



# Práctica 1

- Utilización de la herramienta Xilinx ISE con VHDL
- Diseño y simulación de circuitos sencillos



# Pantalla de bienvenida

Aquí aparecerán los proyectos ya existentes

Did you know...

**Project Management:**  
You can search for missing modules in the design hierarchy by opening the **Find** toolbar in the **Design Hierarchy view** (right-click > **Find...** in the Hierarchy window), then selecting **Missing Module** as the Type, and entering \* in the Find field.

OK  
Next Tip  
Previous Tip  
Help  
 Show Tips at Startup

Console

Console Errors Warnings Find in Files Results

Si nos aparece otra pantalla,  
ir a transparencia 12



# Crear un proyecto (I)

## ■ Desde la ventana de bienvenida

ISE Project Navigator (P.15xf)

New Project Wizard

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project.

Name:

Location: C:\Users

Working Directory: C:\Users

Description:

Select the type of top-level source for the project

Top-level source type: HDL

Next Cancel

Welcome to the ISE® Design Suite

Project commands

- Open Project...
- Project Browser...
- New Project... **Selected**
- Open Example...

Recent projects

Double click on a project in the list below to open it.

Additional resources

Tutorials on the Web  
Design Resources  
Application Notes

Console

Console Errors Warnings Find in Files Results

Siempre en hlocal



# Seleccionar dónde se va a implementar

New Project Wizard

### Project Settings

Specify device and project properties.  
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
<b>Family</b>	Spartan3
Device	XC3S1000
Package	FT256
Speed	-5
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
<b>Simulator</b>	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project	None
Manual Compile Order	None
VHDL Source Analysis Standard	None
Enable Message Filtering	None

**Seleccionar el entorno de simulación ISim**

More Info      Next      Cancel

Spartan 3  
XC3S1000  
FT256  
-5



# Crear un archivo VHDL (I)

- Siempre se inicia creando un fichero tipo VHDL Module

The screenshot shows the Xilinx ISE interface. On the left, the 'Design' view is selected, displaying a project structure for 'practica1' containing an 'xc3s1000-5ft256' device. A callout bubble labeled 'New source' points to the 'Design' tab in the toolbar.

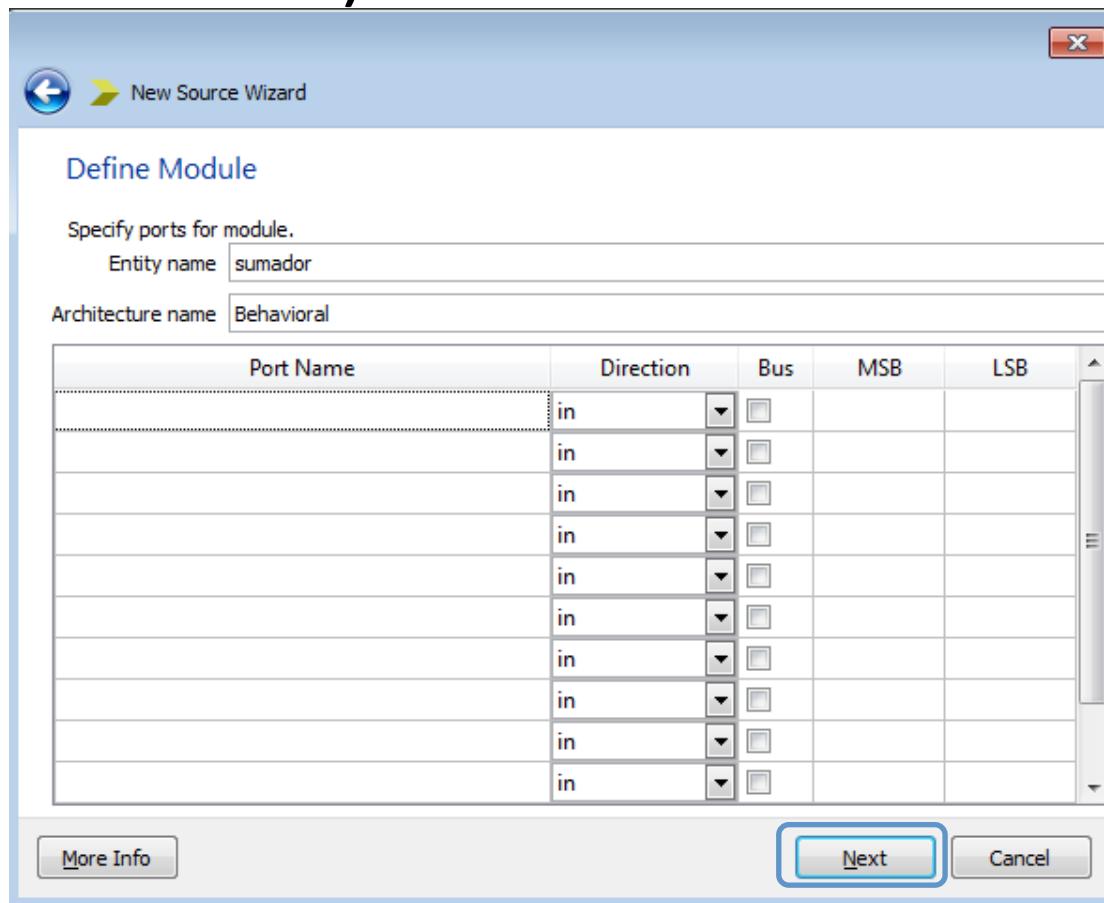
The main window displays the 'New Source Wizard' with the title 'Select Source Type'. The wizard instructions state: 'Select source type, file name and its location.' A list of source types is shown, with 'VHDL Module' highlighted. A callout bubble labeled 'Nombre del fichero VHDL' points to the 'File name:' field, which contains 'C:\Xilinx\practica1'.

A large callout bubble at the bottom right contains the text: 'Si traéis el fichero VHDL desde casa hay que pulsar en Add Source(pasar a transparencia 9)'.



# Crear un fichero VHDL (II)

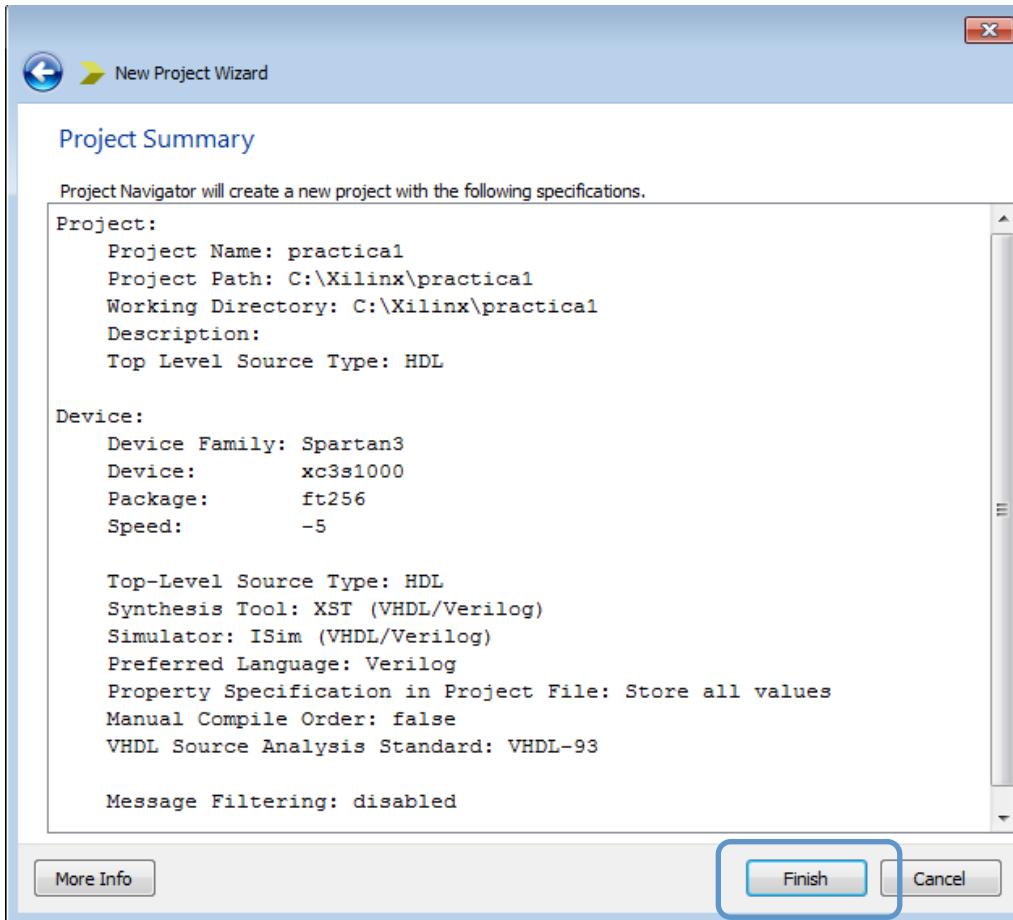
- Wizard para definir las señales de entrada y salida (no utilizar)





# Crear un fichero VHDL (III)

- Todo ha salido correcto





# Crear un fichero VHDL (IV)

## ■ Todo ha salido correcto

ISE Project Navigator (P.15xf) - C:\Xilinx\ejemplo\ejemplo.xise - [sumador.vhd]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ejemplo
  - xc3s1000-5ft256
    - sumador - Behavioral (sumador.vhd)

El fichero aparece en el árbol

```
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx primitives in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity sumador is
35     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
36             B : in STD_LOGIC_VECTOR (3 downto 0);
37             C : out STD_LOGIC_VECTOR (3 downto 0));
38 end sumador;
39
40 architecture Behavioral of sumador is
```

No Processes Running

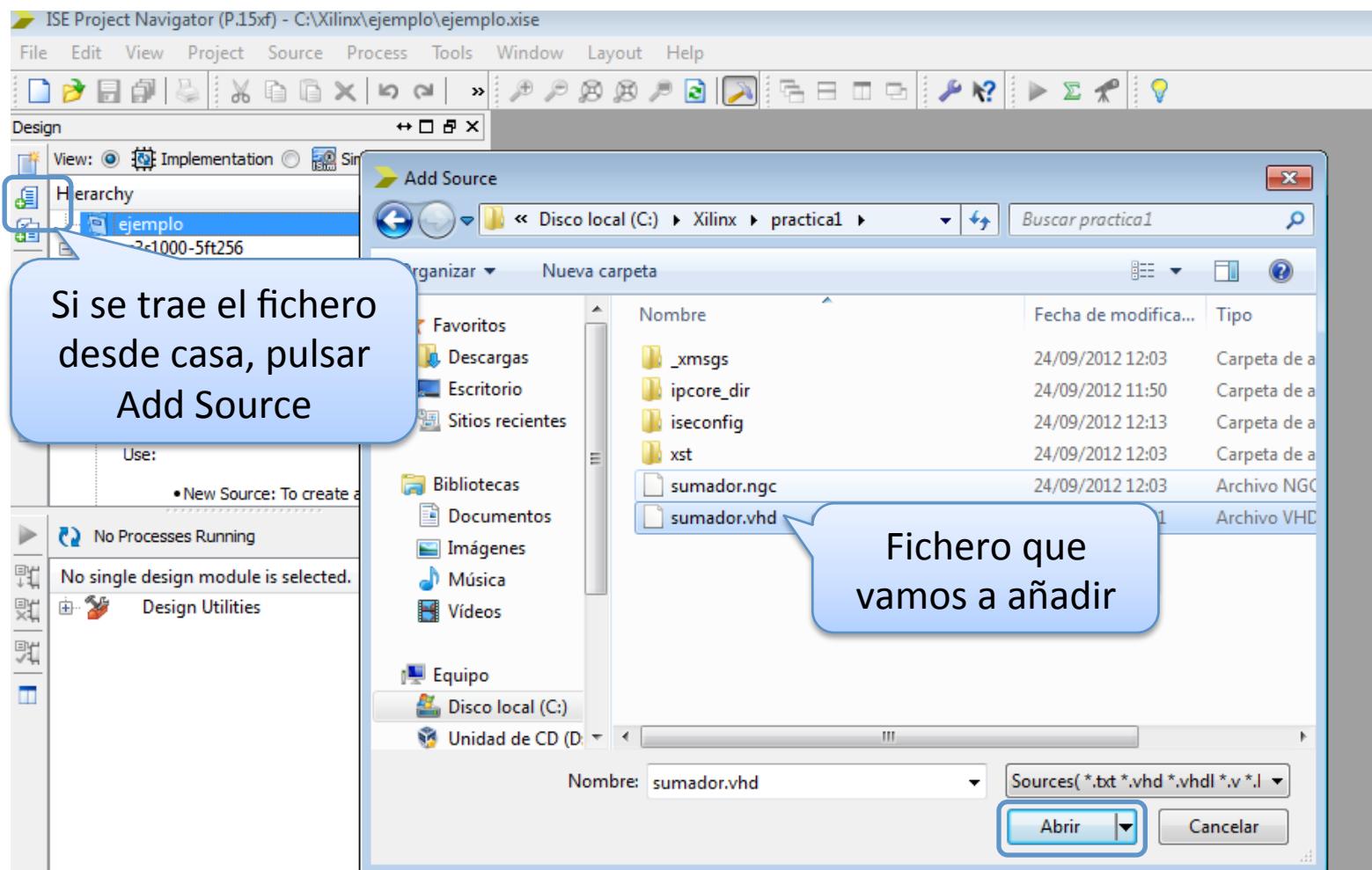
Processes: sumador - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope



# Añadir ficheros al proyecto (I)

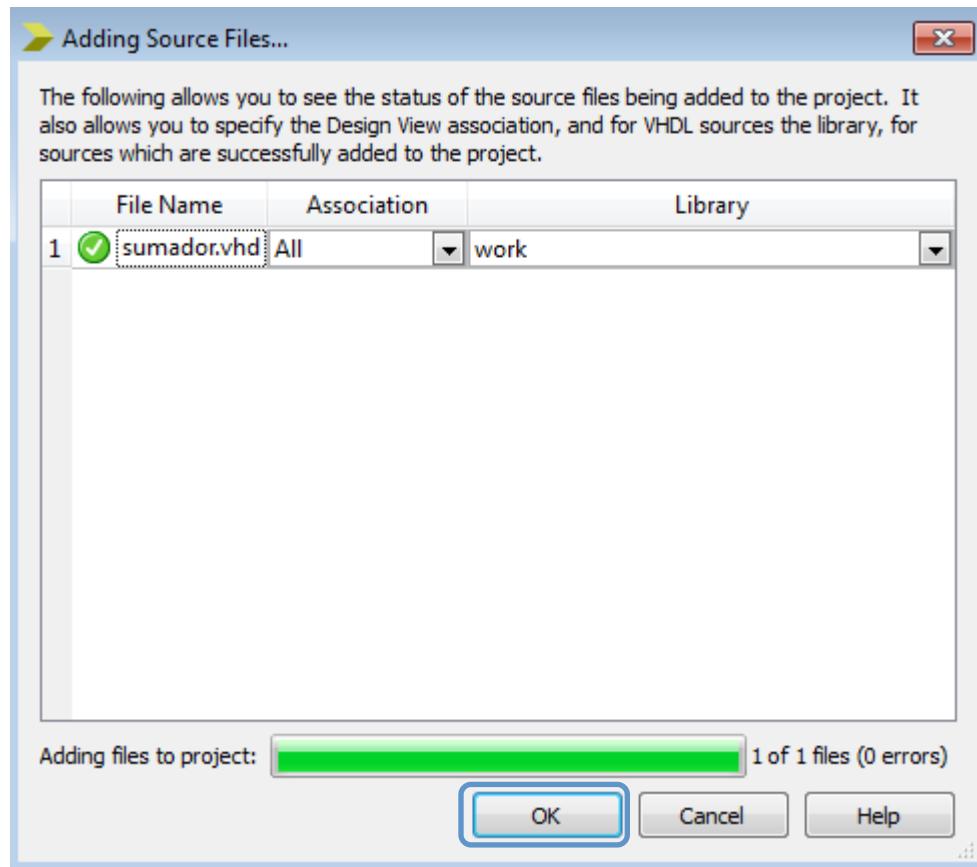
- Los ficheros a añadir deben ser previamente grabados en el directorio hlocal





# Añadir ficheros al proyecto (II)

- Aparece la siguiente ventana que indica que el fichero se está añadiendo al proyecto





# Añadir ficheros al proyecto (III)

- El fichero se ha añadido correctamente

ISE Project Navigator (P.15xf) - C:\Xilinx\ejemplo\ejemplo.xise - [sumador.vhd]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ejemplo
  - xc3s1000-5ft256
    - sumador - Behavioral (sumador.vhd)

El fichero se ha incorporado al proyecto

```
11 -- Description:  
12 --  
13 -- Dependencies:  
14 --  
15 -- Revision:  
16 -- Revision 0.01 - File Created  
17 -- Additional Comments:  
18 --  
19  
20 library IEEE;  
21 use IEEE.STD_LOGIC_1164.ALL;  
22 use IEEE.STD_LOGIC_ARITH.ALL;  
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;  
24  
25 -- Uncomment the following library declaration if using  
26 -- arithmetic functions with Signed or Unsigned values  
27 --use IEEE.NUMERIC_STD.ALL;  
28  
29 -- Uncomment the following library declaration if instantiating  
30 -- any Xilinx primitives in this code.  
31 --library UNISIM;  
32 --use UNISIM.VComponents.all;  
33  
34 entity sumador is  
35     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);  
36                 B : in STD_LOGIC_VECTOR (3 downto 0);  
37                 C : out STD_LOGIC_VECTOR (3 downto 0));  
38 end sumador;  
39  
40 architecture Behavioral of sumador is
```

No Processes Running

Processes: sumador - Behavioral

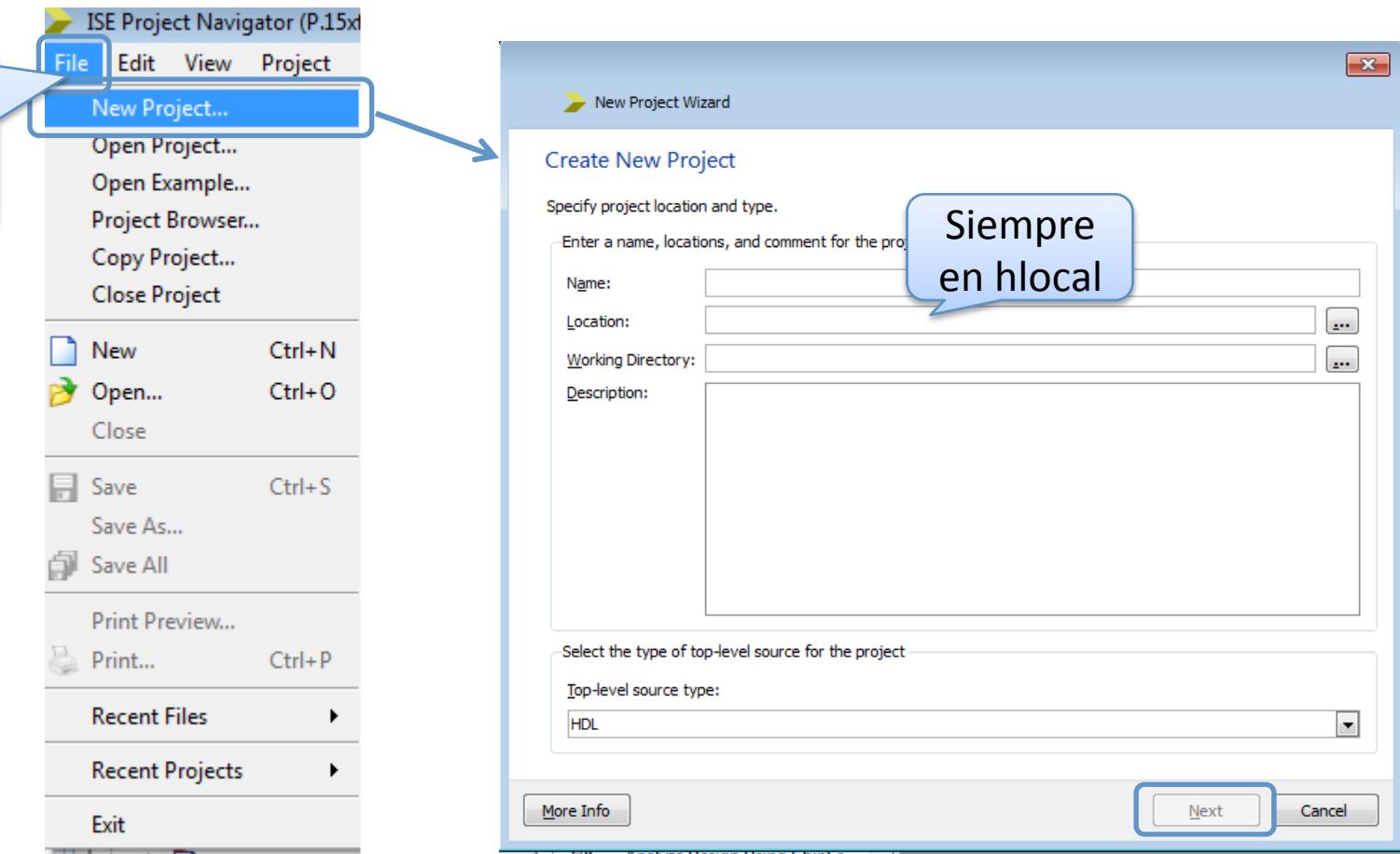
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope



# Crear un proyecto (II)

- Una vez usado el entorno, el ISE Project Navigator se suele abrir con el último proyecto con el que hemos trabajado

Para crear un nuevo proyecto





# Anotaciones



# Pantalla de desarrollo

ISE Project Navigator (P.15xf) - C:\Xilinx\practical\practical1.xise - [sumador.vhd\*]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- practical
- xc3s1000-5ft256
- sumador - Behavioral (sumador)

No Processes Running

Processes: sumador - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

sumador.vhd\*

```
12 --  
13 -- Dependencies:  
14 --  
15 -- Revision:  
16 -- Revision 0.01 - File Created  
17 -- Additional Comments:  
18 --  
19 --  
20 library IEEE;  
21 use IEEE.STD_LOGIC_1164.ALL;  
22  
23 -- Uncomment the following library declaration if using  
24 -- arithmetic functions with Signed or Unsigned values  
25 --use IEEE.NUMERIC_STD.ALL;  
26  
27 -- Uncomment the following library declaration if instantiating  
28 -- any Xilinx primitives in this code.  
29 --library UNISIM;  
30 --use UNISIM.VComponents.all;  
31  
32 entity sumador is  
33 end sumador;  
34  
35 architecture Behavioral of sumador is  
36 begin  
37  
38 end Behavioral;
```

Aquí se escribe el código VHDL que nos interese

Console

```
Started : "Launching ISE Text Editor to edit sumador.vhd".  
Launching Design Summary/Report Viewer...
```



# Práctica 1.a

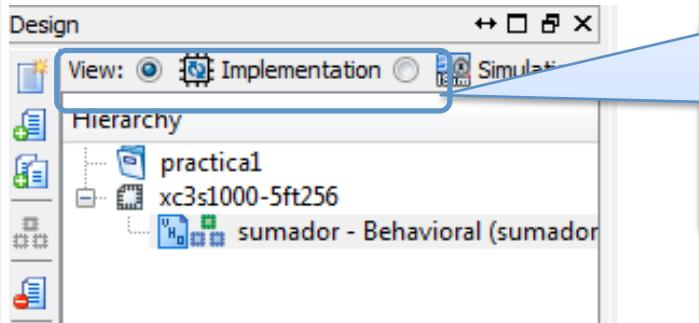
```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

Diseñar en VHDL un circuito  
sumador de números de 4 bits

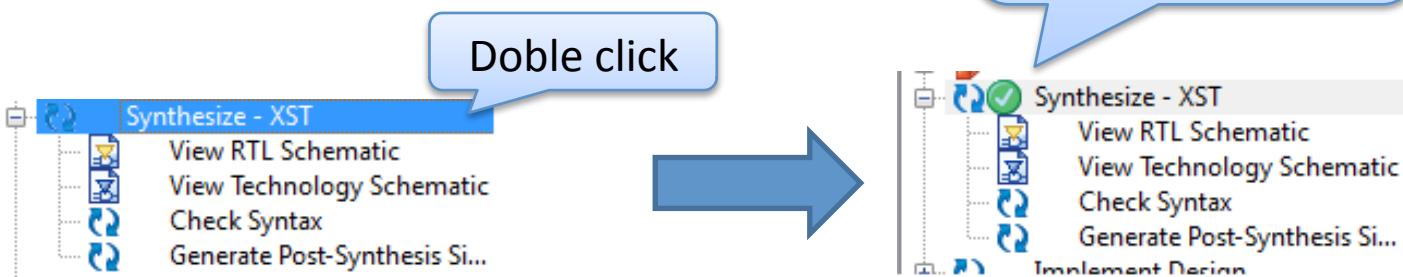
```
entity sumador is  
port (A, B: in std_logic_vector(3 downto 0);  
      C: out std_logic_vector(3 downto 0));  
end sumador;  
  
architecture beh of sumador is  
-- No hace falta definir señales intermedias  
begin  
  
C <= A + B;  
  
end beh;
```



# Sintetizar el diseño



Para poder implementar el circuito tenemos que estar en la opción Implementation



Si sale este símbolo verde es que todo ha salido perfecto



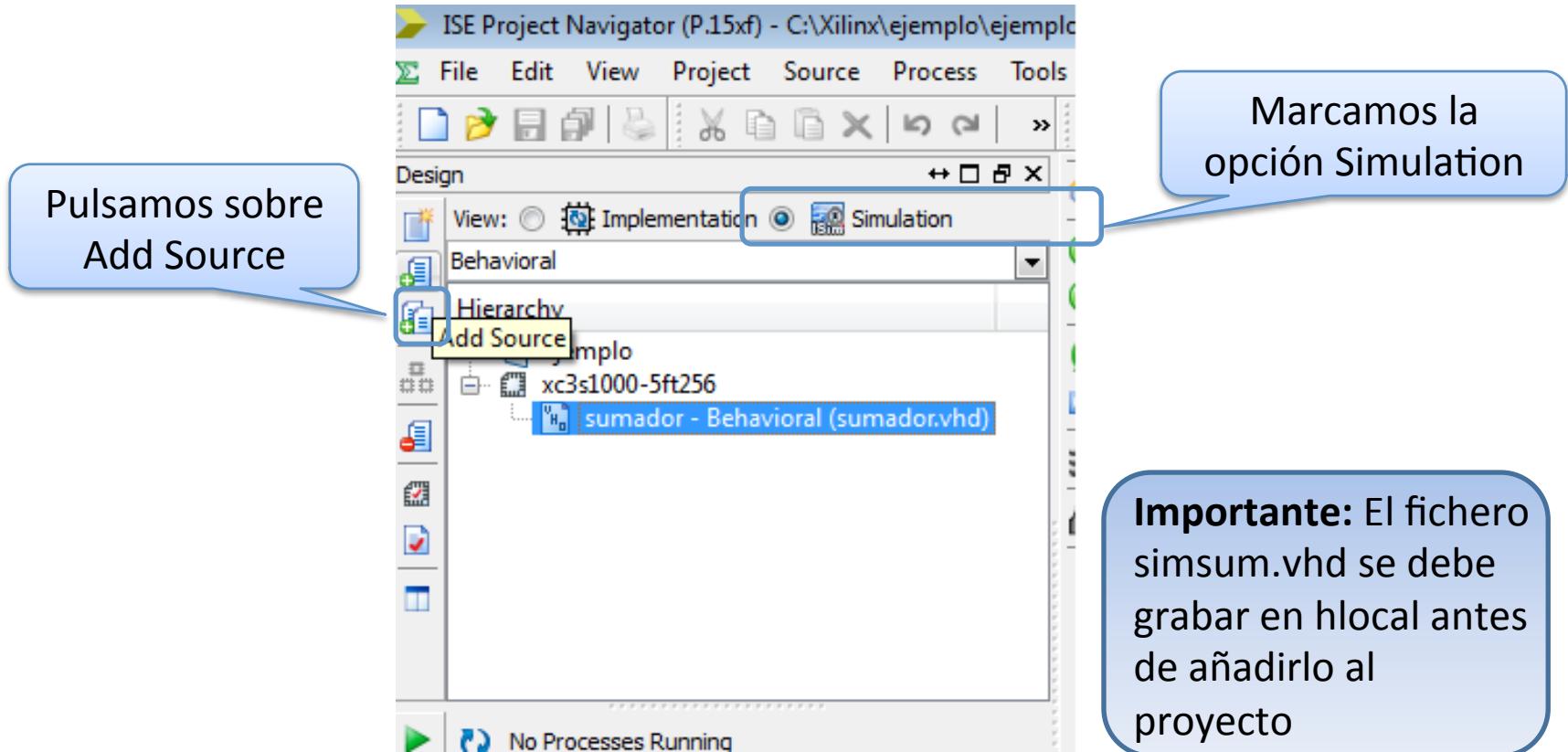
# Comprobar que el Diseño es Correcto

- Para realizar esta comprobación hace falta simular su comportamiento:
  - En **casa y en el laboratorio** podéis crear automáticamente un test de simulación para comprobar que todo funciona correctamente. En esta primera práctica se os dará el fichero de simulación ya creado (ver transparencias Test de simulación)
  - Para **calificar la práctica** tendréis que pasar un archivo de simulación que se os dará en el laboratorio (ver transparencias de Calificación de la Práctica)



# Test de simulación (I)

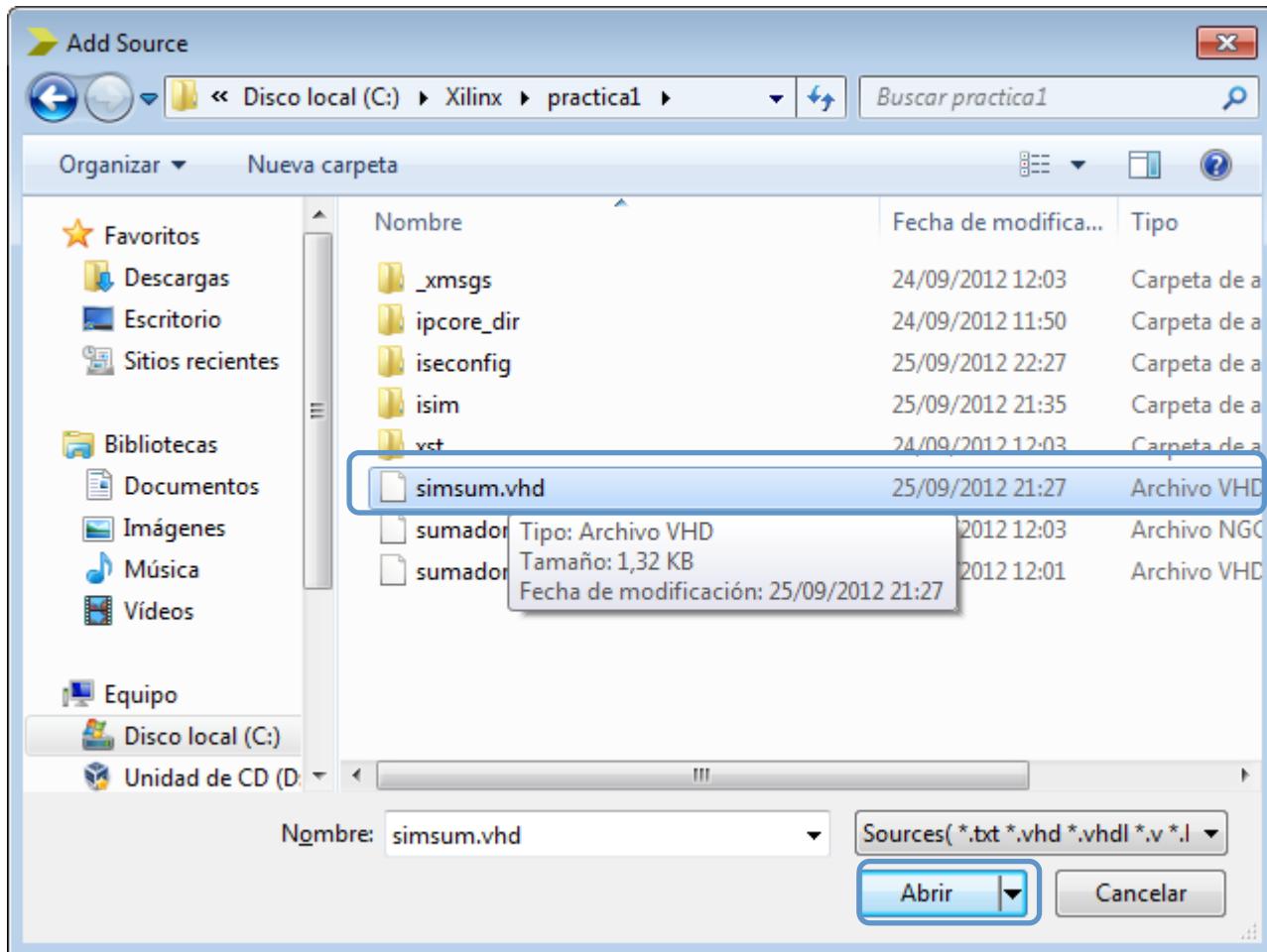
- Añadir el fichero de simulación simsum.vhd





# Test de Simulación (II)

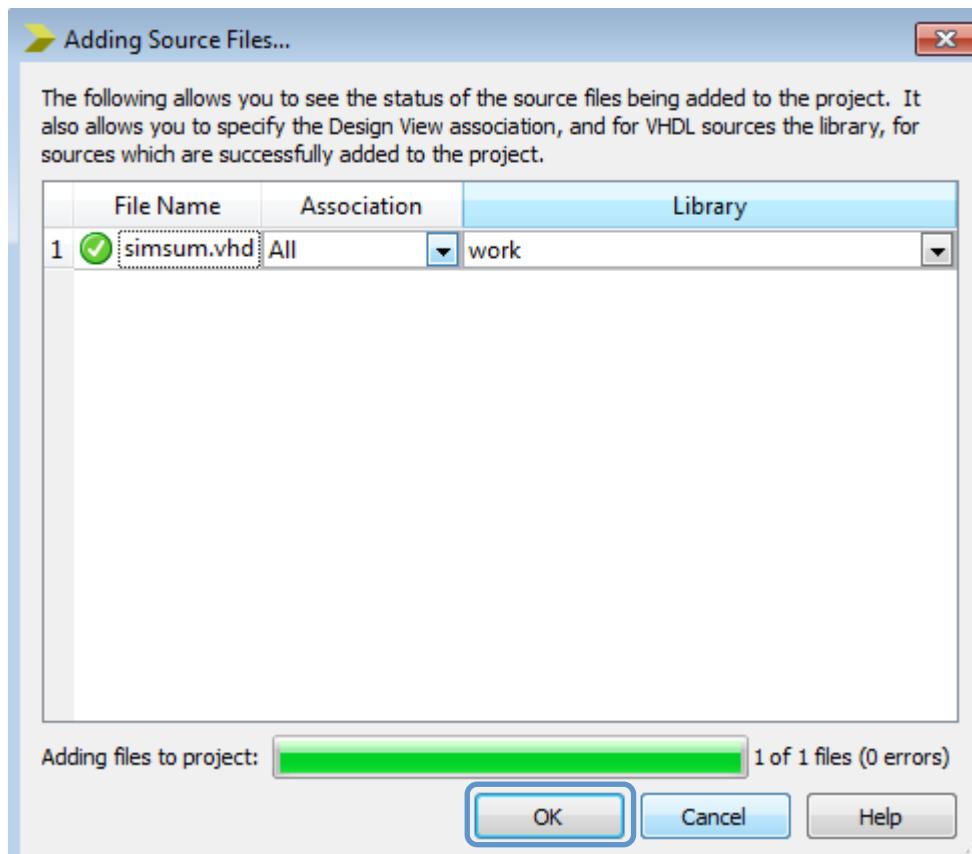
- Seleccionamos el fichero simsum.vhd





# Test de simulación (III)

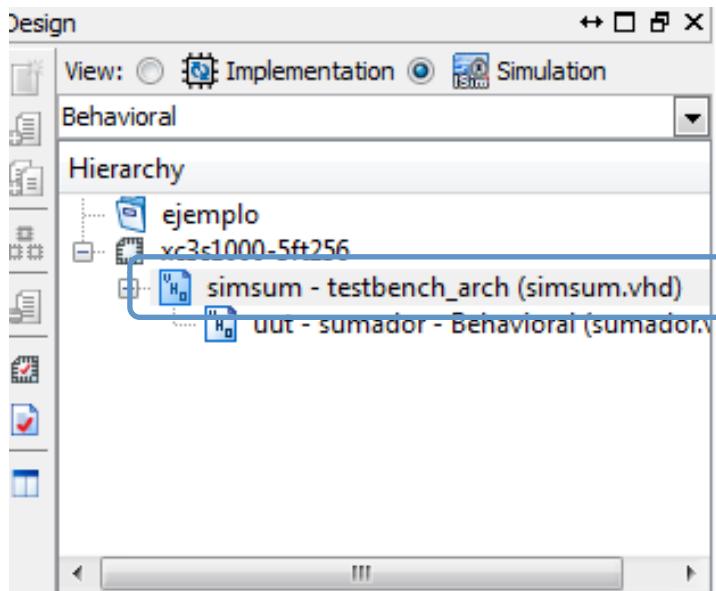
- El fichero se está añadiendo al proyecto





# Test de simulación (IV)

- El fichero se ha añadido correctamente



Aparece en el árbol de ficheros



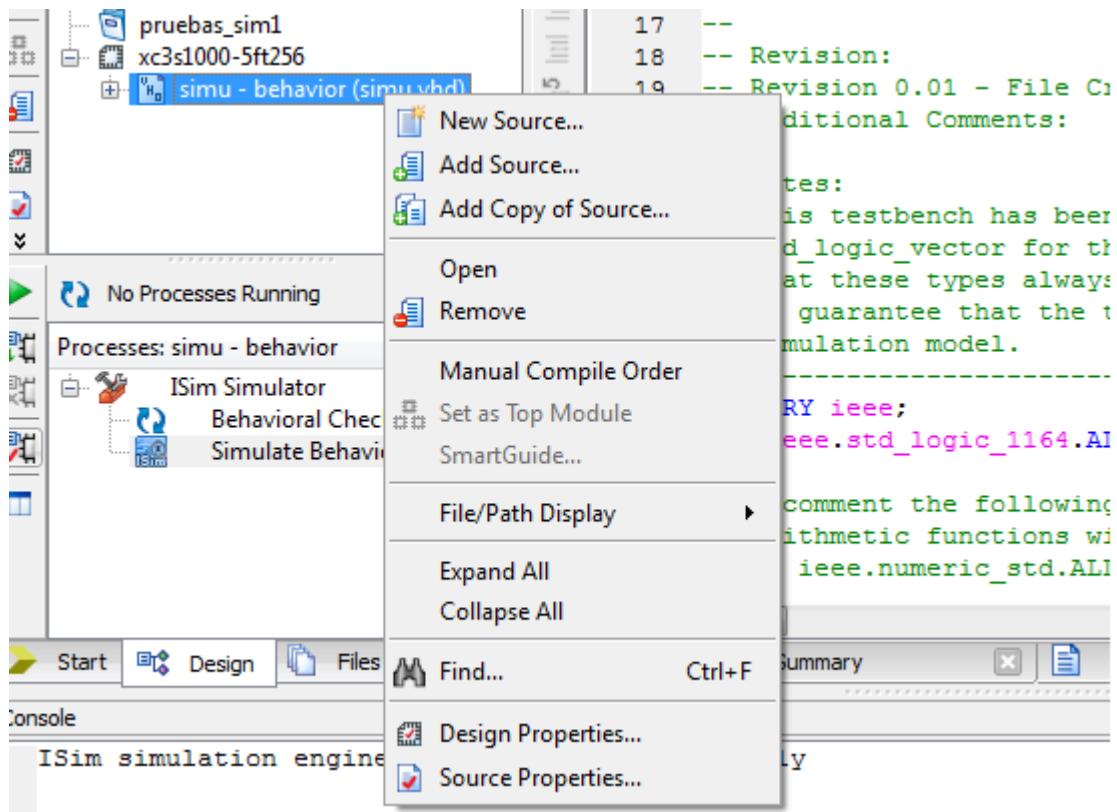
# Fichero de simulación (I)

- Si el fichero de simulación se añade al proyecto no lo reconoce como fichero de simulación
- Asegurarse antes de hacer la implementación que se trata de un fichero de simulación



# Fichero de simulación (II)

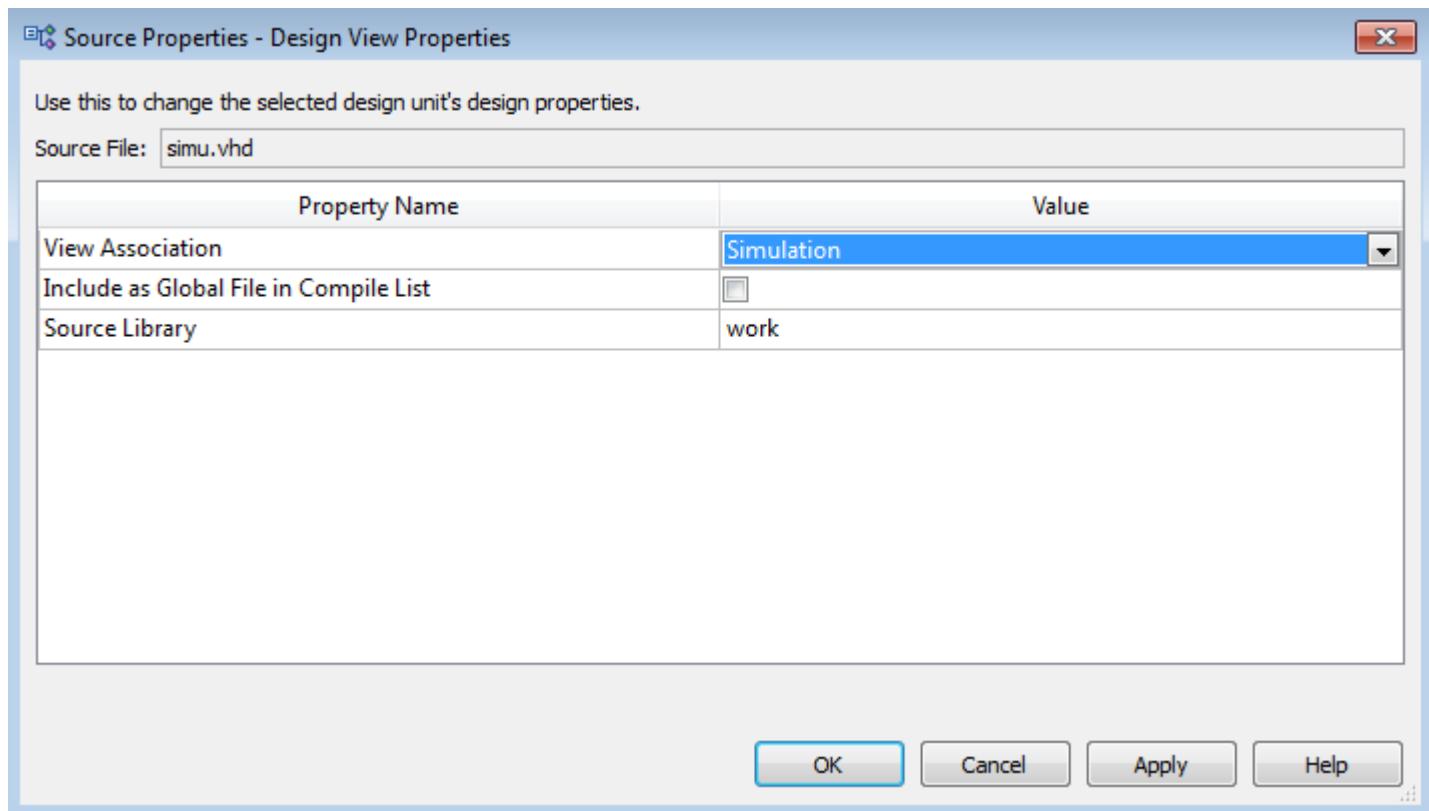
- Pulsar con el botón derecho sobre dicho fichero y elegir la opción *Source properties*





# Fichero de simulación (III)

- En *View Association* aparecerá de tipo *All*, elegir tipo *Simulation*



# Test de simulación (V)



```
-- Añadimos las librerías necesarias
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;

--entidad
entity simsum is
end simsum;

--arquitectura
architecture testbench_arch of simsum is
-- Declaración del componente que vamos a simular
component sumador
    port( A : IN  std_logic_vector(3 downto 0);
          B : IN  std_logic_vector(3 downto 0);
          C : OUT  std_logic_vector(3 downto 0)
        );
end component;
--Entradas
signal op1      : std_logic_vector(3 downto 0) := (others => '0');
signal op2      : std_logic_vector(3 downto 0) := (others => '0');
--Salidas
signal result  : std_logic_vector(3 downto 0);
```



# Test de simulación (VI)

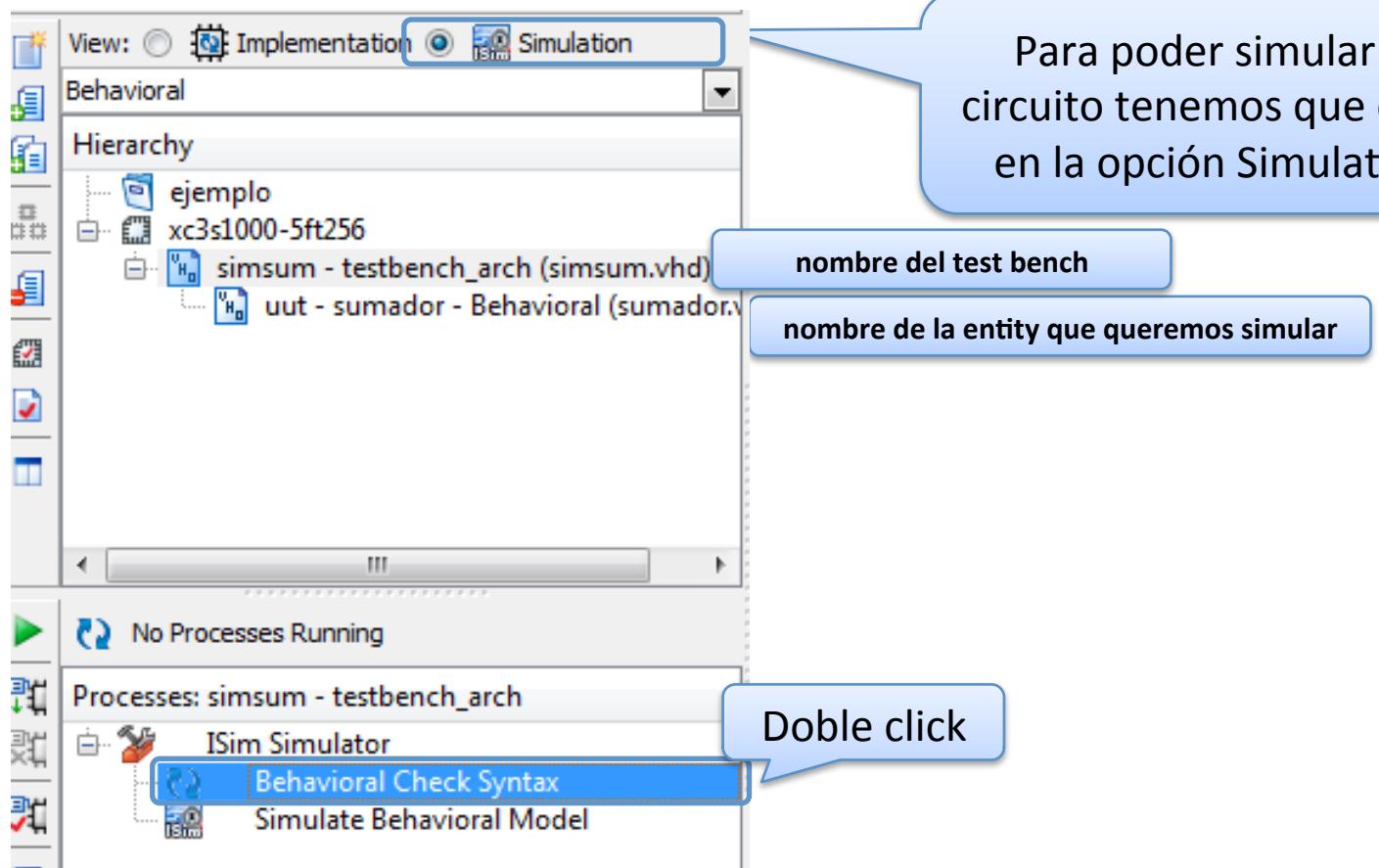
```
Begin
-- Instanciacion de la unidad a simular
uut: sumador port map (
    A => op1    ,
    B => op2    ,
    C => result
);
-- Proceso de estímulos
stim_proc: process
begin
    op1<="0000";
    op2<="0000";
    wait for 100 ns;
    op1<="0101";
    op2<="0100";
    wait for 100 ns;
    A<="0000";
    B<="0111";
    wait for 100 ns;
    A<="0011";
    B<="1000";
    wait for 100 ns;
    A<="1011";
    B<="1111";
    wait for 100 ns;
    A<="1001";
    B<="0110";
    wait;
end process;

end testbench_arch;
```



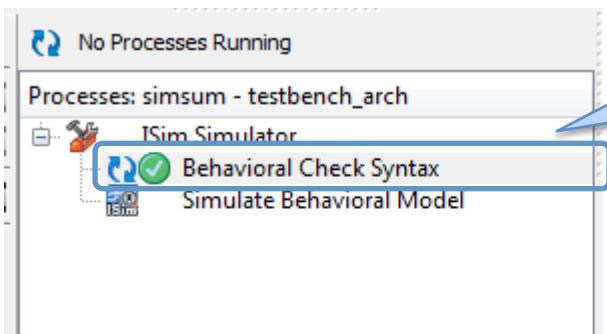
# Simulación (I)

## ■ Desplegar el menú de simulación

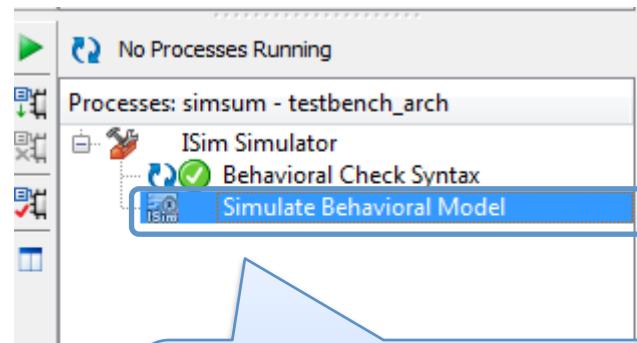




# Simulación (II)



La simulación se ha ejecutado correctamente

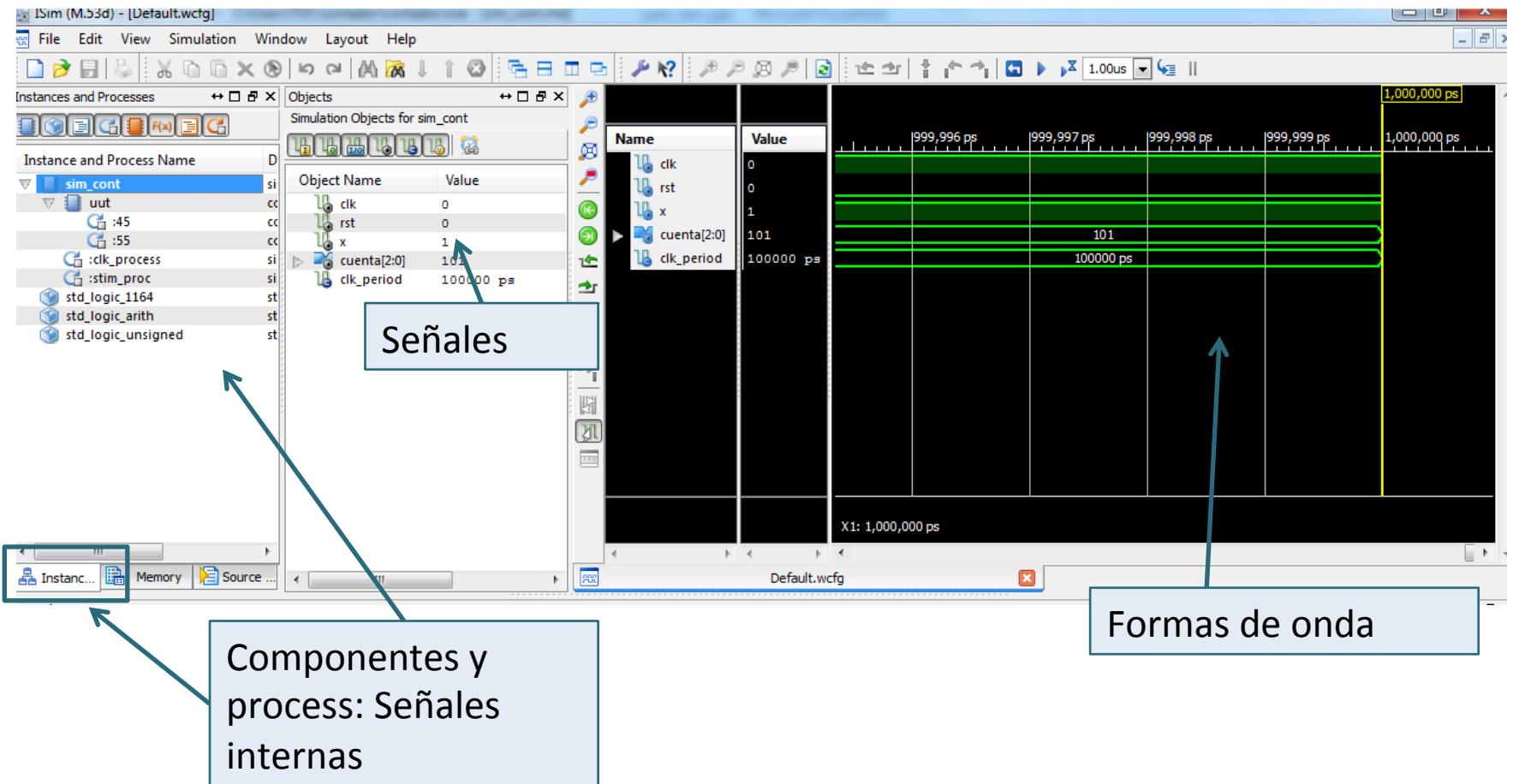


Necesitamos arrancar el ISIM para ver las formas de onda y comprobar que el circuito está bien diseñado



# Simulación (III)

## ■ Ventana simulador ISim



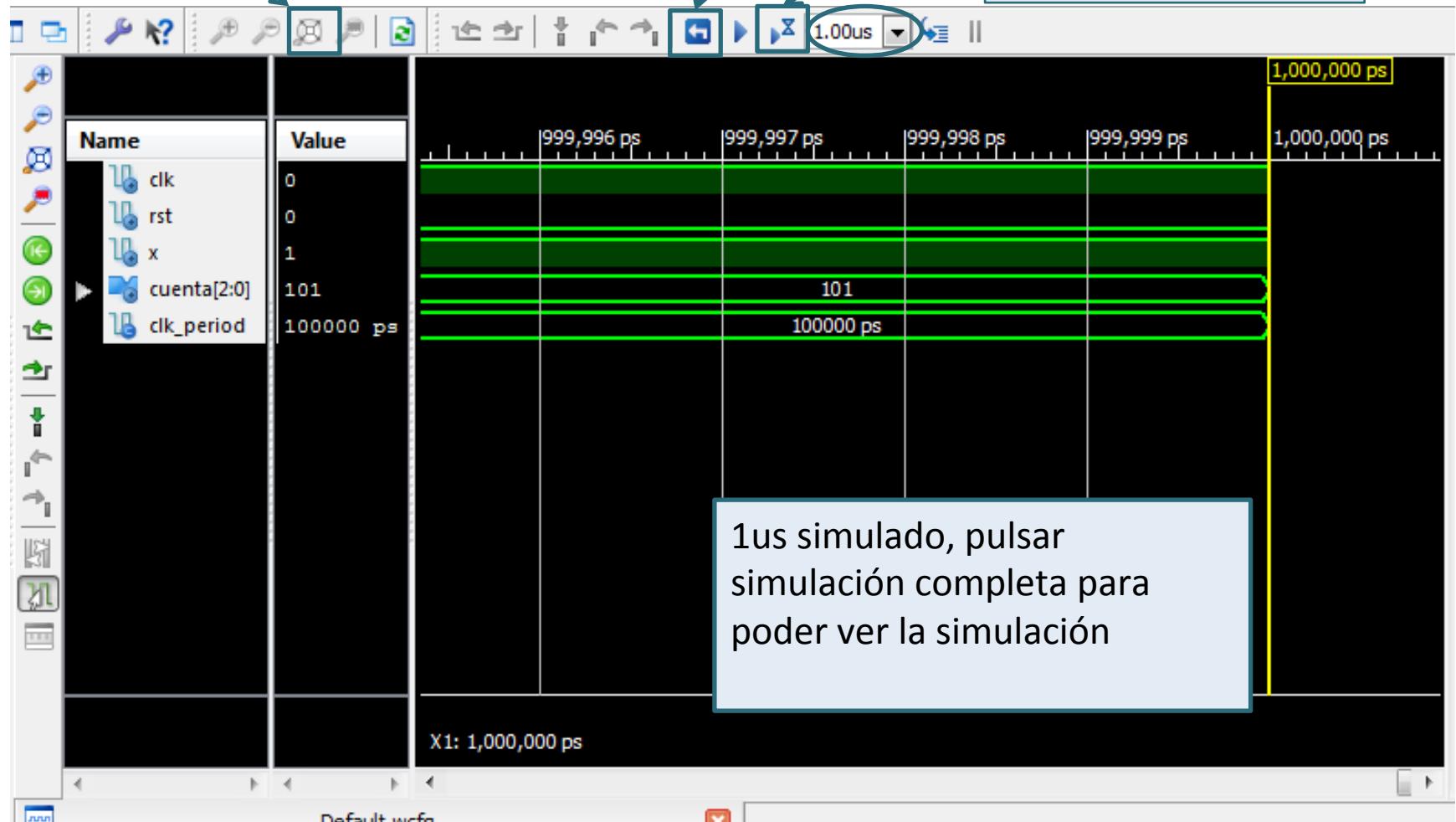


# Simulación (IV)

Ver simulación completa

Reiniciar simulación

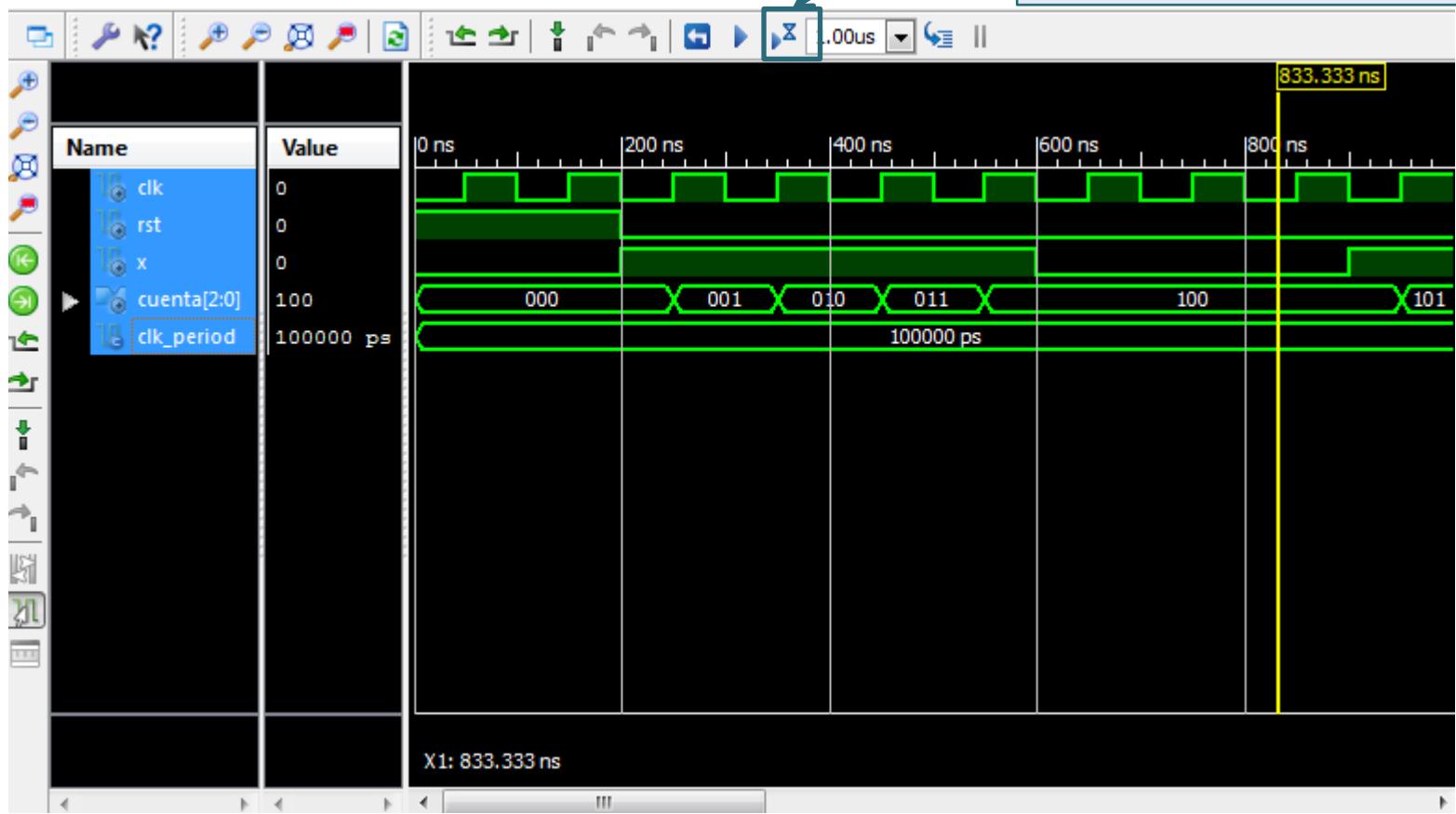
Cada vez que se pulsa este icono se simula 1us





# Simulación (V)

Si queremos simular más tiempo pulsar en este icono





# Simulación (VI)

- Si la simulación no es correcta hay que depurar
  - Al abrir el simulador solo se muestra el valor de los puertos de entrada/salida
    - Es necesario para simular añadir las señales internas y los components

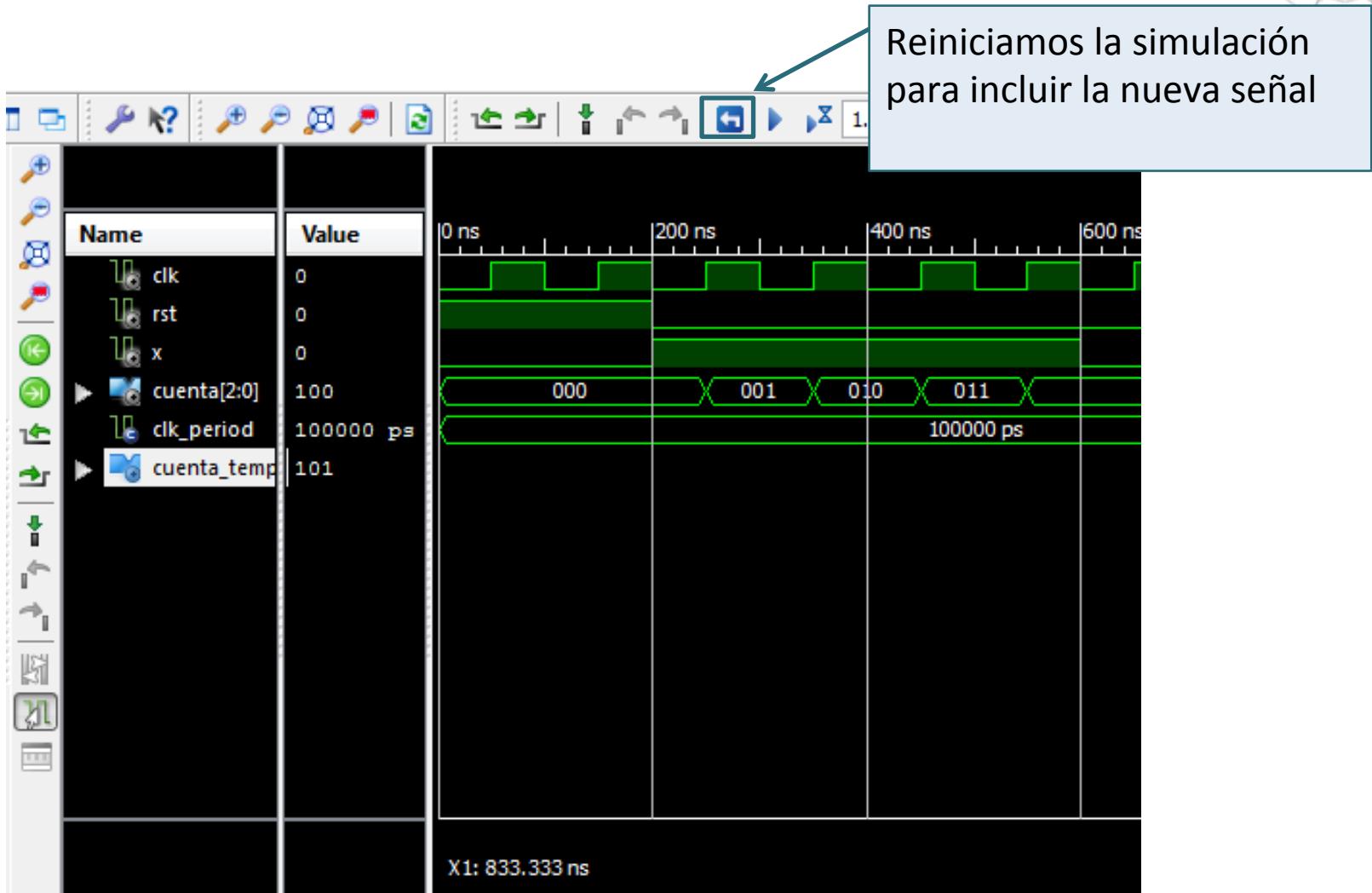
The screenshot shows the ModelSim simulation environment. The Instances and Processes window lists various components and processes. The Objects window displays simulation objects with their names and current values. A context menu is open over the 'cuenta\_temp[2:0]' object, with the 'Add To Wave Window' option selected. The Wave window shows waveforms for the clk, rst, x, cuenta[2:0], and cuenta\_temp[2:0] signals over time.

Buscar las señales internas, para añadirlas pulsar el botón derecho y elegir la opción marcada

This is a Full version of ISim.  
Time resolution is 1 ns.

# Simulación (VII)

- La señal se añadirá a la ventana de simulación





# Anotaciones



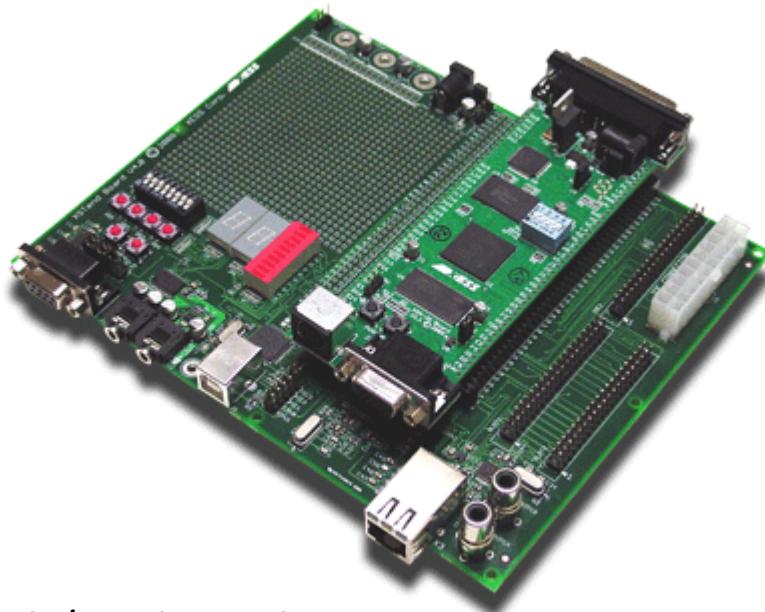
# Implementación (I)

- Podemos comprobar que el circuito funciona perfectamente en el *mundo real*
  - El circuito se va a implementar sobre una FPGA
  - Hay que indicar de dónde se leen las entradas (switches) y en dónde se escriben las salidas (LEDs o display de 7 segmentos)
    - Hay que añadir un fichero UCF donde se le indica a la herramienta lo anterior



# Implementación (II)

- Esta es la plataforma sobre la que se van a implementar todos los diseños de este curso:



<http://www.xess.com/prods/prod039.php>



# Implementación (III)

## ■ Fichero UCF

- En el Campus Virtual encontraréis el fichero UCF completo
- Aquí aparecen los pines particulares que se necesitan para la práctica 1.a

```
#switches placa extendida
```

```
NET A<0> LOC=P12;
```

```
NET A<1> LOC=J1;
```

```
NET A<2> LOC=H1;
```

```
NET A<3> LOC=H3;
```

```
NET B<0> LOC=G2;
```

```
NET B<1> LOC=K15;
```

```
NET B<2> LOC=K16;
```

```
NET B<3> LOC=F15;
```

```
#barra de leds placa extendida
```

```
NET C<0> LOC=L5;
```

```
NET C<1> LOC=N2;
```

```
NET C<2> LOC=M3;
```

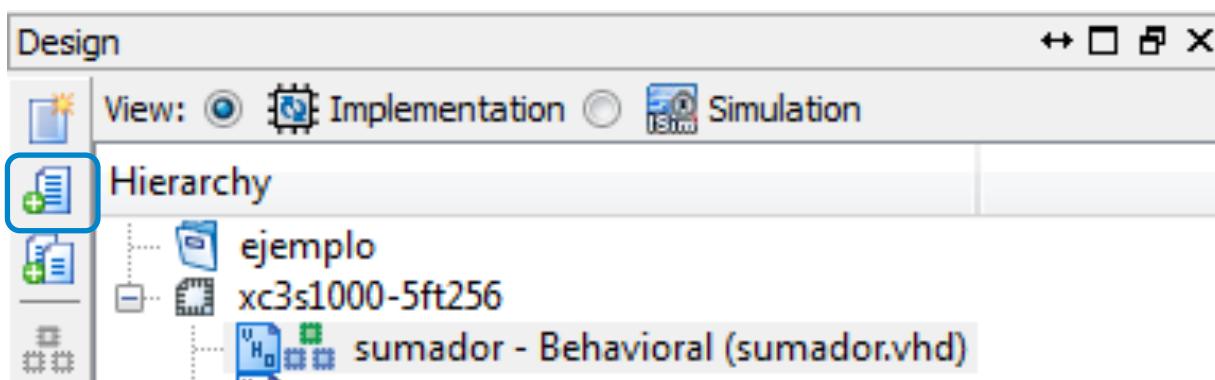
```
NET C<3> LOC=N1;
```

- En el fichero UCF deben aparecer todos los puertos de la entity
- A cada puerto de la entity se le asigna un pin de la FPGA
- Para que lo anterior sea efectivo hay que descomentar la línea correspondiente (quitar #)
- Los pines de la FPGA se instancian LOC = letra+número



# Implementación (IV)

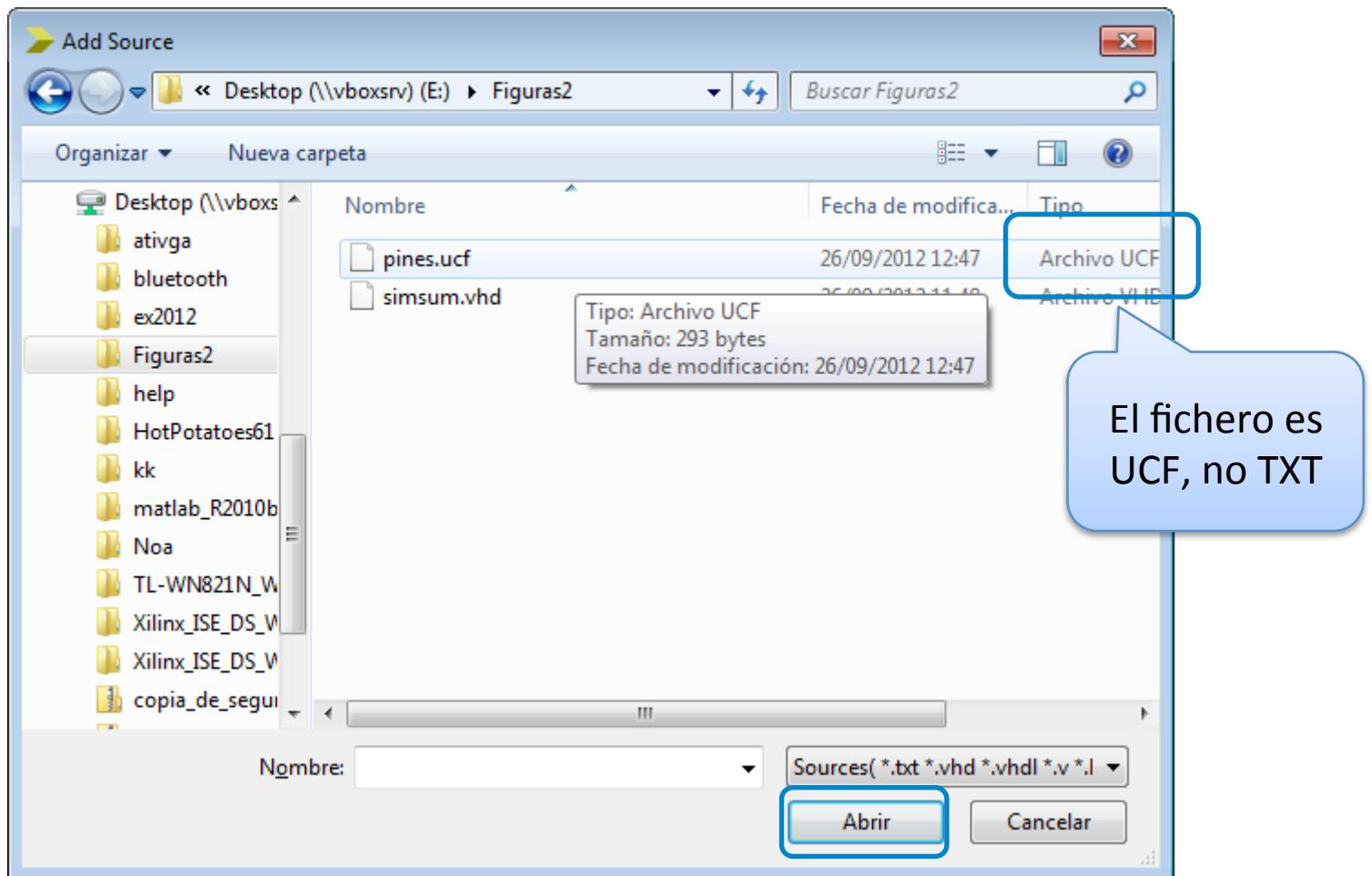
- El fichero pines.ucf se añade al proyecto mediante Add Source





# Implementación (V)

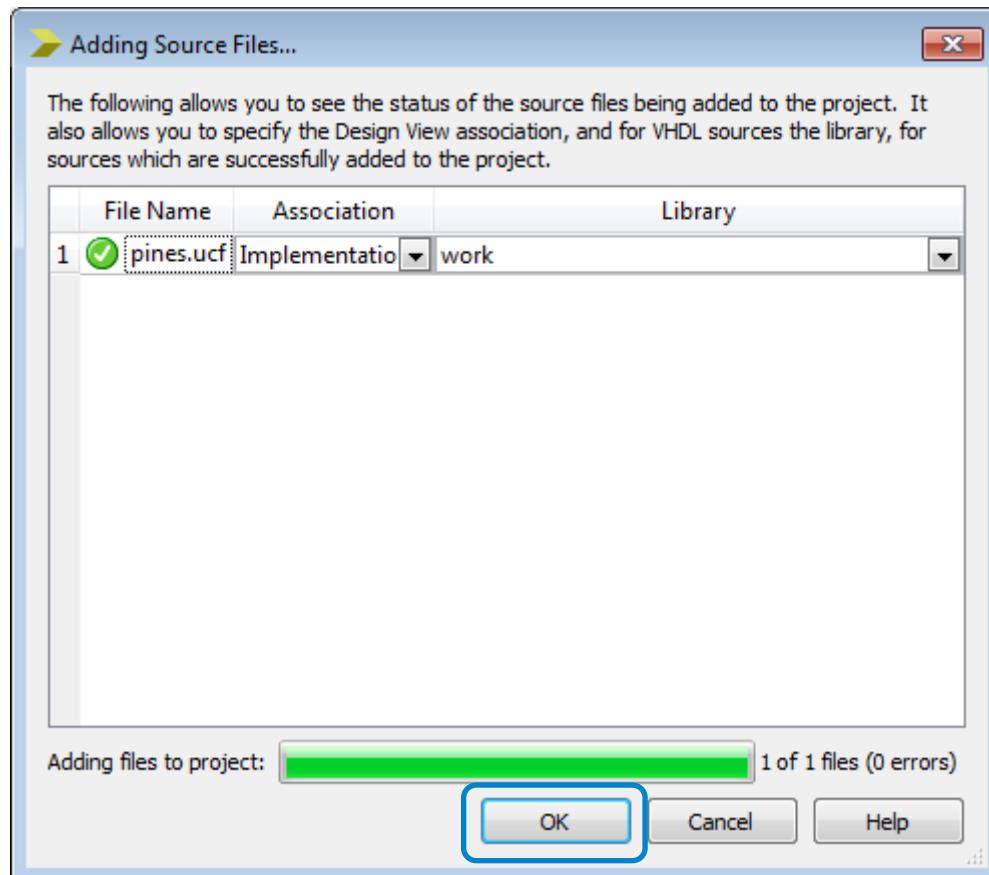
- Seleccionamos el fichero pines.ucf





# Implementación (VI)

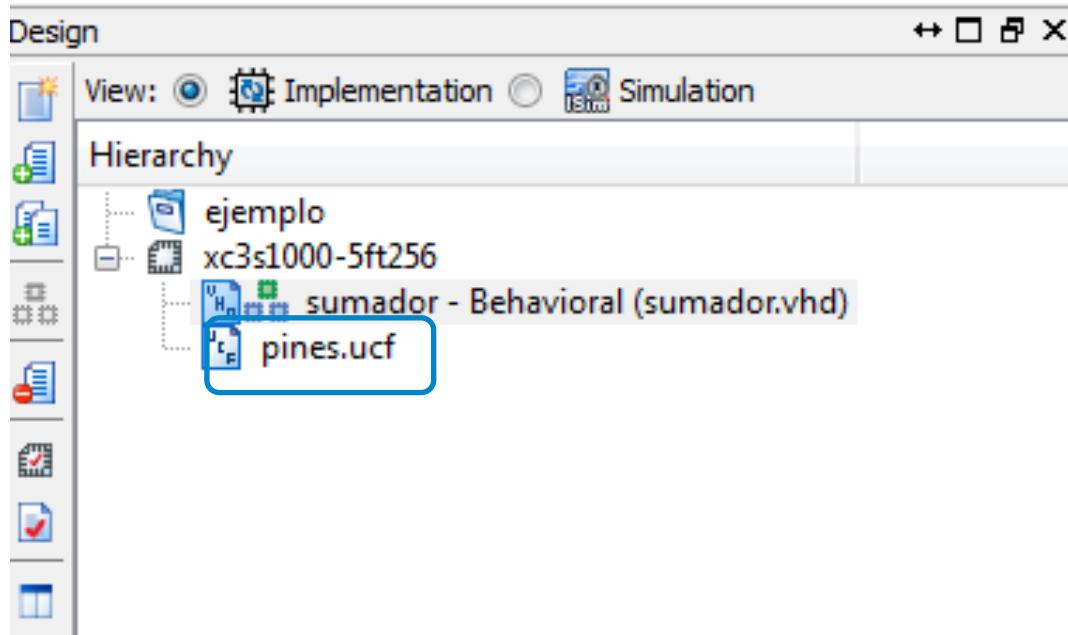
- El fichero se añade al proyecto





# Implementación (VI)

- El fichero aparece en el árbol de jerarquía





# Implementación (VII)

The screenshot shows the Xilinx Vivado interface with the "Implement Design" menu highlighted. A blue callout bubble points to the menu with the text "Doble click". The menu items are:

- Translate
  - Generate Post-Translate Simulation Model
- Map
  - Generate Post-Map Static Timing
  - Manually Place & Route (FPGA Editor)
  - Generate Post-Map Simulation Model
- Place & Route
  - Generate Post-Place & Route Static Timing
  - Analyze Timing / Floorplan Design (PlanAhead)
  - View/Edit Routed Design (FPGA Editor)
  - Analyze Power Distribution (XPower Analyzer)
  - Generate Text Power Report
  - Generate Post-Place & Route Simulation ...
- Generate IBIS Model
- Back-annotate Pin Locations

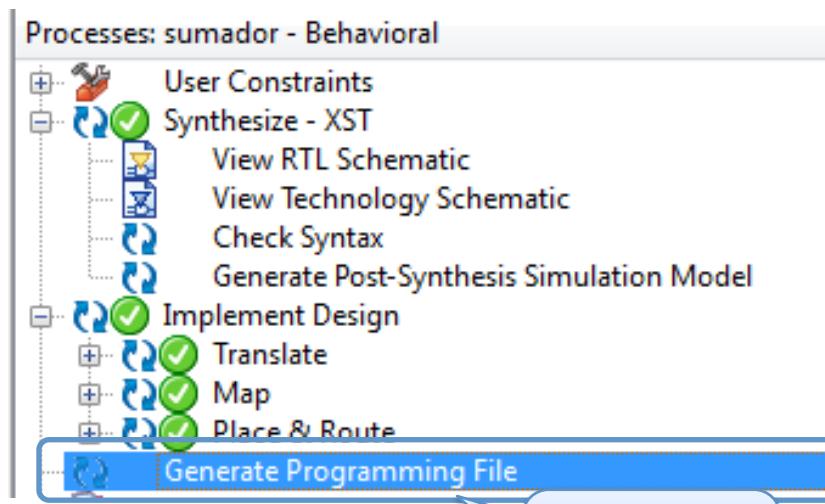
Si todo está correcto  
aparecerá en:

- Implementation Design
- Map
- Place & Route

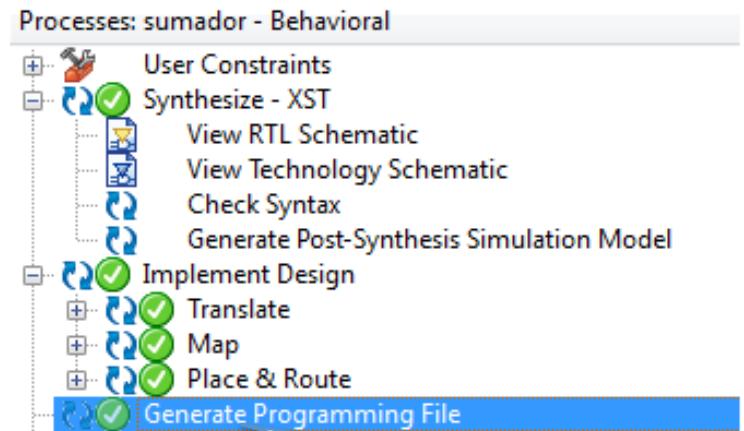


# Implementación (VIII)

- Se genera un fichero de 0s y 1s que describe el circuito, se podrá encontrar en la carpeta del proyecto (nombre\_proyecto.bit)



Doble click

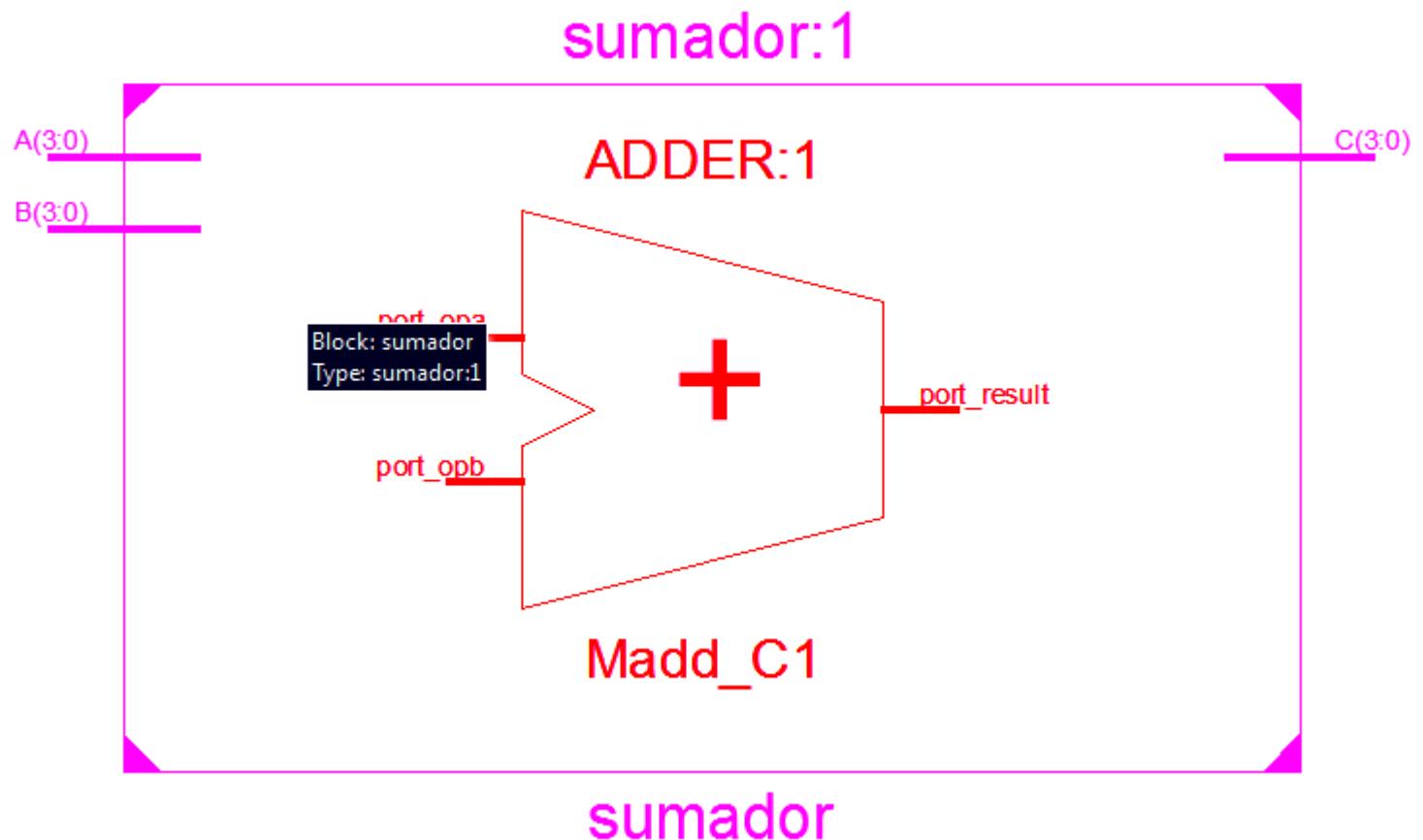


Si todo ha salido  
correctamente



# Implementación (IX)

- Se puede visualizar el esquemático asociado a nuestro diseño VHDL

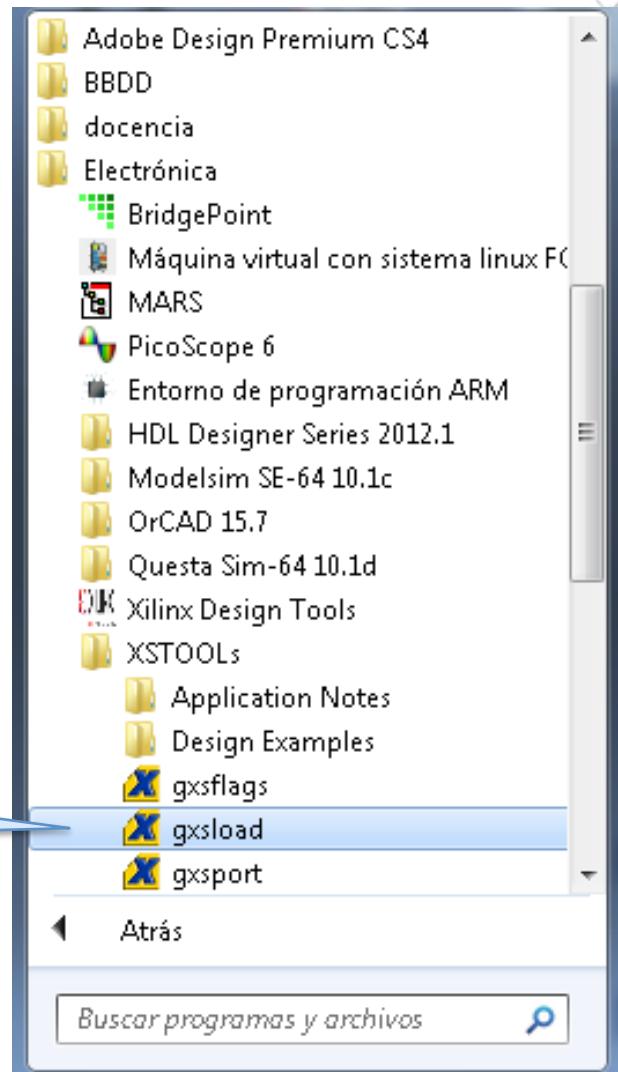




# Descarga .bit sobre FPGA (I)

- Comprobar jumper J9 de la FPGA en la posición XS, de no ser así solicitar otra FPGA
- Para poder volcar el fichero.bit a la FPGA, abrir el programa gxsload, que se encuentra en “Inicio->Electrónica->XSTools”

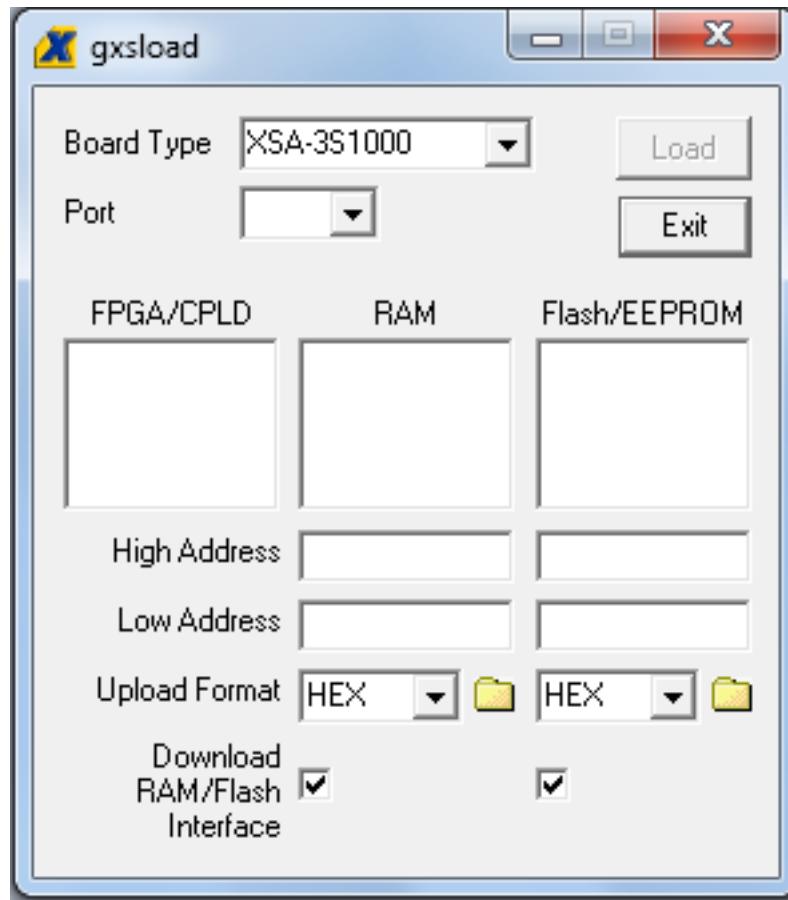
El programa es  
accesible a través  
de esta ruta





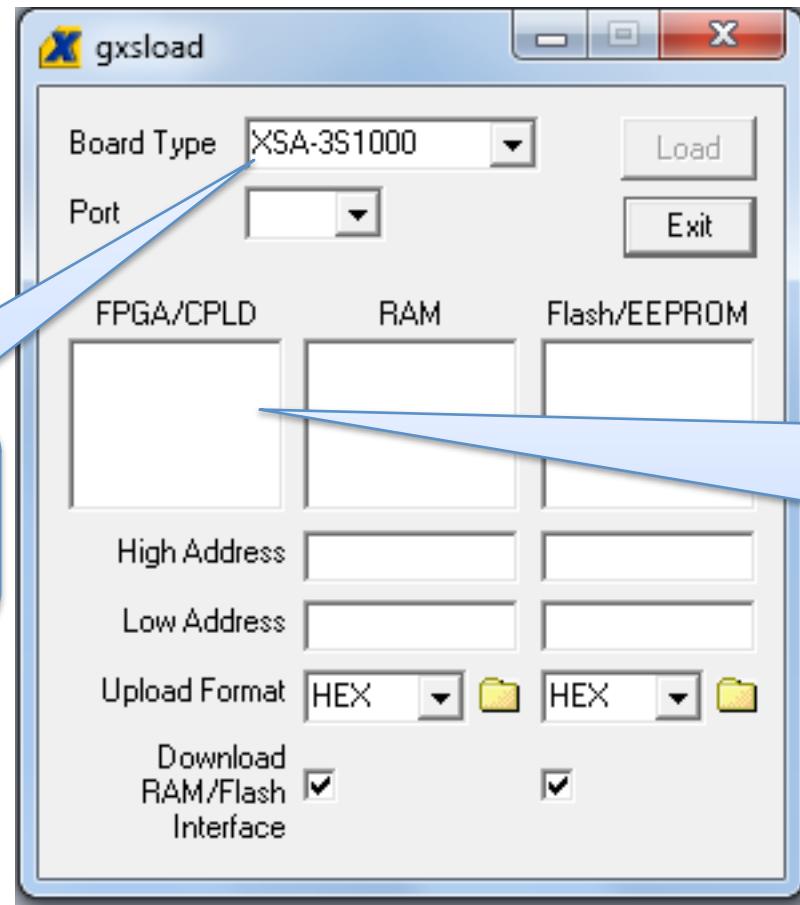
# Descarga .bit sobre FPGA (II)

- Se abrirá la ventana de gxsload



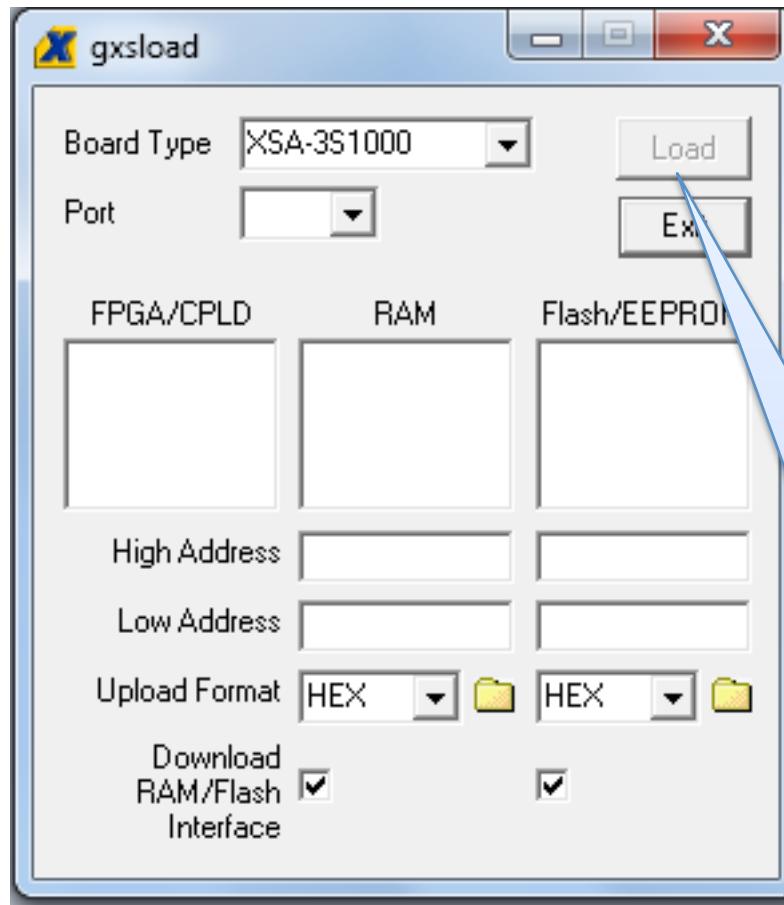


# Descarga .bit sobre FPGA (IV)





# Descarga .bit sobre FPGA (IV)



A continuación,  
pulsamos el botón  
“Load” y la carga  
comienza



# Descarga .bit sobre FPGA (VII)

- Mientras se hace la carga se verá una barra de progreso en la parte inferior de la ventana de gxsload
- Después de esto, ya se puede comprobar sobre la FPGA que nuestro diseño funciona
  - Cambiad las posiciones de los switches de la placa extendida y veréis que el resultado de la suma aparecerá en la barra de LEDs



# Anotaciones



# Práctica 1.b (I)

- Simular e implementar un registro de entrada/salida paralela
- Vamos a estructurarlo en dos partes:
  - Sin divisor de frecuencia (simulación)
  - Con el divisor de frecuencias (implementación)



# Práctica 1.b (II)

```
-- registro de desplazamiento entrada-salida paralelo

library IEEE;
use IEEE.std_logic_1164.all;

entity reg_paralelo is
    port (rst, clk_100MHz, load: in std_logic;
          E: in std_logic_vector(3 downto 0);
          S: out std_logic_vector(3 downto 0));
end reg_paralelo;

architecture circuito of reg_paralelo is

-- Añadir las señales intermedias necesarias
signal clk_1Hz: std_logic;

-- Descomentar para implementación
-- component divisor is
--     port (rst, clk_entrada: in STD_LOGIC;
--           clk_salida: out STD_LOGIC);
-- end component;
```

El clk de la FPGA trabaja a 100 MHz (10 ns), el ojo humano no es capaz de distinguir el parpadeo de los LEDs

El componente divisor genera reloj que trabaja aproximadamente a 1Hz (1s)

Añadiremos como componente un divisor de frecuencia que solo se usará en implementación

Fichero p1b.vhd



# Práctica 1.b (III)

```
begin
-- Descomentar para implementación
-- Nuevo_reloj: divisor port map (rst, clk_100MHz, clk_1Hz);

-- Comentar para la implementacion
clk_1Hz <= clk_100MHz;
```

Para la implementación, necesitamos el divisor de frecuencia

En simulación el reloj es clk\_100MHz y en implementación es clk\_1Hz

```
--Añadimos el resto del código del registro paralelo
process(rst,clk_1Hz)
begin
    if rst='1' then
        S<="0000";
    elsif clk_1Hz'event and clk_1Hz='1' then
        if load='1' then
            S<=E;
            end if;
        end if;
    end process;
end circuito;
```



# Práctica 1.b (IV)

- Para implementar el registro se añade al proyecto el fichero divisor.vhd
  - Se trata de un divisor de frecuencias
  - Se instanciará como componente en el fichero vhd del registro paralelo (descomentando las líneas indicadas en el fichero)
  - Solo se usará para la implementación sobre FPGA
  - Si hacemos simulación todas las referencias a este divisor deben ser comentadas



# Implementación Práctica 1.b

- Pin del reloj:

NET clk\_100MHz LOC=T9;

- Hasta que no se resetee el circuito no funcionará correctamente



# Anotaciones



# Calificación (I)

- El estudiante debe acudir al laboratorio con la práctica 1.a y 1.b simulada e implementada.
- El estudiante debe enseñar al profesor del laboratorio cada una de las partes funcionando sobre la FPGA.
- El estudiante contestará un test en el Campus Virtual el 15 de octubre sobre los apartados 1.a y 1.b de la práctica.
  - Para ello el estudiante cargará un fichero de simulación (ver simulación para calificación) y responderá a las preguntas de Moodle.
- Se realizará una apartado avanzado relacionado con lo hecho en casa durante la sesión de laboratorio
- La práctica 1 no se recupera.



# Calificación (III)

- Simulación para calificación:
  - En la sesión de laboratorio el alumno dispondrá de dos ficheros VHDL
    - simu\_1a.vhd
    - simu\_1b.vhd
  - Añadir los ficheros de simulación
    - Add Source
  - Simular y contestar a las preguntas del test