

# Tutorial Xilinx-ISE

## Manejo básico

**Susana Holgado**  
**Escuela Politécnica Superior**  
**UAM**

Modificaciones:



Alberto Sánchez (2012, 2014)  
Ángel de Castro (2006, 2014)  
Francisco Javier Gómez Arribas (2008)  
Víctor Apéstegui Palacio (2009)  
Fernando Jesús López Colino (2010)  
Fernando Barbero (2012)



# Xilinx-ISE

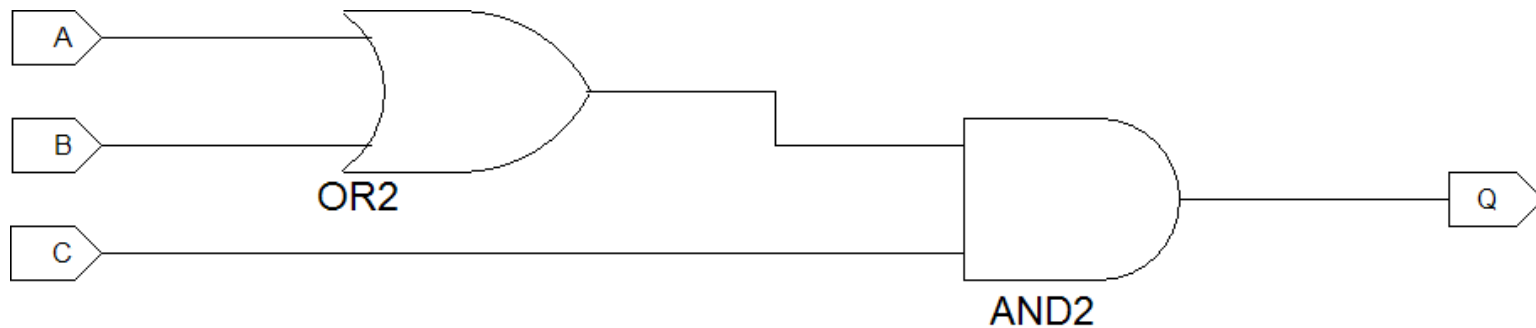
La herramienta **Xilinx-ISE** (*Integrated Software Environment*) es una herramienta de diseño de circuitos profesional que nos va a permitir, entre otras funciones, la introducción del diseño (mediante esquemas o un lenguaje HDL) y su posterior simulación.

En este tutorial se va a utilizar concretamente:

-  **Entorno ISE:** donde se realizará el diseño del circuito, utilizando un lenguaje específico de diseño (VHDL)
-  **ModelSim:** donde podrá realizarse la simulación del funcionamiento del circuito y de este modo comprobar si cumple con las especificaciones establecidas



Se describirá a continuación el funcionamiento de las herramientas, y para ello se realizará el diseño de un circuito lógico que cumpla el siguiente esquema:



$$Q = (A \text{ or } B) \text{ and } C$$



# 1. Creación de un nuevo proyecto

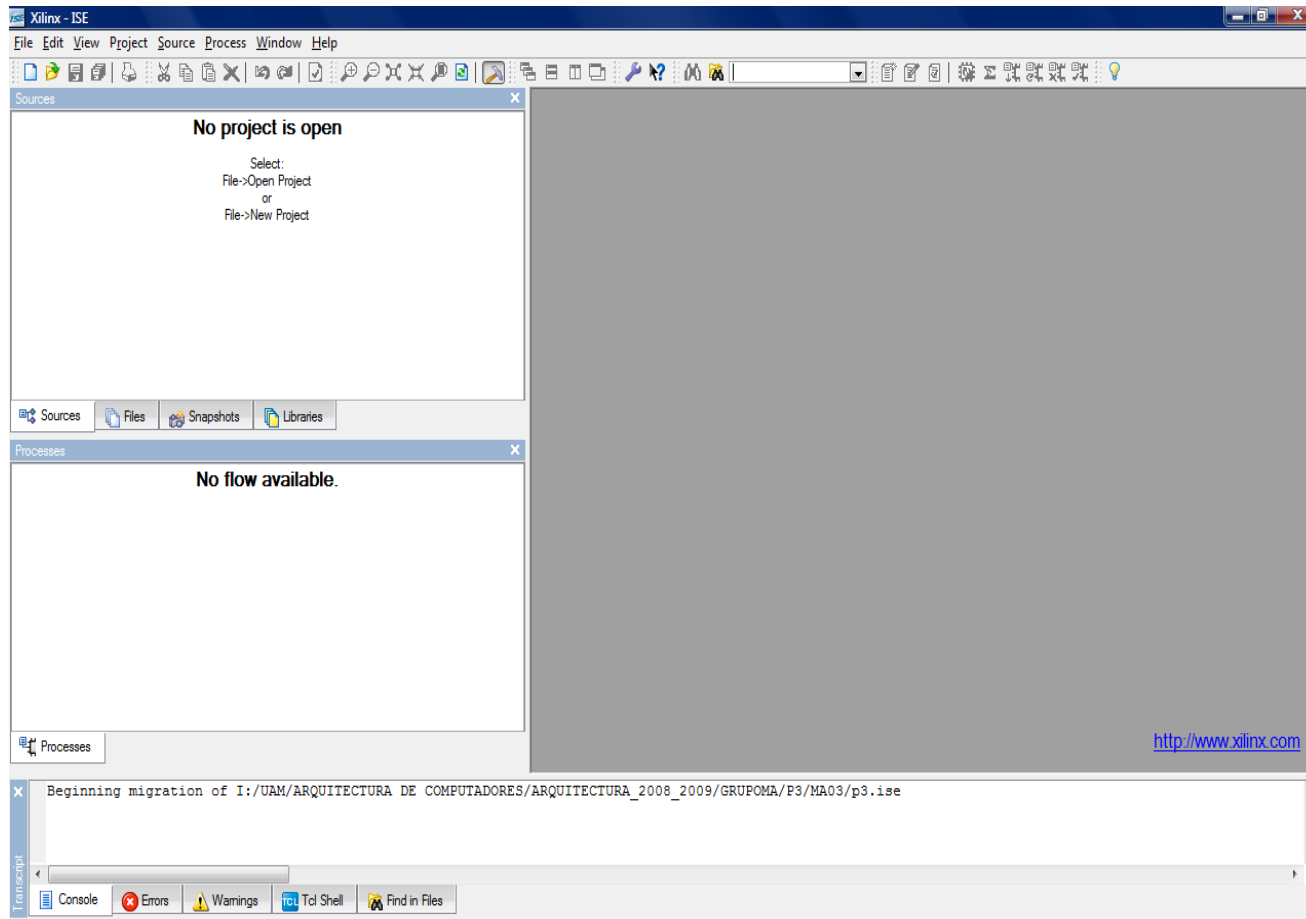
Un proyecto es un conjunto de ficheros de diseño, tales como esquemas, ficheros de texto con código fuente escrito en un Lenguaje de *Descripción de Hardware* (*HDL* en inglés), listas de conexionado (“netlists”), bibliotecas de componentes, vectores de test para la simulación, etc., seleccionados para un diseño específico

Nada más acceder al programa, aparecerá por pantalla una ventana como la que se muestra en la siguiente figura, y que da acceso al programa de diseño

Si hubiera ya algún proyecto cargado, se puede cerrar seleccionando la opción de menú *File → Close Project*



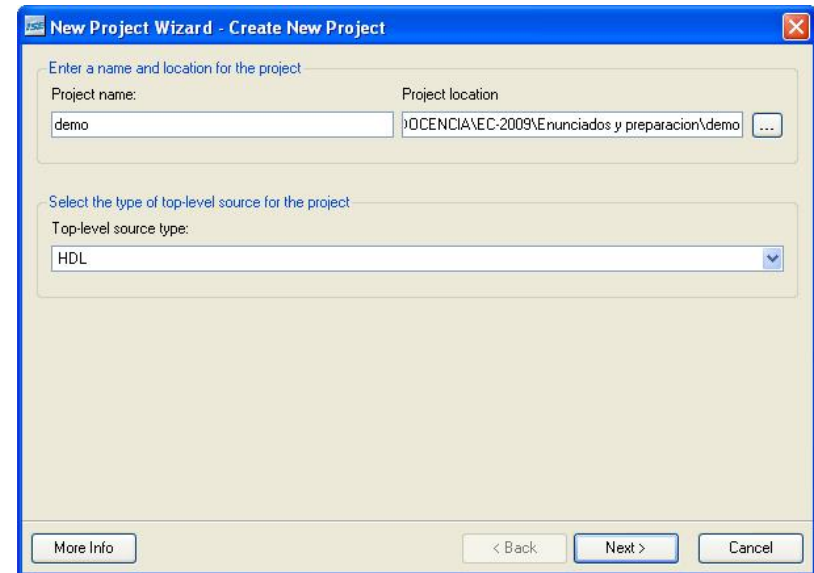
# 1. Creación de un nuevo proyecto



# 1. Creación de un nuevo proyecto

Para crear un nuevo proyecto:

- Seleccionar *File → New Project*
- En la ventana de diálogo de New Project indicar el directorio de ubicación del proyecto en *Project Location*  
Colocar el proyecto en D:\ y un directorio sin espacios
- Añadir el nombre (p. ej. “tutorial”) en *Project Name*
- En la opción *Top-Level Module Type* se elige “HDL” porque se va a utilizar un lenguaje de descripción del hardware como VHDL
- Automáticamente se crea un subdirectorio en la ruta indicada en *Project Location* con el nombre del proyecto (p. ej. “tutorial”), y donde se almacenará todo lo relacionado con este proyecto



# 1. Creación de un nuevo proyecto

En la segunda pantalla hay que seleccionar la FPGA sobre la que se va a sintetizar el diseño. Para este caso se seleccionará:

- *Product Category:* General Purpose
- *Device Family:* Spartan3
- *Device:* XC3S200
- *Package:* FT256
- *Speed grade:* -4
- *Synthesis Tool:* XST (VHDL/Verilog)
- *Simulator:* Modelsim SE VHDL (herramienta de simulación)
- *Preferred Language:* VHDL (lenguaje asociado para los ficheros que crea la herramienta)

Property Name	Value
Product Category	General Purpose
Family	Spartan3
Device	XC3S200
Package	FT256
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Modelsim-SE VHDL
Preferred Language	VHDL
Enable Enhanced Design Summary	<input checked="" type="checkbox"/>
Enable Message Filtering	<input type="checkbox"/>
Display Incremental Messages	<input type="checkbox"/>

More Info      < Back      Next >      Cancel

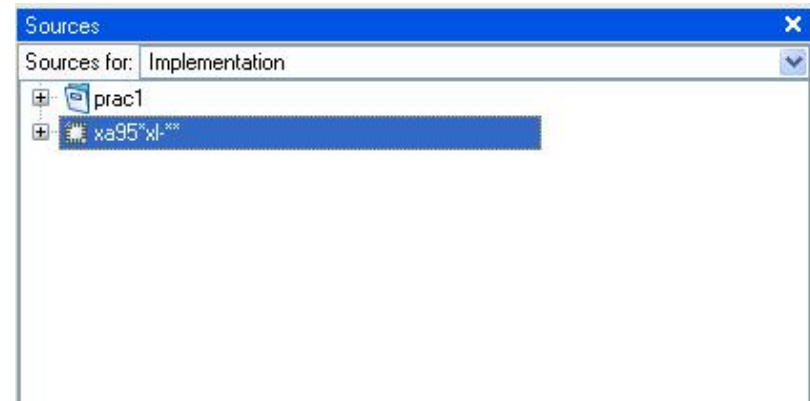


# 1. Creación de un nuevo proyecto

Después se pasan las dos siguientes pantallas eligiendo *Next* y al dar a *Finish*, ISE crea y muestra el nuevo proyecto.

Se observarán cambios con respecto al aspecto inicial de la ventana en la parte izquierda, en *Sources*, donde aparece el proyecto creado.

El paso siguiente será el diseño del circuito que responda a la funcionalidad indicada al comienzo del tutorial.





## 2. Diseño del circuito.

- Seleccionar *File* → *New*
- Seleccionar *Text File*



## 2. Diseño del circuito. Incluir librerías.

- En VHDL se debe definir el conjunto de librerías con el que se quiere trabajar.

```
library IEEE;  
use IEEE.std_logic_1164.ALL;  
use IEEE.std_logic_arith.ALL;  
use IEEE.std_logic_unsigned.ALL;
```

- Copiar este código en el fichero en blanco.
- Seleccionar *File* → *Save*
- Nombrar el fichero “P1e1.vhd”



## 2. Diseño del circuito. Definir la Entidad.



Cuando definimos la entidad, estamos describiendo las entradas y salidas del sistema.

```
entity P1e1 is  
  port (  
    A : in std_logic;  
    B : in std_logic;  
    C : in std_logic;  
    Q : out std_logic  
  );  
end P1e1;
```



## 2. Diseño del circuito. Describir la arquitectura

```
architecture Practica of P1e1 is
```

```
    signal s : std_logic;
```

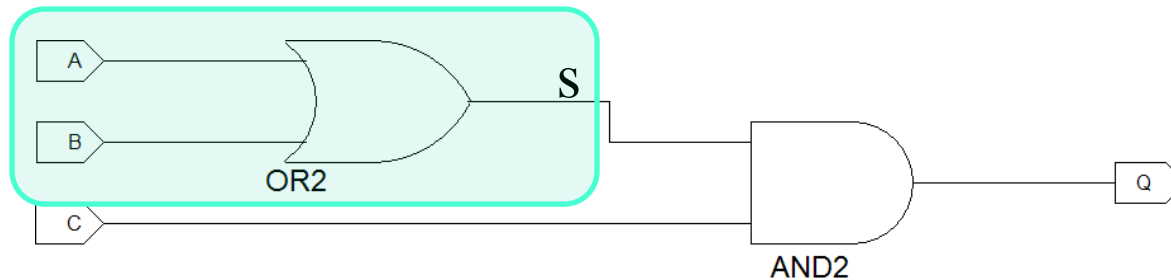
```
begin
```

```
    ...
```

```
end Practica;
```



## 2. Diseño del circuito. Describir la arquitectura



architecture Practica of P1e1 is

signal s : std\_logic;

begin

...

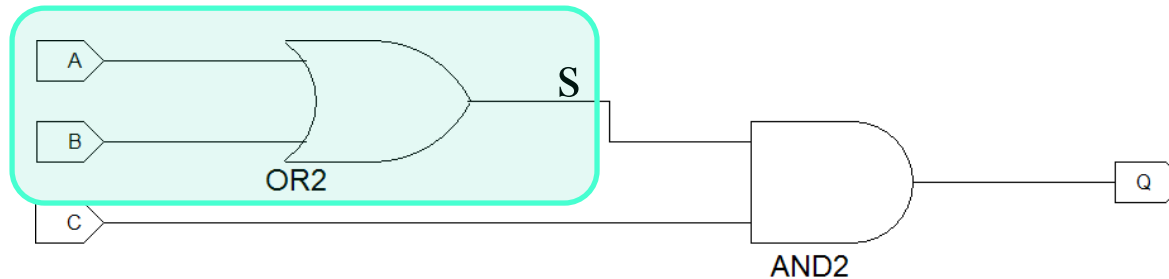
end Practica;



Utilizamos una señal auxiliar dentro de la arquitectura



## 2. Diseño del circuito. Describir la arquitectura



```

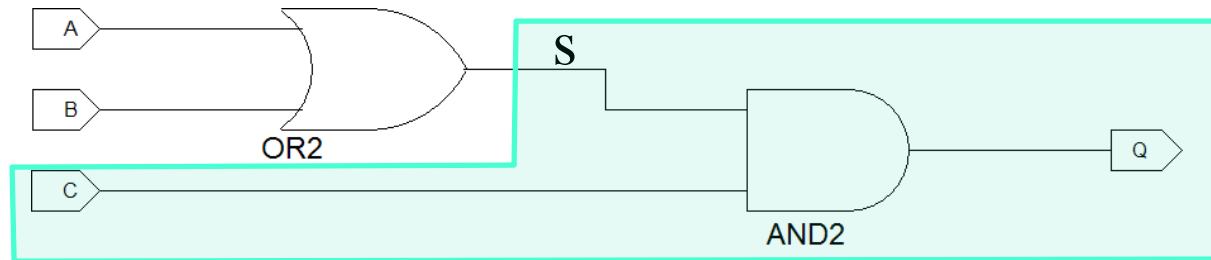
process (A, B)
begin
  if A = '1' or B='1' then
    s <= '1';
  else
    s <= '0';
  end if;
end process;
  
```

→ Lista de sensibilidad: cada cambio de uno de estos elementos hace que se evalúe el proceso.

→ Asignación del valor de salida.



## 2. Diseño del circuito. Describir la arquitectura



$Q \leq s \text{ and } C;$   $\longrightarrow$  Asignación del valor de salida.



## 2. Diseño del circuito. Describir la arquitectura

```
architecture Metodo1 of P1e1 is
    signal s : std_logic;
begin
    process (A, B)
    begin
        if A = '1' or B = '1' then
            s <= '1';
        else
            s <= '0';
        end if;
    end process;
    Q <= s and C;
end Metodo1;
```

```
architecture Metodo2 of P1e1 is
    signal s : std_logic;
begin
    S <= A or B;
    Q <= s and C;
end Metodo2;
```

```
architecture Metodo3 of P1e1 is
begin
    Q <= (A or B) and C;
end Metodo3;
```

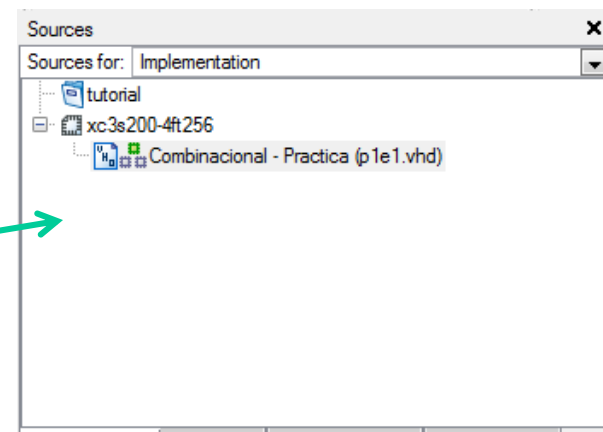
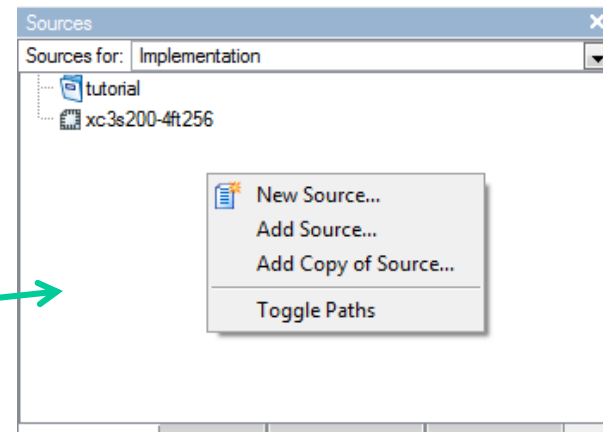
```
architecture Metodo4 of P1e1 is
    signal s : std_logic;
begin
    process (A, B)
    begin
        if A = '1' or B = '1' then
            s <= '1';
        else
            s <= '0';
        end if;
    end process;
    process (s, C)
    begin
        if s = '1' and C = '1' then
            Q <= '1';
        else
            Q <= '0';
        end if;
    end process;
end Metodo4;
```

Todos los métodos tienen  
la misma funcionalidad



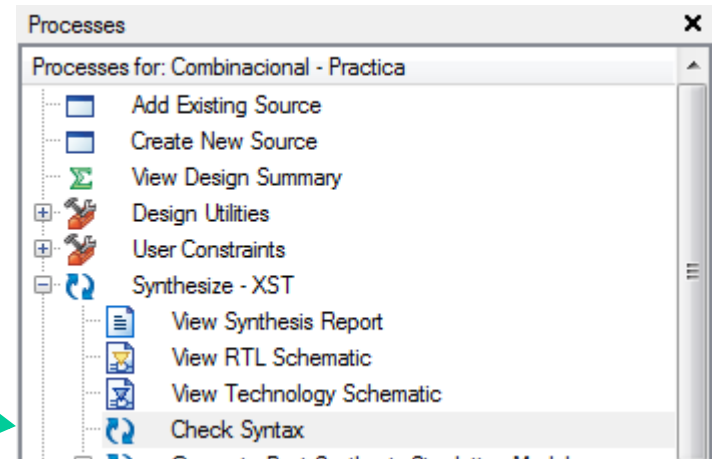
### 3. Salvar el fichero y agregarlo al proyecto.

- Seleccionar *File* → *Save*
- En el panel *Sources* hacer clic con el botón derecho del ratón, indicar *Add Source* y seleccionar el fichero “P1e1.vhd”.
- Dar a OK en el cuadro de diálogo que aparece.
- Al hacer esto se actualiza el árbol de jerarquía en el panel *Sources*.

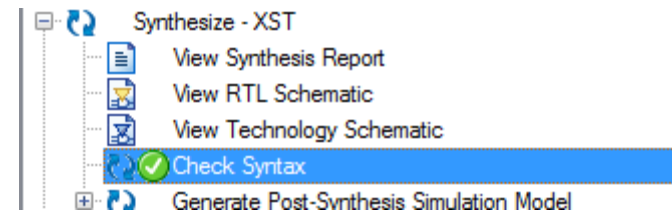


## 4. Verificar Sintaxis.

- Hacer clic con el botón izquierdo del ratón sobre la línea del panel *Sources* que hace referencia a nuestro fichero fuente, para seleccionarlo.
- En el panel *Processes*, hacer doble clic sobre *Check Syntax*, en la sección *Synthesize – XST*.

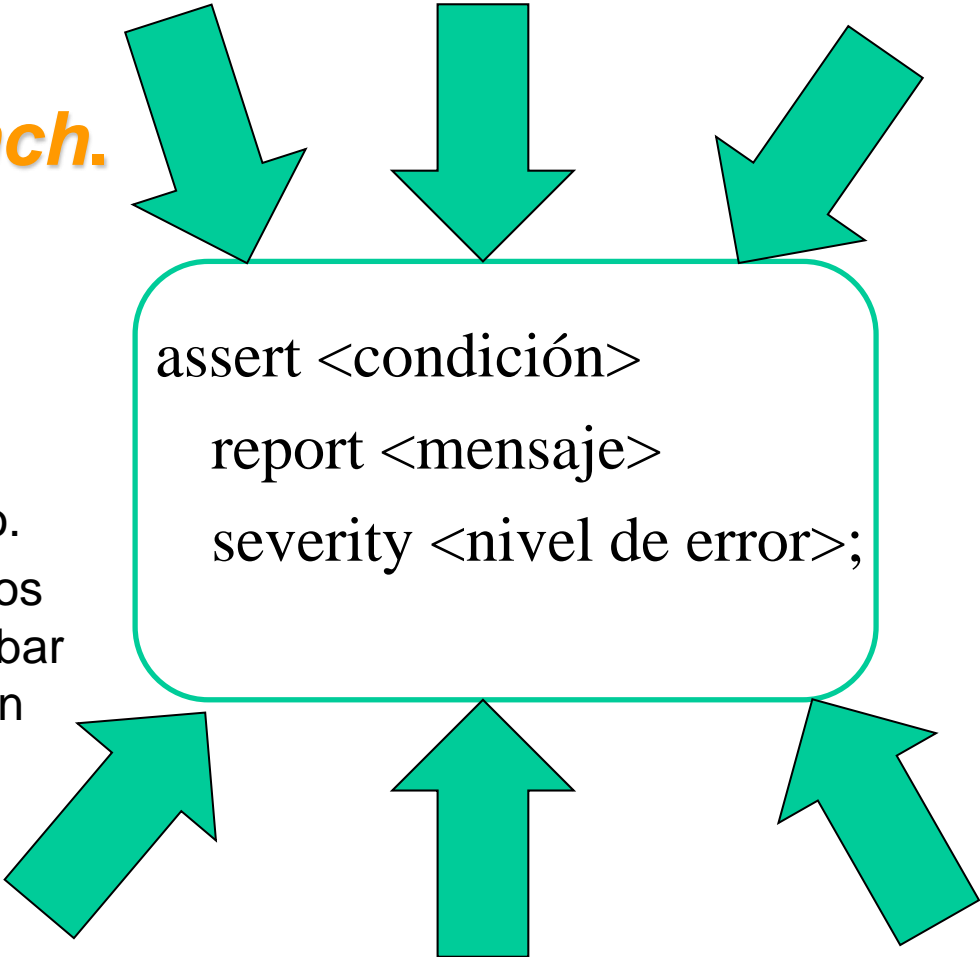


- Si todo está bien, aparecerá lo siguiente:
- Si hay errores o *warnings*, aparecen abajo, en las solapas correspondientes del panel *Transcript*.



## 5. Generar *Testbench*.

- El *testbench* es un set de pruebas que se utiliza para verificar el correcto funcionamiento de un diseño.
- El *testbench* debe cubrir todos los casos posibles y comprobar que las salidas obtenidas son iguales a las esperadas.



```
assert <condición>  
  report <mensaje>  
  severity <nivel de error>;
```



## 5. Generar *Testbench*.

```
library IEEE;  
use IEEE.std_logic_1164.ALL;  
use IEEE.std_logic_arith.ALL;  
use IEEE.std_logic_unsigned.ALL;
```

```
entity P1e1_tb is  
end P1e1_tb;
```

- Copiar el código en otro fichero nuevo.
  - Seleccionar *File* → *Save*
  - Nombrar el fichero “p1e1\_tb.vhd”.
- 
- En el panel *Sources*, indicar *Add Source* y seleccionar el fichero “P1e1\_tb.vhd”.



## 5. Generar Testbench (II)

architecture Practica of P1e1\_tb is

component P1e1

port(

A : in std\_logic;

B : in std\_logic;

C : in std\_logic;

Q : out std\_logic

);

end component;

-- Entradas

signal sigA : std\_logic := '0';

signal sigB : std\_logic := '0';

signal sigC : std\_logic := '0';

-- Salidas

signal q : std\_logic;

constant ESPERA : time := 10 ns;

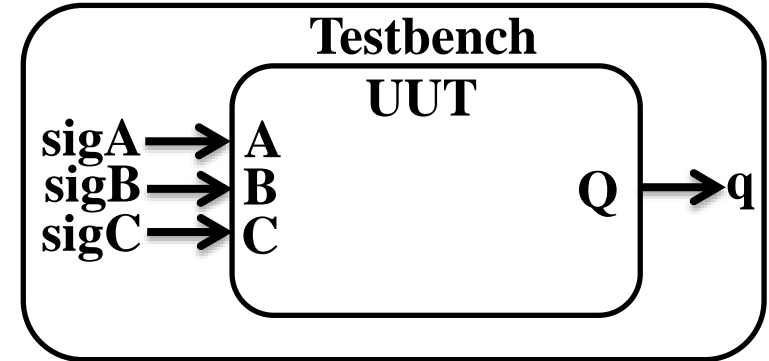
Definir el componente que vamos a probar. Describir sus entradas y salidas.

Definir las señales que vamos a utilizar para probar el sistema.

Definir constantes.



## 5. Generar Testbench (III)



begin

```

uut: P1e1 port map (
  A => sigA,
  B => sigB,
  C => sigC,
  Q => q
);
  
```

Pueden llamarse igual

Nombre de esta instancia.  
Podría tener varias.  
UUT: *Unit Under Test*

Asignar las señales del  
*testbench* a las entradas y  
salidas del módulo a probar.

Señal del  
módulo  $\Rightarrow$  Señal del  
*testbench*



## 5. Generar Testbench (IV)

-- Proceso de estímulos

process

begin

sigA <= '0'; sigB <= '0'; sigC <= '0';

—————→ Definir las entradas.

wait for ESPERA;

—————→ Esperar a que las salidas se actualicen.

assert q = '0'

report "Error en el caso" & std\_logic'image(sigA) & std\_logic'image(sigB) & std\_logic'image(sigC)

severity failure;

—————→ Comprobar el valor de la salida.

-- Continuar con todos los casos restantes

wait;

—————→ Detener la ejecución del proceso.

end process;

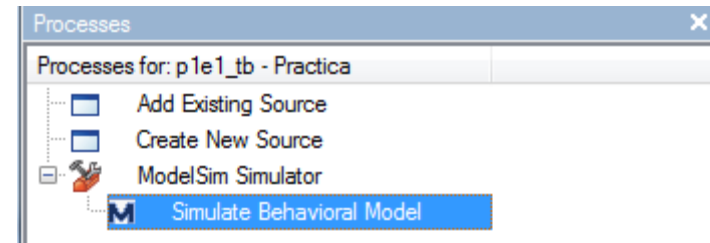
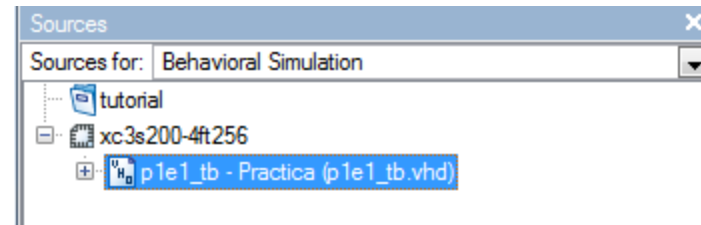
end Practica;

- Guardar el fichero.



## 6. Simulación

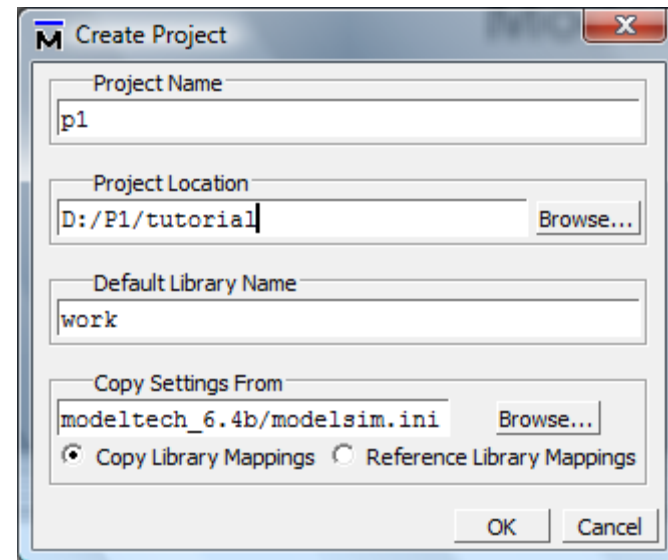
- En el panel *Sources* seleccionamos *Sources for: Behavioral Simulation* y seleccionamos el fichero del *testbench* *P1e1\_tb*.
- En el panel *Processes* hacemos doble clic sobre: *Simulate Behavioral Model*
- Se abrirá el *ModelSim* y permitirá simular nuestro diseño.





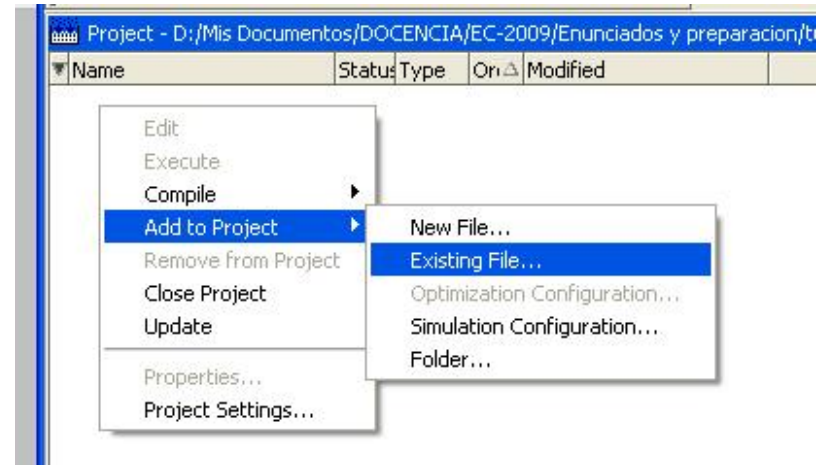
## 6. Simulación





- Abrimos *ModelSim*
- Creamos un nuevo proyecto:  
“*File → new → Project*”
- Definimos en la ventana el nombre del proyecto y su localización.



## 6. Simulación



- Añadimos los ficheros correspondientes.
  - P1e1.vhd
  - P1e1\_tb.vhd





Project - D:/Mis Documentos/DOCENCIA/EC-2009/Enunciados y preparacion/tutorial/demo					
Name	Status	Type	On	Modified	
 p1e1_tb.vhd		VHDL	0	01/22/10 11:10:13 AM	
 p1e1.vhd		VHDL	1	01/25/10 09:33:21 AM	

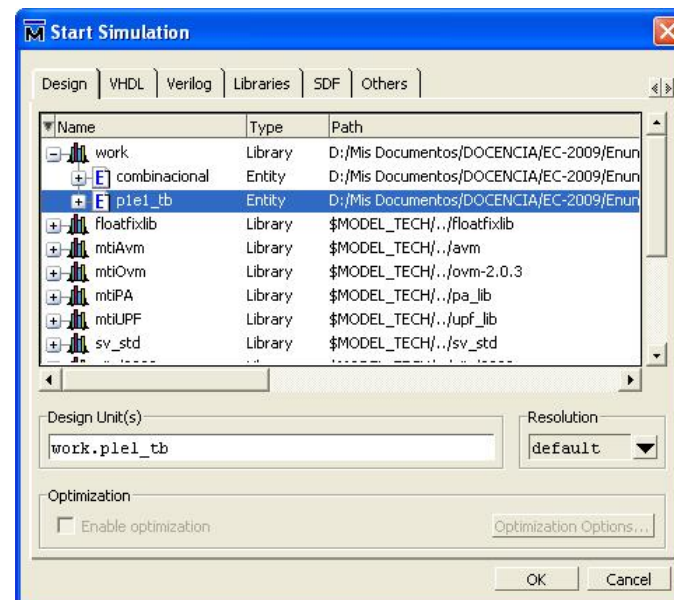


## 6. Simulación

- Compilamos los ficheros que acabamos de adjuntar . Si todo es correcto, deberían aparecer como en la imagen.
- A continuación debemos comenzar la simulación . Seleccionamos en la ventana el *testbench*.

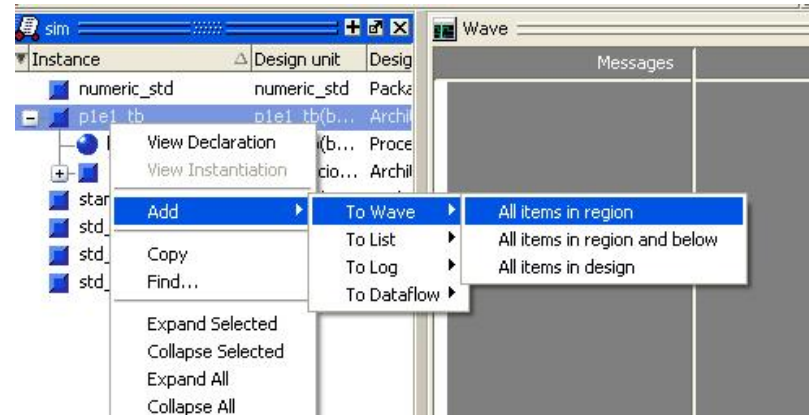
Project - D:/Mis Documentos/DOCENCIA/EC-2009/Enunciados y preparacion/tutorial/demo

Name	Status	Type	On	Modified
p1e1_tb.vhd		VHDL	0	01/22/10 11:10:13 AM
p1e1.vhd		VHDL	1	01/25/10 09:33:21 AM

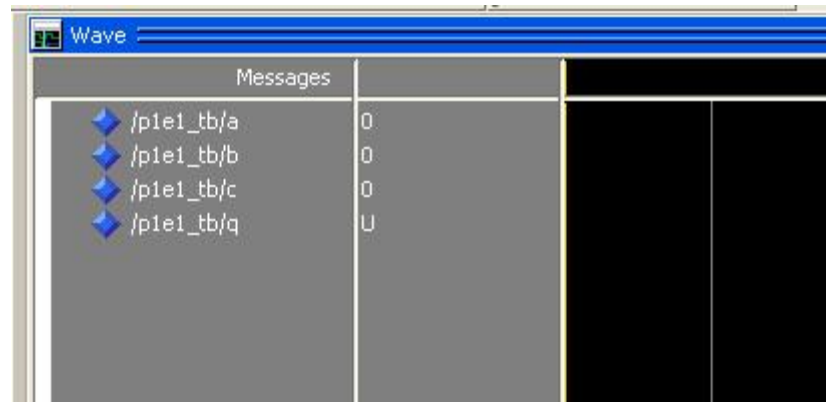


## 6. Simulación

- Seleccionamos el *testbench* e indicamos que añada a la ventana de onda todos los elementos.

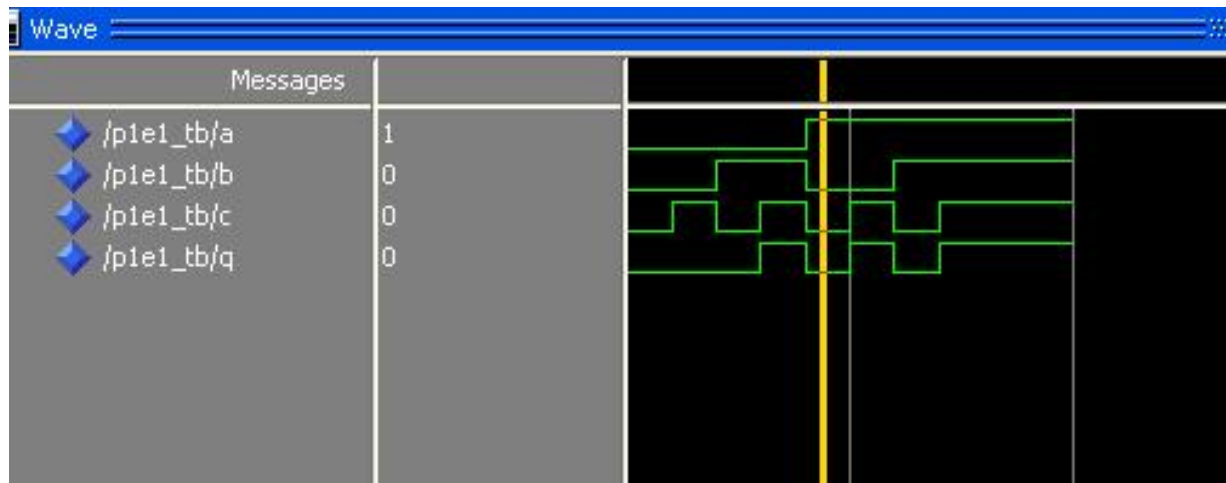
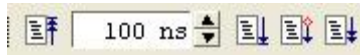


**SEÑALES  
INTERNAS**



## 6. Simulación

- Avanzamos en la simulación.

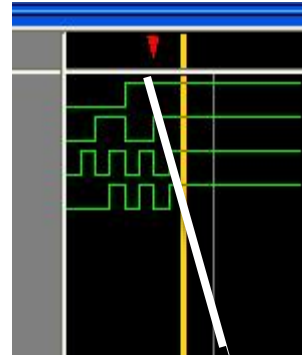


*(Estas ondas corresponden a un fichero de testbench completado)*



## 6. Simulación

- Si se detecta un error...



```
# Loading ieee.std_logic_arith(body),
# Loading ieee.std_logic_unsigned(body)
# Loading ieee.numeric_std(body)
# Loading work.ple1_tb(behavior)
# Loading work.combinacional(practica)
add wave sim:/ple1_tb/*
VSIM 7> run
# ** Failure: Error en el caso '1' '0' '1'
#   Time: 60 ns  Iteration: 0  Process: /ple1_tb,
# Break in Process line_77 at D:/Mis Documentos/I
VSIM 8> run
```

Marca el error en la simulación e imprime el mensaje definido

