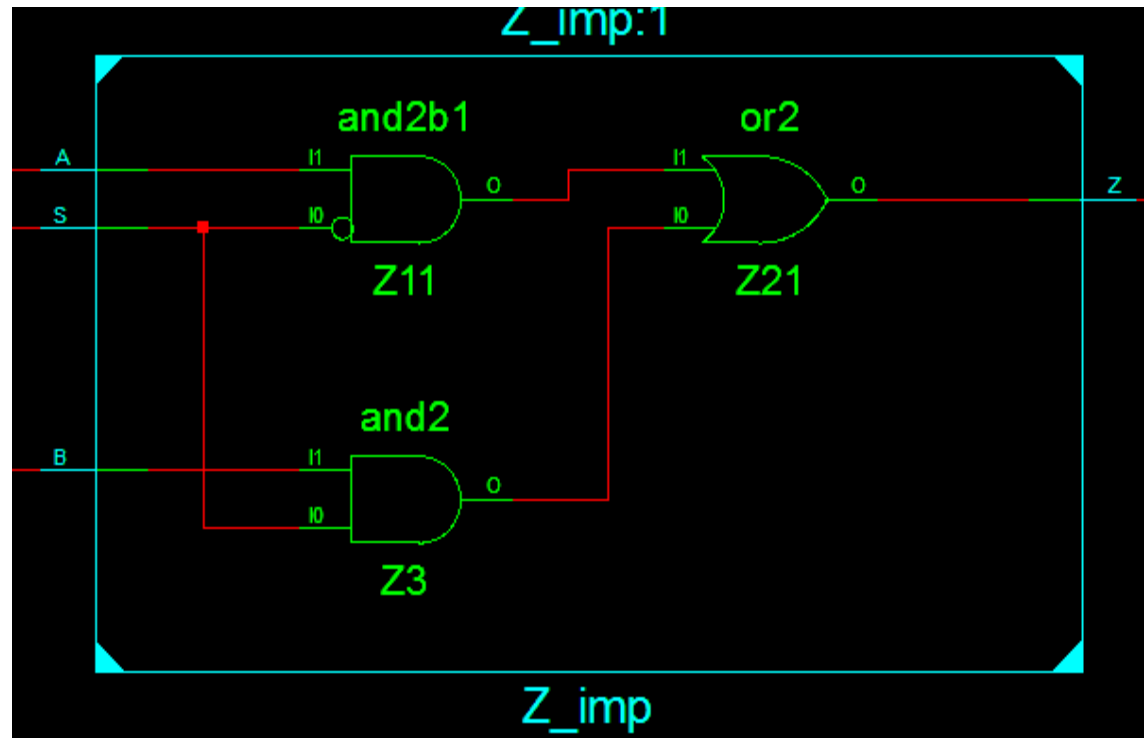
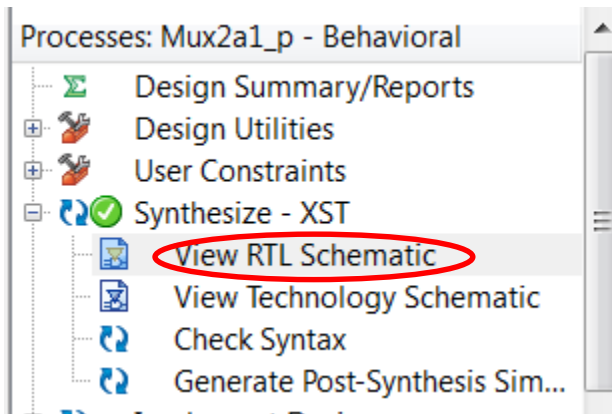
Señales!!

Todas se ejecutan a la vez !!  
(Concurrente)  
El orden no importa

# 1. Esquemas del diseño

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida.

Vista del esquema RTL (idéntico para ambas arquitecturas)



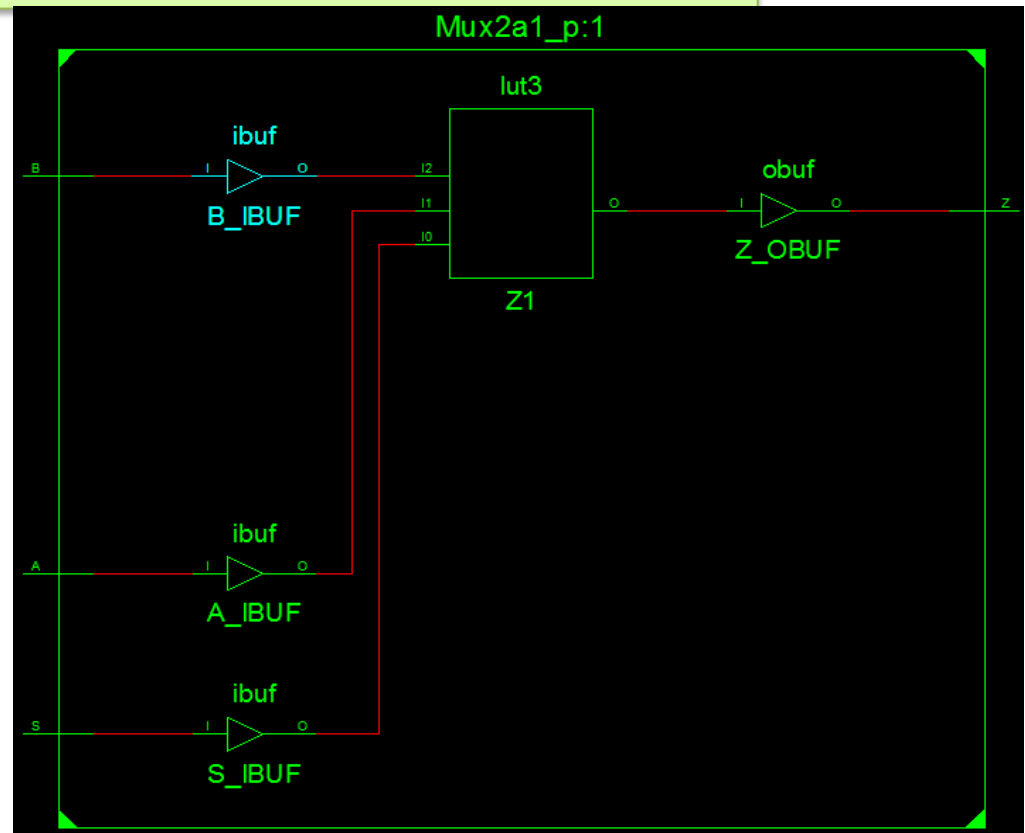
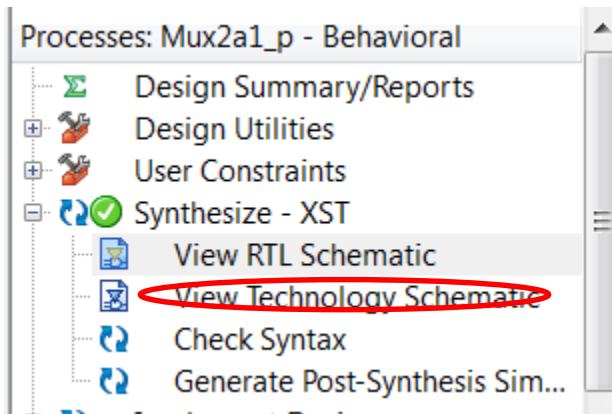
¿Como resuelve el sintetizador la descripción?



# Esquemas del diseño

Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit”, de dos entradas y una salida.

Vista del esquema tecnológico (idéntico para ambas arquitecturas)



¿Qué elementos de la FPGA usa?

## Descarga del diseño en Basys2

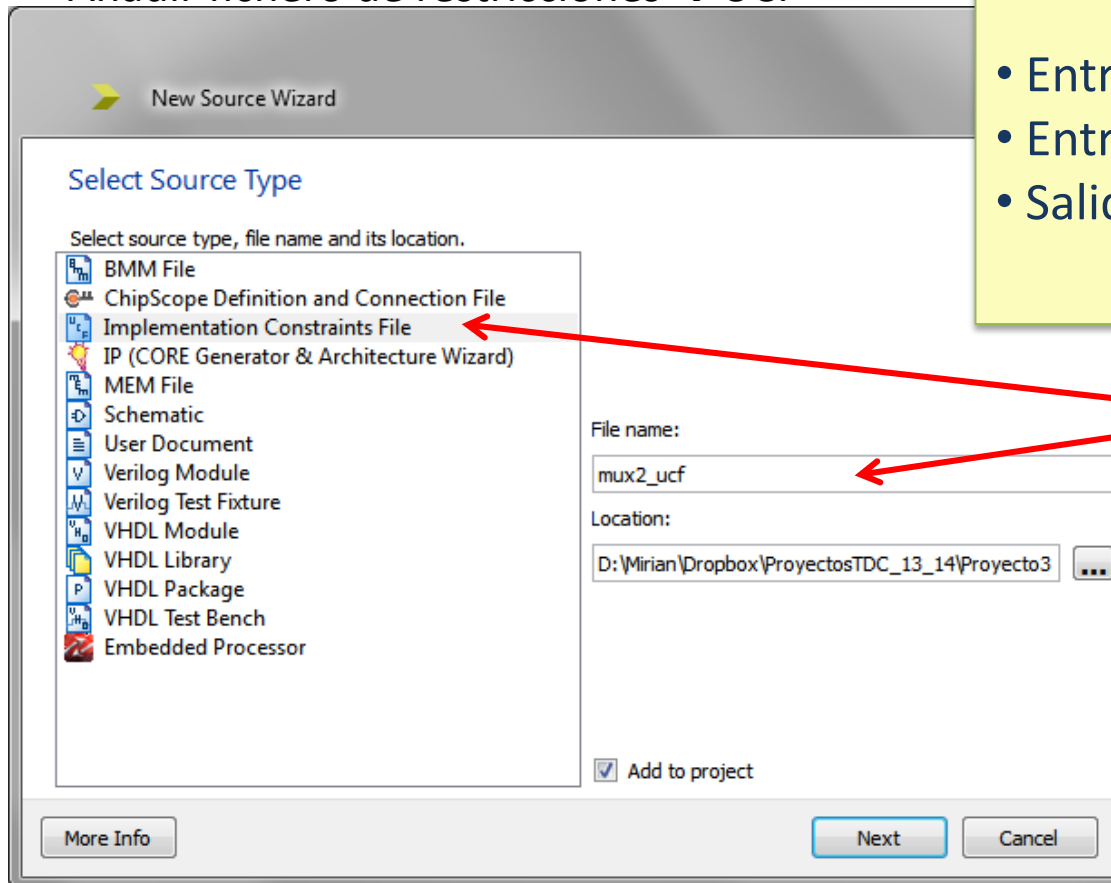
Proyecto03. Diseño de un multiplexor, “mux2\_df\_1bit” de dos entradas y una salida.

1. Añade al proyecto un fichero de restricciones para comprobar en la placa Basys2 el correcto funcionamiento del circuito. Utiliza interruptores para las entradas “A” y “B”, un led para la salida “Y” y un pulsador para la entrada de selección “Sel”

# Descarga del circuito a Basys2

## Fichero de restricciones

- Añadir fichero de restricciones → UCF



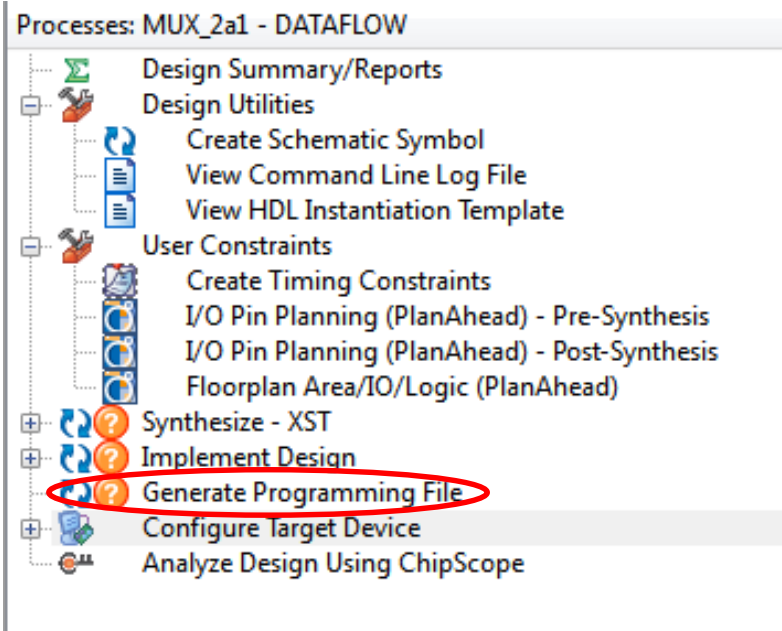
## Recursos:

- Entradas A y B → SW0 y SW1
- Entrada sel → Pulsador Btn0
- Salida Y → LED7.

# 1. Descarga del diseño en Basys2

## FLUJO DE DISEÑO CON FPGA:

- Síntesis
- Implementación con restricciones (UCF)
- Generación del fichero de programación (BIT)
- Configuración de la FPGA



## 2. Estilo de descripción Behavioral.

## 2. Estilo de descripción Behavioral

### a) Concurrentes:

#### ➤ Asignación de señales

- ✓ Simple o incondicional (  $\leq$  )
- ✓ Condicional (when-else)
- ✓ Selectiva (with –select- when)

#### ➤ Procesos

### b) Secuenciales (dentro de un proceso)

#### ➤ Asignación de señales

#### ➤ Control del flujo de ejecución

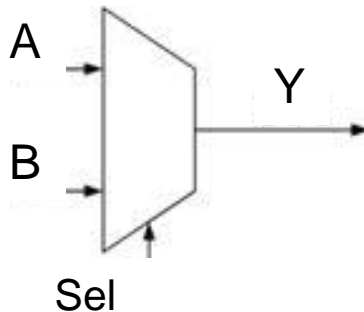
- ✓ If-then-else
- ✓ case

## 2.1. Behavioral: Sentencia concurrente condicional

## 2. Estilo de descripción Behavioral

**Proyecto03.** Diseño de un multiplexor, “mux2\_1bit”, de dos entradas y una salida.

Entidad



New Source Wizard

Define Module

Specify ports for module.

Module name: mux2\_1bit

Port Name	Direction	Bus	MSB	LSB
A	input	<input type="checkbox"/>		
B	input	<input type="checkbox"/>		
Sel	input	<input type="checkbox"/>		
Y	output	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		

Next Cancel

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

entity mux2_1bit is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          sel : in  STD_LOGIC;
          Y : out  STD_LOGIC);
end mux2 ;
```



## 2. Estilo de descripción Behavioral

**Proyecto03.** Diseño de un multiplexor, “mux2\_1bit”, de dos entradas y una salida.

### Arquitectura Behavioral 1ª

- Más cercana al lenguaje humano
- No necesita conocer las ecuaciones booleanas

A “Y” se le asigna “A” cuando S=’0’ y si no se le asigna “B”

```
architecture behavioral of mux2_1bit is
begin
    -- Descripción del MUX mediante
    -- sentencia concurrente condicional

    Y <= A when Sel='0' else B;

end behavioral ;
```

## 2. Estilo de descripción Behavioral

**Proyecto03.** Diseño de un multiplexor, “mux2\_1bit” de dos entradas y una salida.

2. Obtén los esquemas RTL y tecnológicos del diseño del multiplexor “mux2\_1bit” . ¿Hay alguna diferencia con la síntesis obtenida para las arquitecturas de estilo DATAFLOW?

Al incluir más de un módulo VHDL en el mismo proyecto, es necesario establecer como “Top Module” aquel con el que nos interese trabajar en cada momento.

(Selecciona el fichero “mux2\_1bit” y con el menú contextual haz click sobre “Set as Top Module”)

## 2.2. Behavioral: Sentencia secuencial condicional

## 2. Estilo de descripción Behavioral

**Proyecto03.** Multiplexor, “mux2\_1bit”, de dos entradas y una salida.

### Arquitectura Behavioral 2ª

```
architecture behavioral of mux2_1bit is
begin
    -- Descripción del MUX mediante
    -- sentencia concurrente condicional
        -- Y <= A when Sel='1' else B;
    -- Descripción con procesos

    process (A,B,Sel) ← Lista de sensibilidad
    begin
        if Sel='0' then
            Y<=A;
        else
            Y<=B;
        end if;
    end process;
end behavioral;
```

- Dentro de un proceso las sentencias se ejecutan secuencialmente
- Los procesos entre si son concurrentes
- Lista de sensibilidad**: todas las señales que se leen dentro del proceso.
- Se parece más a los lenguajes de alto nivel

## 2. Estilo de descripción Behavioral

Proyecto03. Diseño de un multiplexor, “mux2\_1bit” de dos entradas y una salida con estilo de descripción BEHAVIORAL.

3. Abrir el informe (*report*) de síntesis Determinar qué y cuántos elementos lógicos se han utilizado. (IOBs y *Slices*).
4. Añadir un fichero de restricciones especificando los periféricos a utilizar de Basys2 y comprobar el correcto funcionamiento del diseño (sirve el mismo fichero ya creado para las arquitecturas *dataflow*).
5. Abrir FPGA Editor para comprobar cómo se ha implementado el diseño sobre la FPGA.

## **2. 3. Ejercicios**

6. Realizar un **sumador binario completo de dos bits**. Determina y utiliza las ecuaciones booleanas para crear una arquitectura del tipo “DATAFLOW”.

- Nombre del Proyecto: **Proyecto04**
- Nombre del Módulo: **add\_1bit**
- Nombre de la arquitectura: **Dataflow**.
- Nombre de la entradas: **A, B, Cin**
- Nombre de las salidas: **Cout, Result**

## Recursos de Basys2:

- Sw1,Sw0 como entradas **A** y **B** y Sw3 como **Cin**
- Led7,Led0 como salidas **Cout** y **Result**

### **3. Tipos de datos (Arrays)**



## Definición de vectores en el fichero de restricciones

```
NET "I<0>"      LOC = "P11" ;  
NET "I<1>"      LOC = "L3" ;  
NET "I<2>"      LOC = "K3" ;  
NET "I<3>"      LOC = "B4" ;
```

# TDC\_Práctica1: Ejercicios

7. Realizar un **decodificador de 2 a 4** con entrada de **habilitación a nivel alto**.

Utilizar un **proceso**. (Referencia: Apartado 4.2 [3])

Nombre del Proyecto: **Proyecto05**



Nombre del módulo: **Deco2to4**

Nombres de las arquitectura: **Behavioral**

Nombre E/S y Recursos en placa:

- Sw1,Sw0 para la entrada: **I (1:0)**
- Sw3 como entrada **Enable**
- Led7,Led6,Led5,Led4 como salida: **S(3:0)**

8. Realizar un **codificador de 4 a 2** **con prioridad** y entrada de habilitación activa a nivel alto.

Cuando Enable sea 0, la salida deberá quedar en estado de alta impedancia (Z). También será el valor de salida por defecto. Utiliza **sentencias secuenciales**. (Referencia: Apartado 4.3 [3])

Nombre del Proyecto: **Proyecto06**



Nombre del Módulo: **Cod4to2\_Prior**

Nombre de la arquitectura: **Behavioral**

Nombre E/S y Recursos en placa:

- Sw3,Sw2,Sw1,Sw0 como entrada: **I (3:0)**
- Pulsador BTN3 como entrada **Enable**
- Led7,Led6 como salida **S(1:0)**

# TDC\_Práctica1: Ejercicios

9. Realiza el diseño de un **multiplexor de dos entradas de 4 bits de datos**.

(Referencia: apartado 3.3. de ([3]).

Terminado y comprobado añadir al mismo proyecto un módulo que describa un **multiplexor con dos entradas de 3 bits**.

Nombre del Proyecto: **Proyecto07**

Nombre de los Módulos: **Mux2\_4bits, Mux2\_3bits**

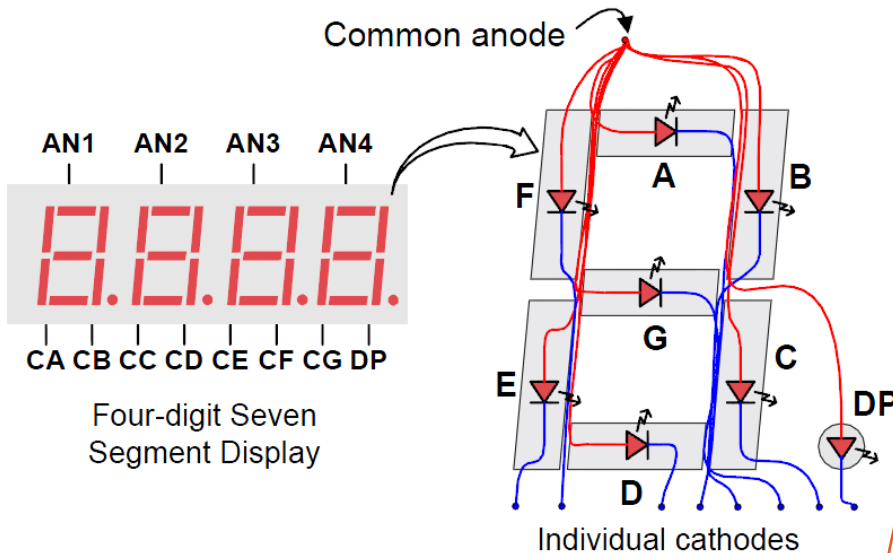
Nombre de las arquitecturas: **Behavioral**.

Nombre E/S y Recursos en placa:

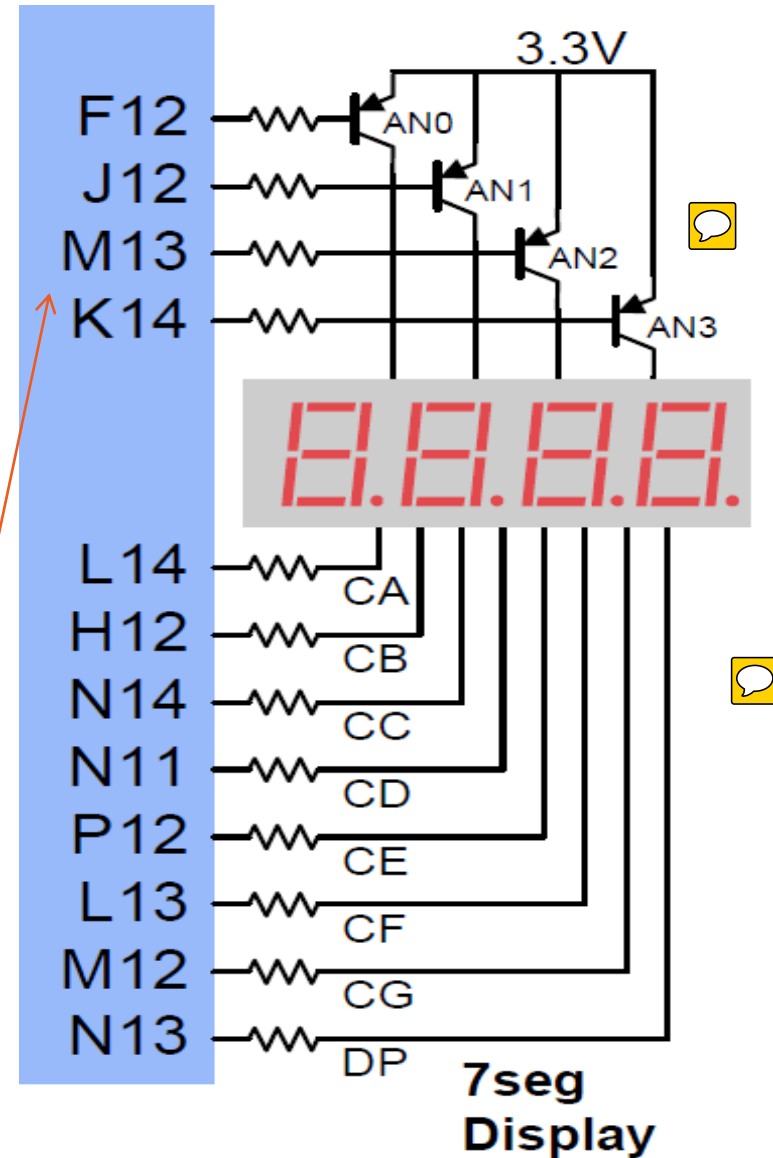
- Sw3,Sw2,Sw1,Sw0 como entrada **A(3:0)**
- Sw7,Sw6,Sw5,Sw4 como entrada **B(3:0)**
- BTN0 como entradas **Sel**
- LED3,LED2,LED1,LED0 como salida **Z(3:0)**

# TDC\_Práctica1: Ejercicios

## Displays de 7 segmentos de Basys2



**Error en la placa**



# TDC\_Práctica1: Ejercicios

10. Realiza un circuito que **muestre en uno de los 4 displays** el equivalente del valor binario introducido por cuatro interruptores. El **display** que lucirá será **seleccionado** mediante otros dos interruptores.

## Notas:

- Utilizar solo sentencias concurrentes.
- Ayudarse del manual de Basys2 y del apartado 4.1. del libro ([3]) para aprender a utilizar los displays de 7 segmentos .

Nombre del Proyecto: **Proyecto08**

Nombre del Módulo: **Disp7Seg**

Nombres de la arquitectura: **Behavioral**.

Nombre E/S y Recursos en placa:

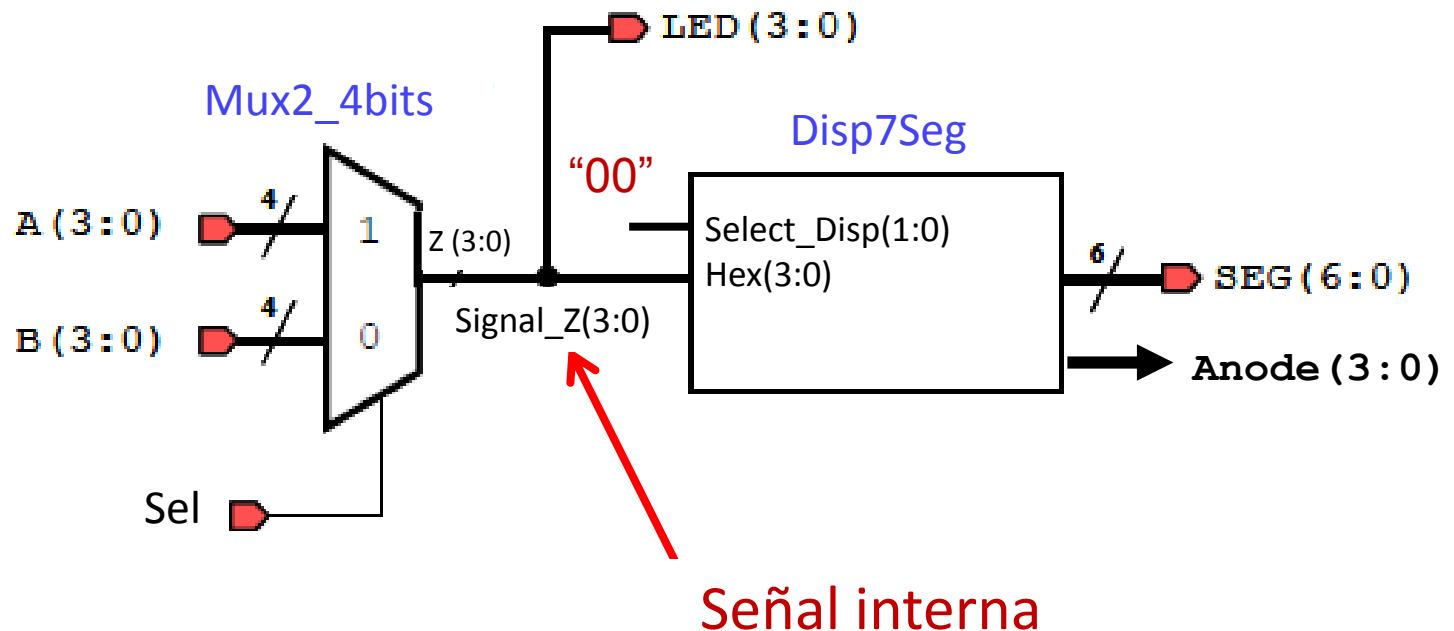
- Sw3,Sw2,Sw1,Sw0 como entradas **Hex(3:0)**
- Sw7 y Sw6 como selectores del display **Select\_Dis(1:0)**
- CA,CB....CG como **Seg(6:0)**
- AN3,AN2,A1 y AN0 como **Anode(3:0)**

Teoría

## 4. Diseño estructural

## 5. Descripción estructural

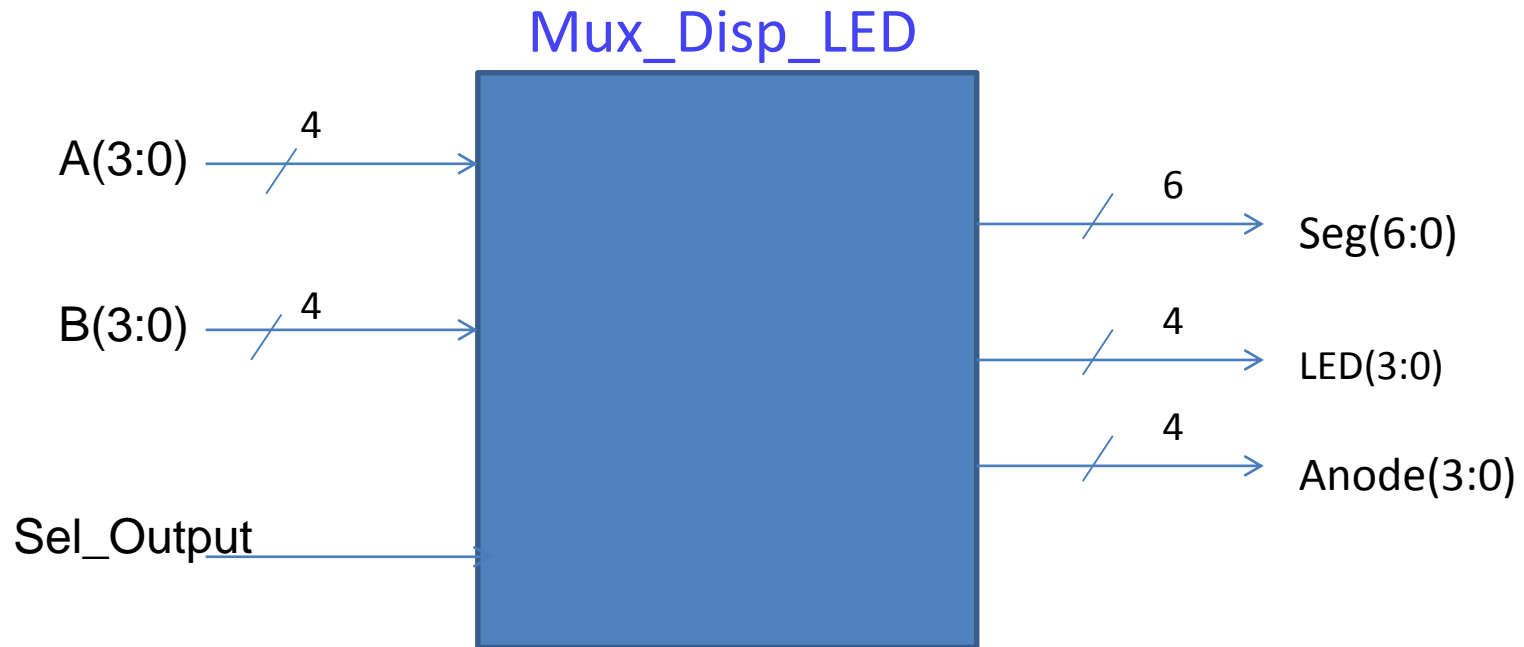
**Proyecto09.** Diseño de un circuito que muestre el valor seleccionado por un multiplexor de 2 entradas de 4 bits por los cuatro LED y además por un display de siete segmentos. ([3], Apartado 4.1)



## 4. Descripción estructural

**Proyecto09.** Diseño de un circuito que muestre el valor seleccionado por un multiplexor de 2 entradas de 4 bits por los cuatro LED y además por un display de siete segmentos (Display AN0). ([3], Apartado 4.1)

**Paso1.** Diagrama de bloques del nuevo módulo. Será la **entidad**.





## 4. Descripción estructural

**Proyecto09.** Diseño de un circuito que muestre el valor seleccionado por un multiplexor de 2 entradas de 4 bits por los cuatro LED y además por un display de siete segmentos. ([3], Apartado 4.1)

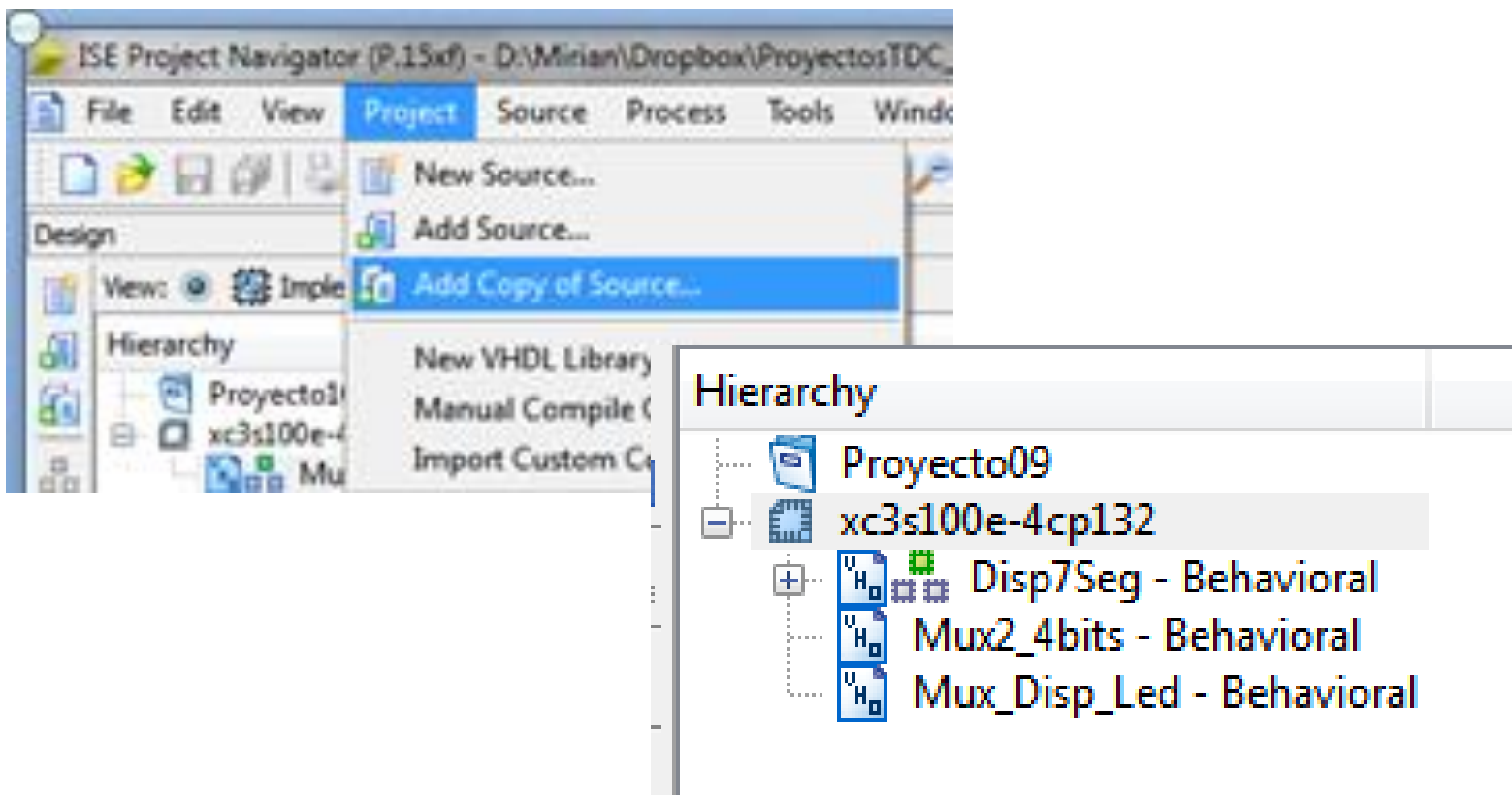
### Paso1. Describe la entidad

```
entity Mux_Displ_LED is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0) ;
           B : in STD_LOGIC_VECTOR (3 downto 0) ;
    Sel_Output : in STD_LOGIC ;
           Seg : out STD_LOGIC_VECTOR (6 downto 0) ;
           Anode : out STD_LOGIC_VECTOR (3 downto 0) ;
           LED : out STD_LOGIC_VECTOR (3 downto 0) ) ;
end Mux_Displ_LED ;
```

## 4. Descripción estructural

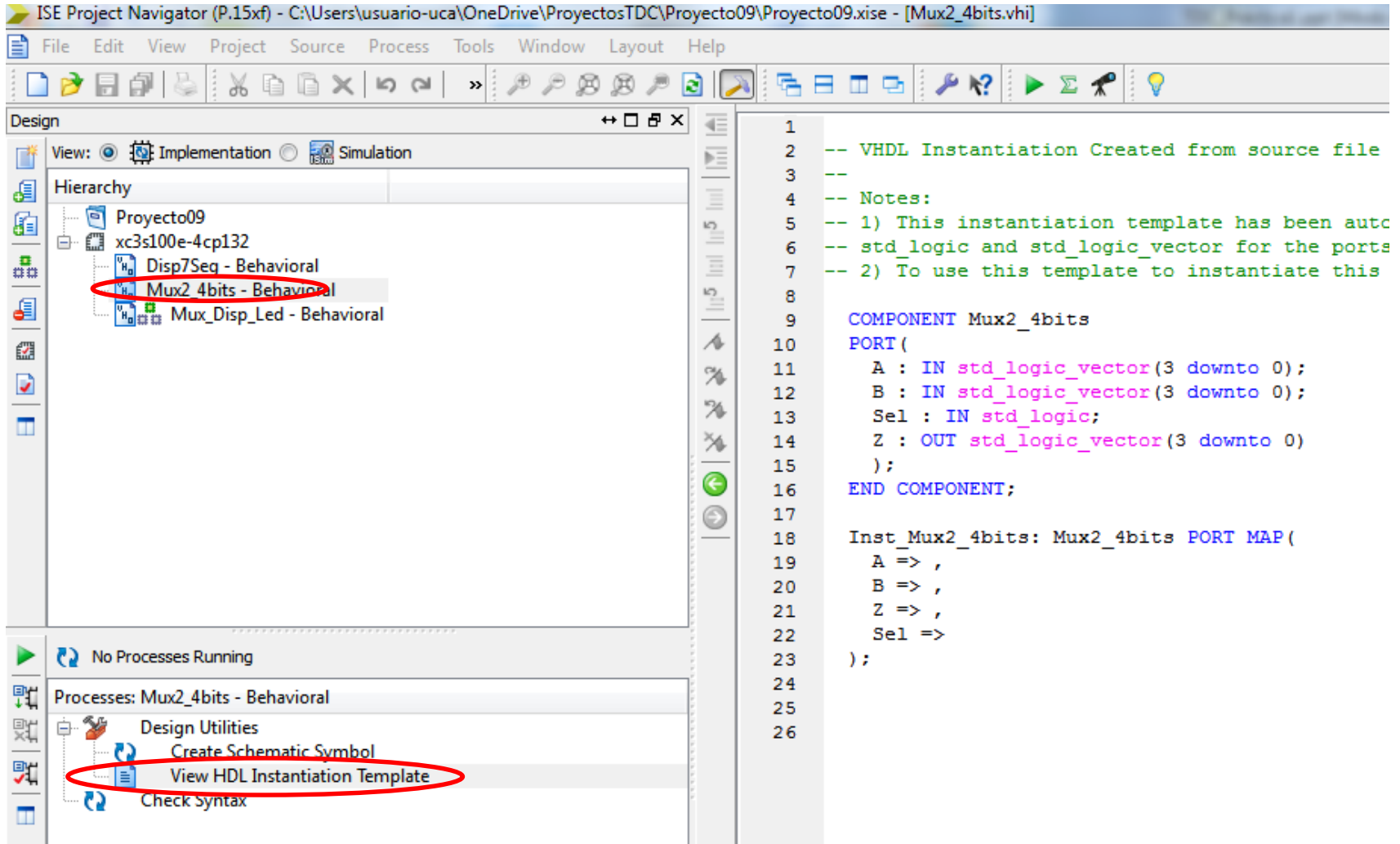
**Paso2.** Añadir y declarar los **componentes/módulos** de menor nivel usados en el diseño.

- Mux2\_4bits
- Disp7Seg



## 4. Descripción estructural

**Paso2.** Añadir y declarar los módulos de menor nivel usados en el diseño.



The screenshot shows the ISE Project Navigator interface. The top menu bar includes File, Edit, View, Project, Source, Process, Tools, Window, Layout, and Help. The Design window is active, showing the Hierarchy view. The hierarchy tree on the left lists the project structure: Proyecto09, xc3s100e-4cp132, Disp7Seq - Behavioral, Mux2\_4bits - Behavioral (highlighted with a red circle), and Mux\_Displ\_Led - Behavioral. The Processes window at the bottom shows the list of processes for Mux2\_4bits - Behavioral: Design Utilities, Create Schematic Symbol, View HDL Instantiation Template (highlighted with a red circle), and Check Syntax. The right pane displays the VHDL code for Mux2\_4bits.vhi, which includes a component declaration and an instantiation.

```
1
2 -- VHDL Instantiation Created from source file
3 --
4 -- Notes:
5 -- 1) This instantiation template has been auto
6 -- std_logic and std_logic_vector for the ports
7 -- 2) To use this template to instantiate this
8
9 COMPONENT Mux2_4bits
10 PORT (
11     A : IN std_logic_vector(3 downto 0);
12     B : IN std_logic_vector(3 downto 0);
13     Sel : IN std_logic;
14     Z : OUT std_logic_vector(3 downto 0)
15 );
16 END COMPONENT;
17
18 Inst_Mux2_4bits: Mux2_4bits PORT MAP (
19     A => ,
20     B => ,
21     Z => ,
22     Sel =>
23 );
24
25
26
```

## 4. Descripción estructural

```
architecture Structural of Mux_Displ_LED is
-----
-- LIST of COMPONENTS -
-----

COMPONENT Mux2_4bits
    PORT( A : IN std_logic_vector(3 downto 0);
          B : IN std_logic_vector(3 downto 0);
          Sel : IN std_logic;
          Z : OUT std_logic_vector(3 downto 0) );
END COMPONENT;

COMPONENT Disp7Seg
    PORT( Hex : IN std_logic_vector(3 downto 0);
          Select_Displ : IN std_logic_vector(3 downto 0);
          Anode : OUT std_logic_vector(3 downto 0);
          Seg : OUT std_logic_vector(6 downto 0) );
END COMPONENT;
-----

begin

end Structural;
```

## 5. Descripción estructural

**Proyecto09.** Diseño de un circuito que muestre el valor seleccionado por un multiplexor de 2 entradas de 4 bits por los cuatro LED y además por un display de siete segmentos. ([3], Apartado 4.1)

**Paso3.** Declarar las señales internas que interconectarán los módulos.

```
COMPONENT Disp7Seg
    PORT ( Hex : IN std_logic_vector(3 downto 0);
          Select_Dis: IN std_logic_vector(1 downto 0);
          Anode : OUT std_logic_vector(3 downto 0);
          Seg : OUT std_logic_vector(6 downto 0) );
END COMPONENT;

-----
-- LIST OF SIGNALS
-----

signal signal_Z : std_logic_vector(3 downto 0);

Begin
end Structural;
```

**Dummy signal**

## 4. Descripción estructural

**Proyecto09.** Diseño de un circuito que muestre el valor seleccionado por un multiplexor de 2 entradas de 4 bits por los cuatro LED y además por un display de siete segmentos. ([3], Apartado 4.1)

**Paso 4:** Instanciar y “mapear” los módulos declarados para conseguir el diseño de mayor jerarquía

```
1
2  -- VHDL Instantiation Created from source file Mux2_4bits.vhd -- 12:05:54 09/19/2013
3  --
4  -- Notes:
5  -- 1) This instantiation template has been automatically generated using types
6  -- std_logic and std_logic_vector for the ports of the instantiated module
7  -- 2) To use this template to instantiate this entity, cut-and-paste and then edit
8
9  COMPONENT Mux2_4bits
10 PORT(
11     A : IN std_logic_vector(3 downto 0);
12     B : IN std_logic_vector(3 downto 0);
13     Sel : IN std_logic;
14     Z : OUT std_logic_vector(3 downto 0)
15 );
16 END COMPONENT;
17
18 Inst_Mux2_4bits: Mux2_4bits PORT MAP(
19     A => ,
20     B => ,
21     Z => ,
22     Sel =>
23 );
24
25
26
```

## 4. Descripción estructural

**Paso 4:** Instanciar y “mapear” los módulos declarados para conseguir el diseño de mayor jerarquía

```
begin
MUX_U0: Mux2_4bits PORT MAP (
    A => ,
    B => ,
    Z => ,
    Sel => );

Disp7Seg_U0: Disp7Seg PORT MAP (
    Hex => ,
    Select_Disg => ,
    Anode => ,
    Seg => );
end Structural;
```

## 4. Descripción estructural

**Paso 4:** Instanciar y “mapear” los módulos declarados para conseguir el diseño de mayor jerarquía

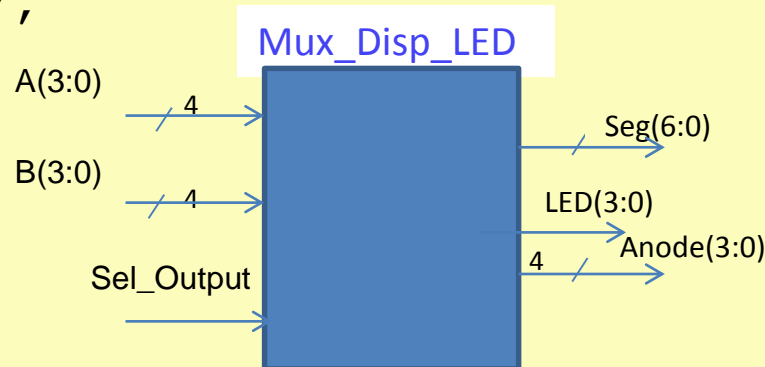
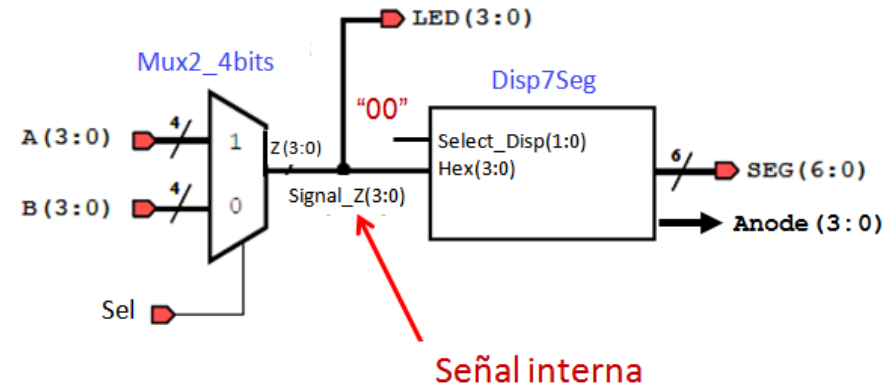
**begin**

```
MUX_U0: Mux2_4bits PORT MAP (  
    A => A ,  
    B => B ,  
    Z => signal_Z ,  
    Sel => Sel_Output ) ;
```

```
Disp7Seg_U0: Disp7Seg PORT MAP (  
    Hex => signal_Z ,  
    Select_Displ => "00" ,  
    Anode => Anode ,  
    Seg => Seg) ;
```

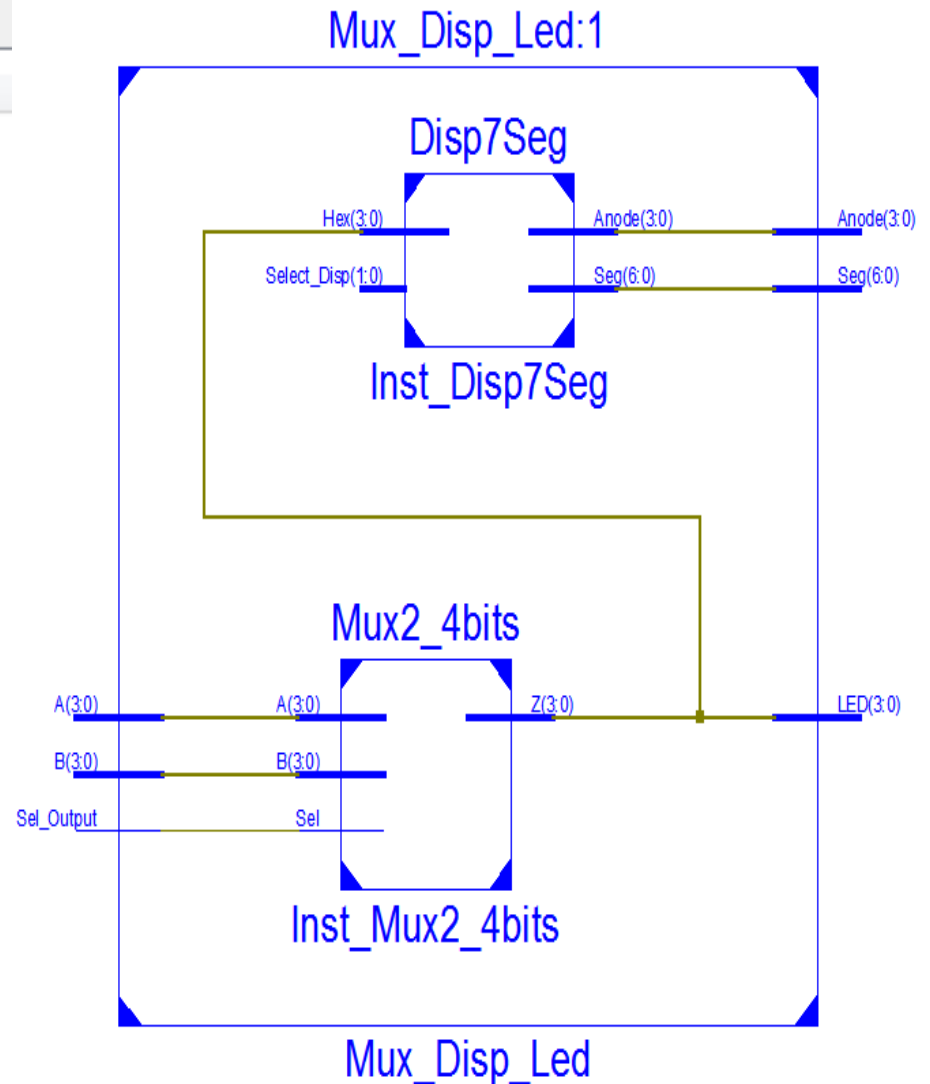
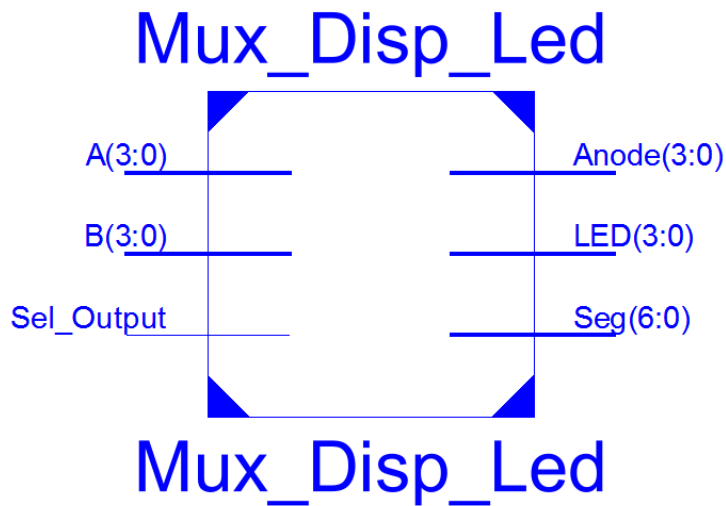
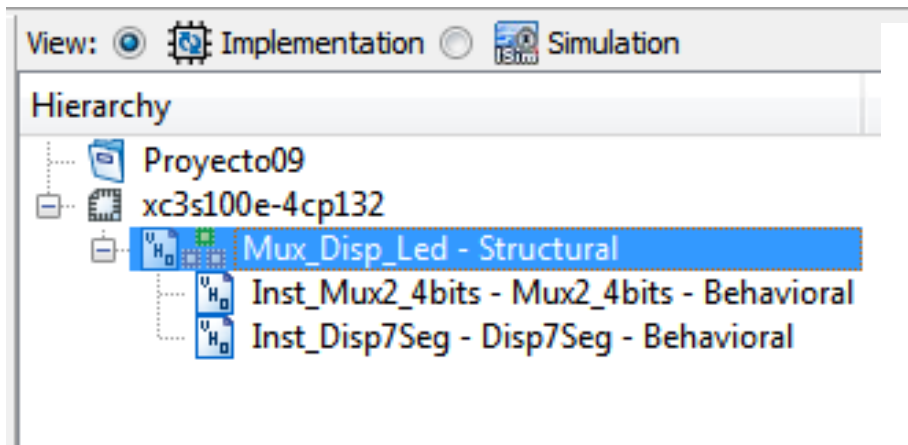
**LED<=signal\_Z;**

**end Structural;**






## 4. Descripción estructural



## 4. Descripción estructural

16. **Proyecto09.** Añadir el fichero de restricciones para comprobar el funcionamiento en Basys2.
17. **Proyecto09.** Averigua cuántos LUTs y cuantos Slices son utilizados en este diseño. 

## 4.1. Ejercicios de Diseño estructural

## TDC\_Práctica1: Ejercicios

18. **Proyecto10**. Usando descripción estructural, diseña una ALU para dos operandos de 1 bit.

Las operaciones a realizar son: AND, XNOR, SUMA, Paso transparente de A.

Los módulos de los operadores AND, XNOR y SUMA ya están diseñado en los **proyectos 01, 02 y 04** respectivamente. Añade otros que consideres necesarios.

Comprobar su funcionamiento en Basys2.

Nombre del Proyecto: **Proyecto10**

Nombre del Módulo Top: **ALU\_1bit**

Nombres de la arquitectura: **Structural**

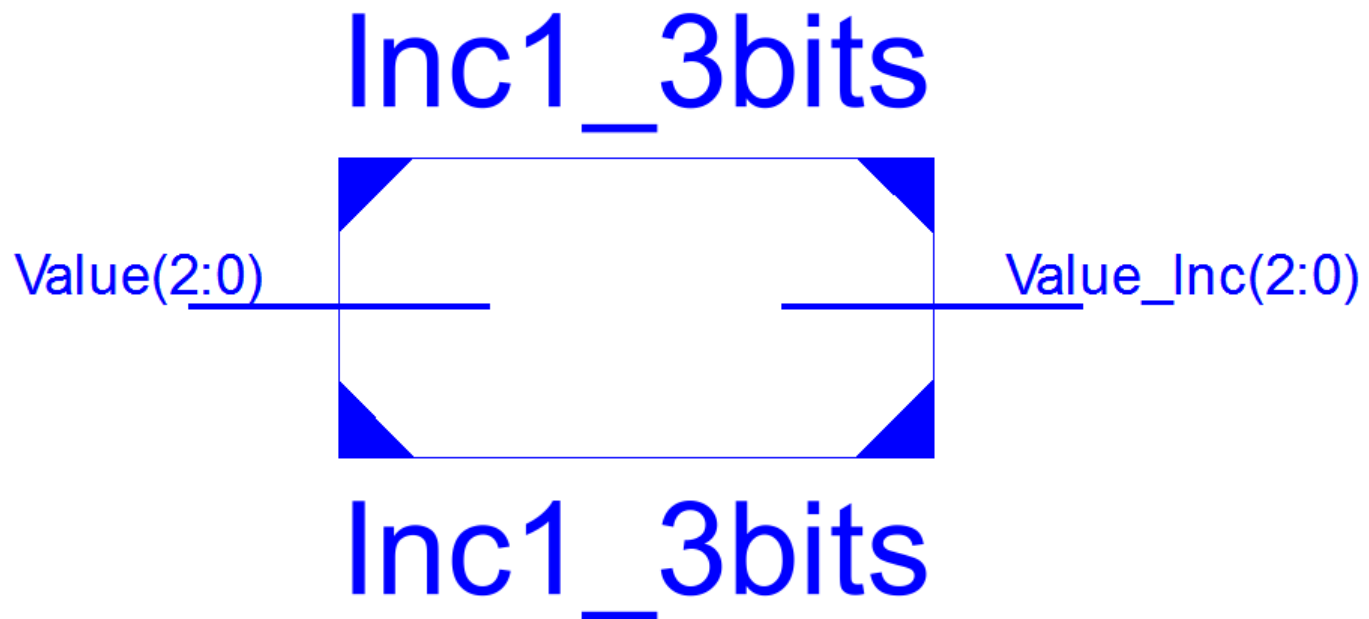
Recursos:

- Sw7-Sw5 como entradas: **A y B, Carry\_In**
- Sw1-0 como entrada de selección de la operación: **Sel\_Ope(1:0)**
- Leds LD7, LD6 para las salidas: **Carry\_Out, Result**

## **5. Tipo de datos Unsigned y Signed. Operaciones aritméticas**

## TDC\_Práctica1: Ejercicios

19. **Proyecto11**. Realizar el diseño de un circuito que incremente en 1 el valor de entrada de un operando de 3 bits. (*Inc1\_3bits*) Utilizar el tipo unsigned y el operador “+”. Comprobar el funcionamiento en Basy2.

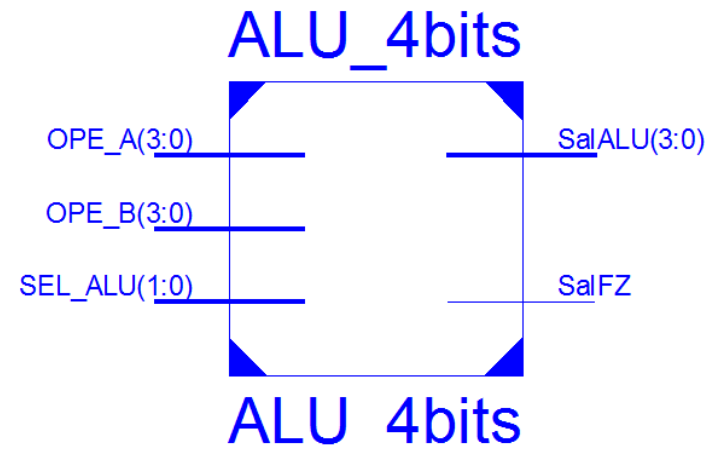


## TDC\_Práctica1: Ejercicios

20. **Proyecto12.** Realizar el diseño de una **ALU** mediante descripción **behavioral**. Los **operandos** de entrada serán de **4 bits**. Las operaciones y su código de selección correspondiente se detallan en la tabla de más abajo. La ALU contará con una salida que actuará como señalizador de cero (**SalFZ**). Cuando el resultado de una operación sea cero, deberá tomar valor '1'. Utilizar el **tipo unsigned** y los **operadores aritméticos** “+” y “-”.

Comprobar el funcionamiento en Basys2.

SEL_ALU(1)	SEL_ALU(0)	Funcion
0	0	Paso de A
0	1	-
1	0	A + B
1	1	A - B



## Bibliografía del tema

[1] Manual de la placa de evaluación Basys2

[http://www.digilentinc.com/Data/Products/BASYS2/Basys2\\_rm.pdf](http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf).

[2] Datasheet de la FPGA Spartan3E de Xilinx

[http://www.xilinx.com/support/documentation/data\\_sheets/ds312.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf)

[3] Diseño de circuitos digitales con VHDL

<http://eciencia.urjc.es/handle/10115/4045>