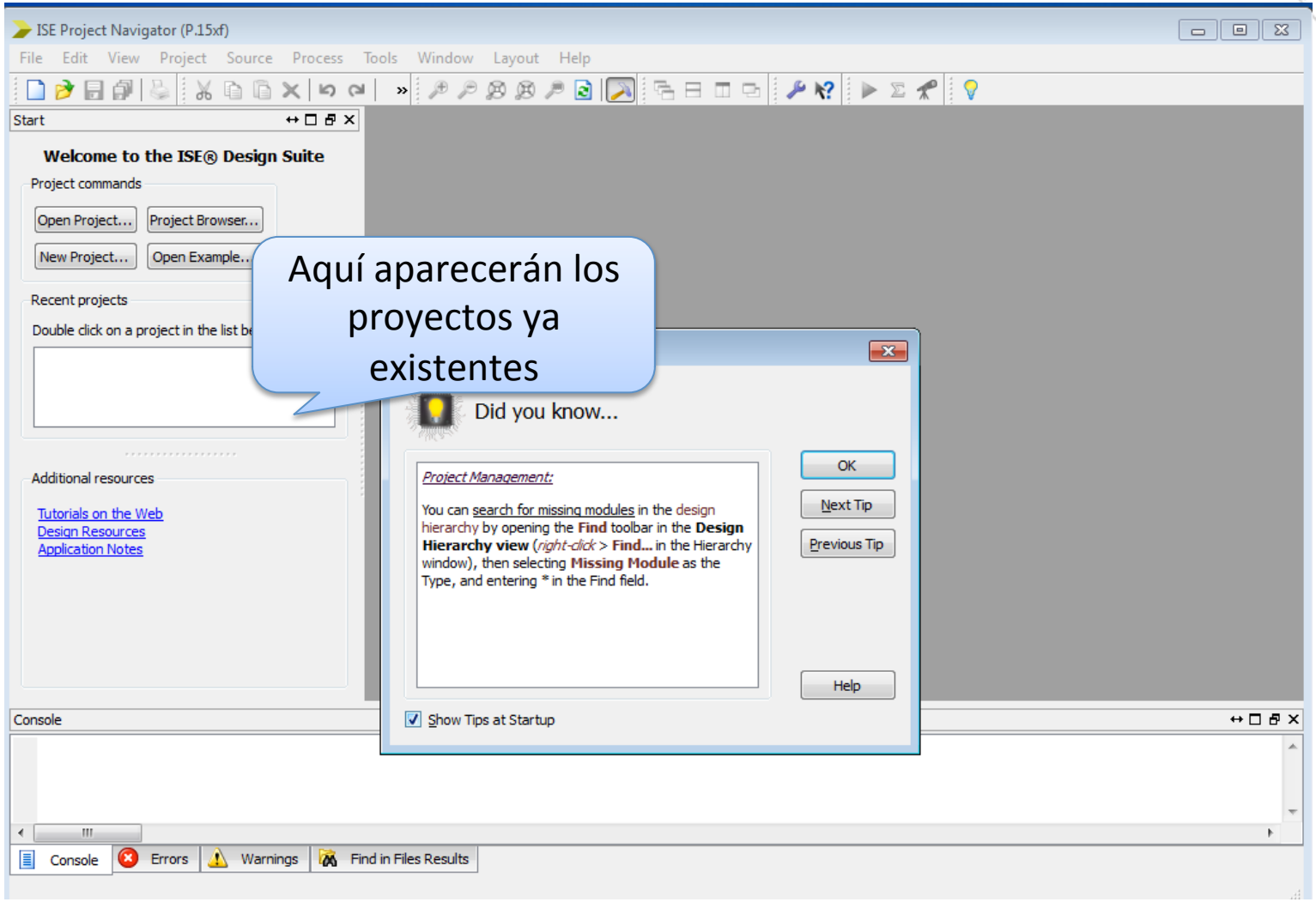
Práctica 1

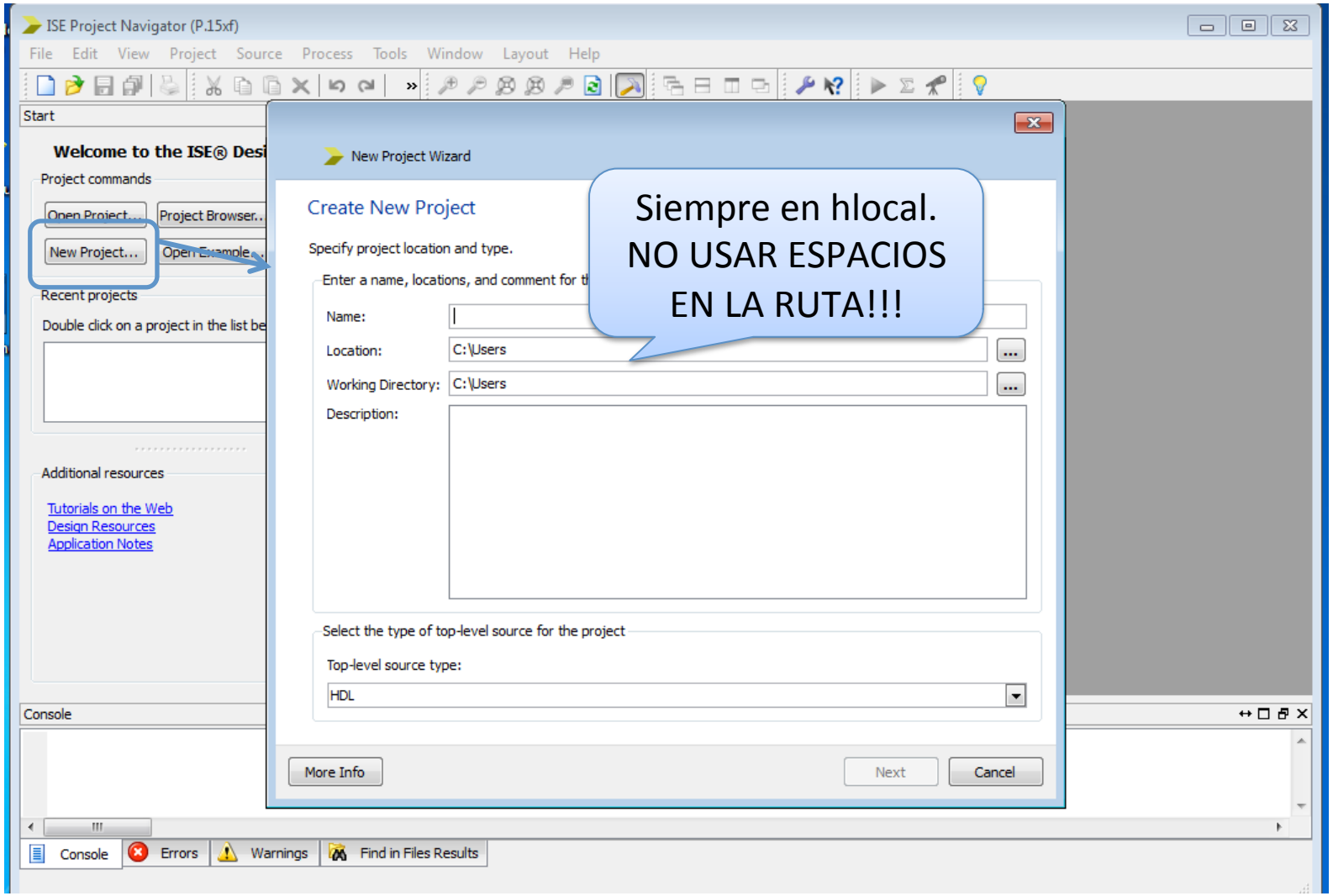
- Utilización de la herramienta Xilinx ISE con VHDL
- Diseño y simulación de circuitos sencillos

Pantalla de bienvenida

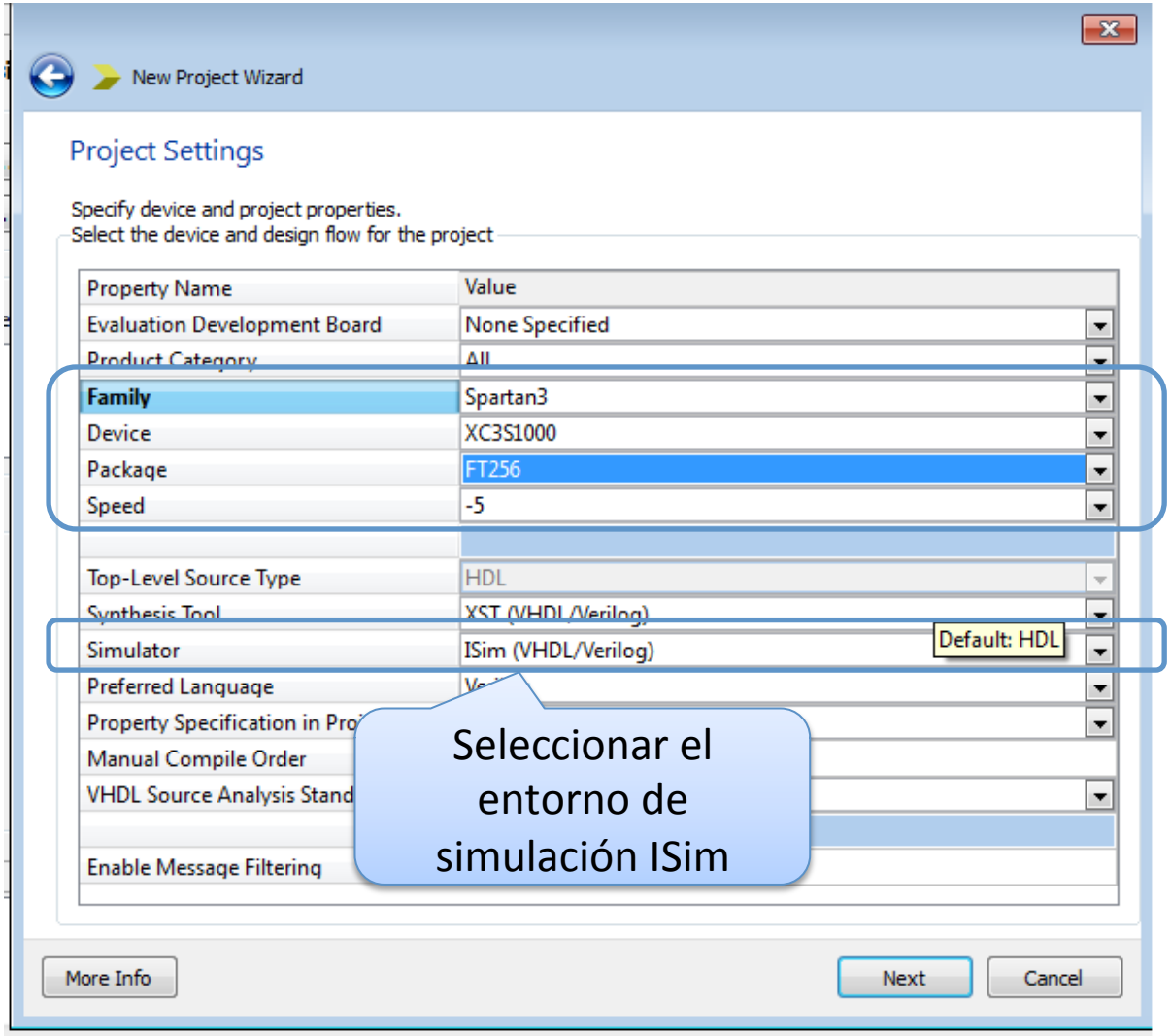


Crear un proyecto (I)

- Desde la ventana de bienvenida



Seleccionar dónde se va a implementar



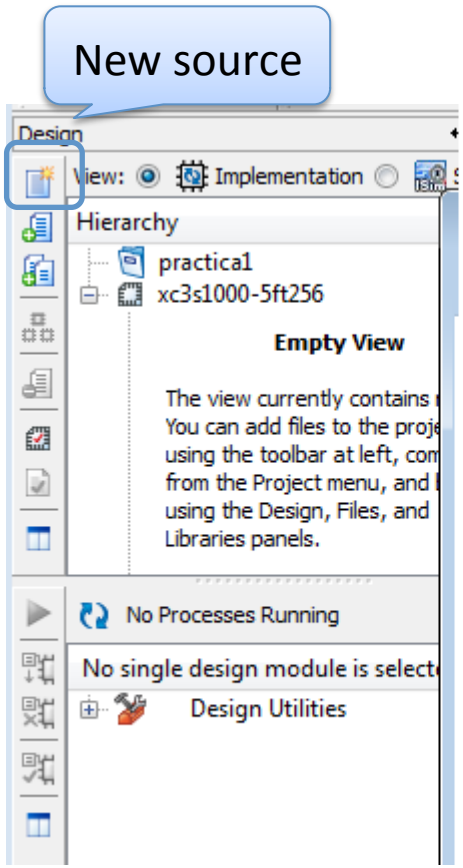
Seleccionar el entorno de simulación ISim

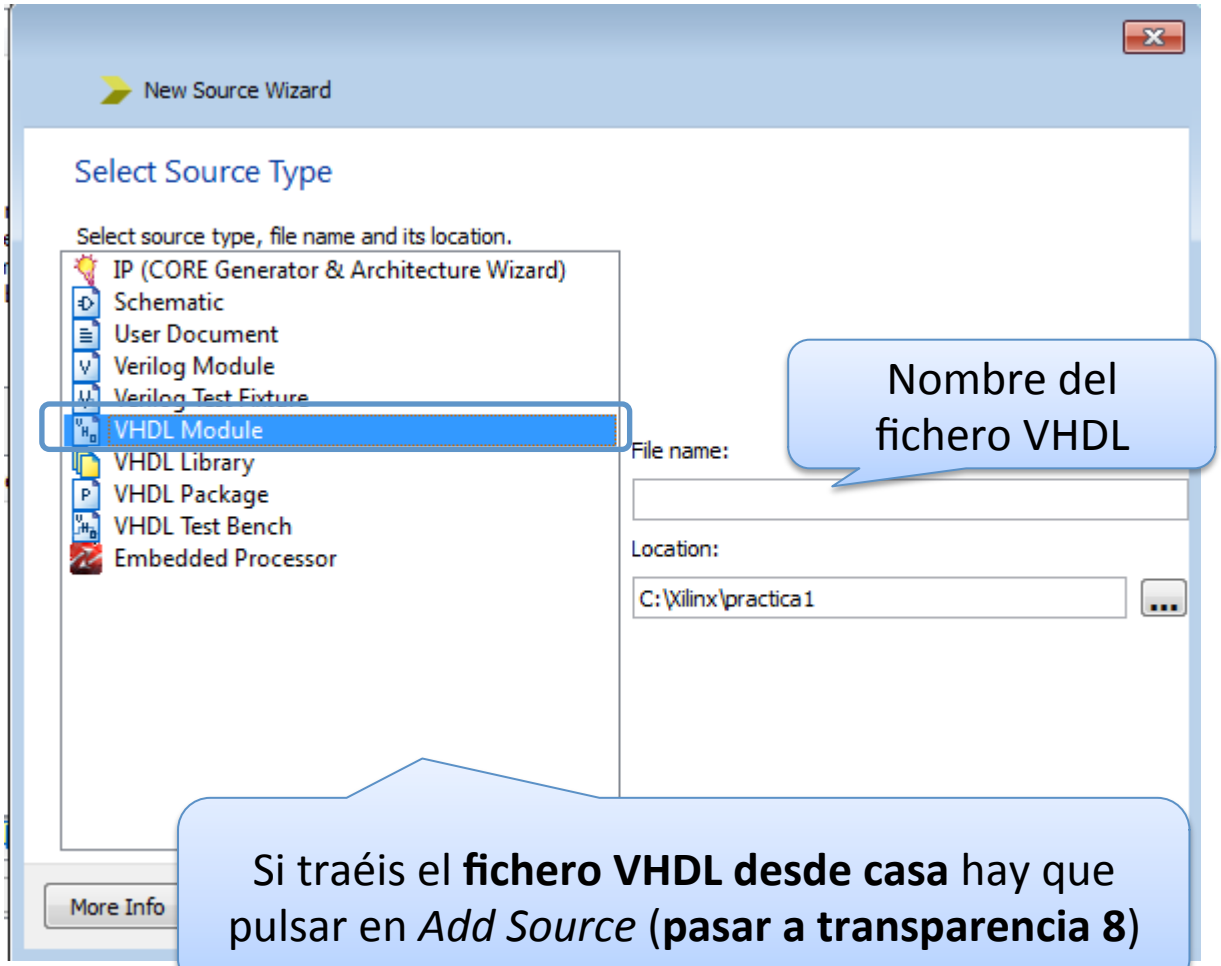
Spartan 3
XC3S1000
FT256
-5

toc

Crear un archivo VHDL (I)

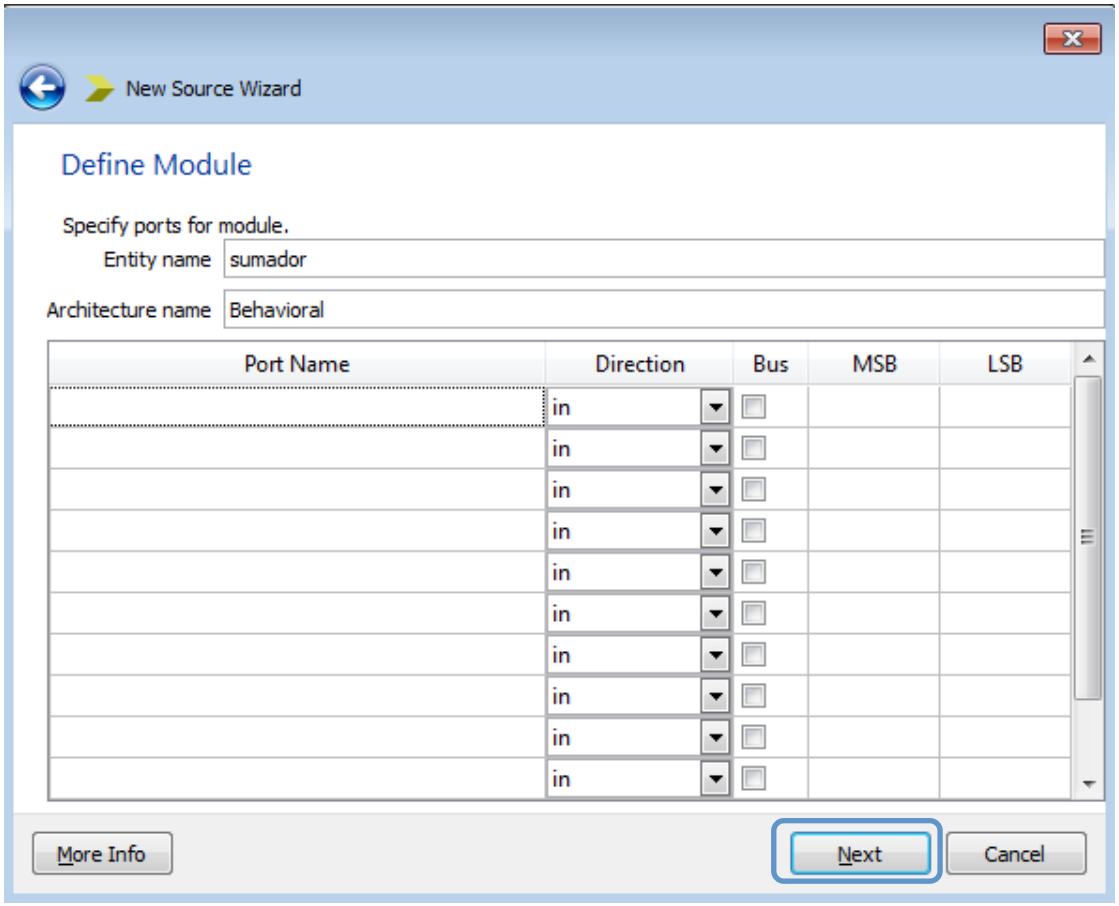
- Siempre se inicia creando un fichero tipo VHDL Module





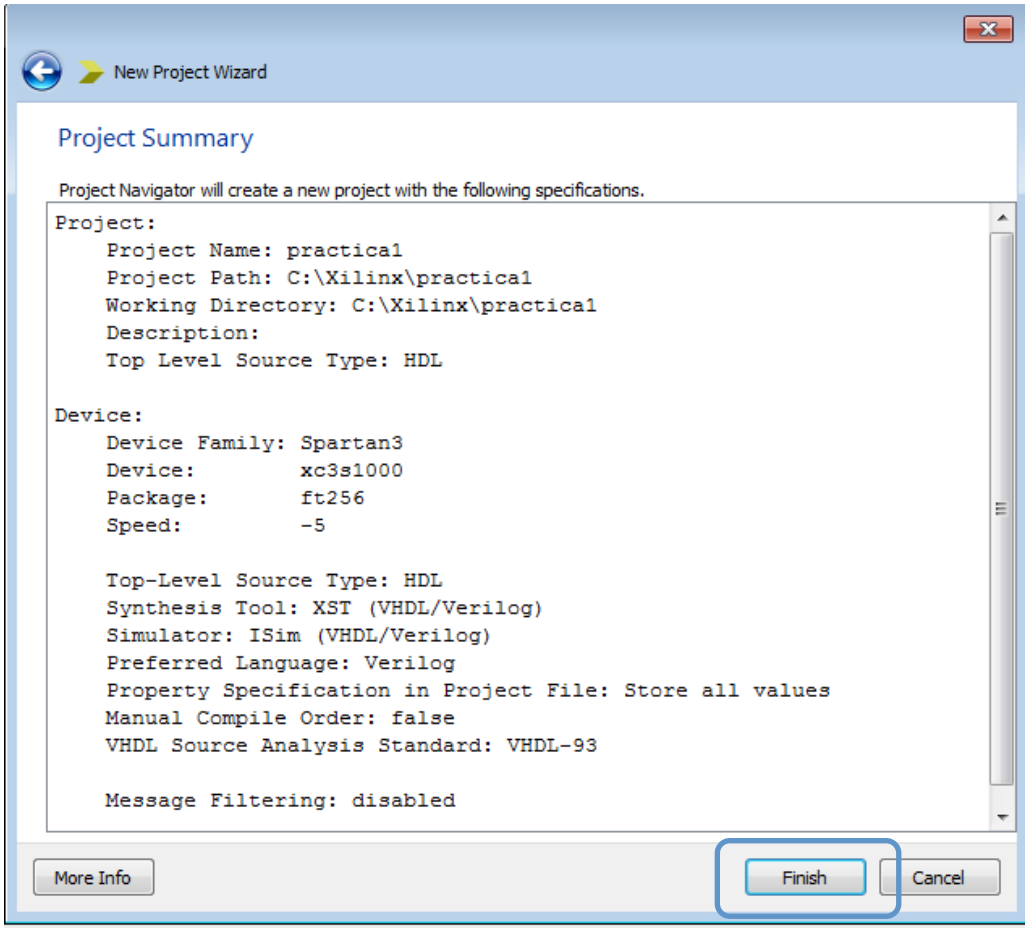
Crear un archivo VHDL (II)

- Wizard para definir las señales de entrada y salida (NO UTILIZAR. Hacer click en “Next”)



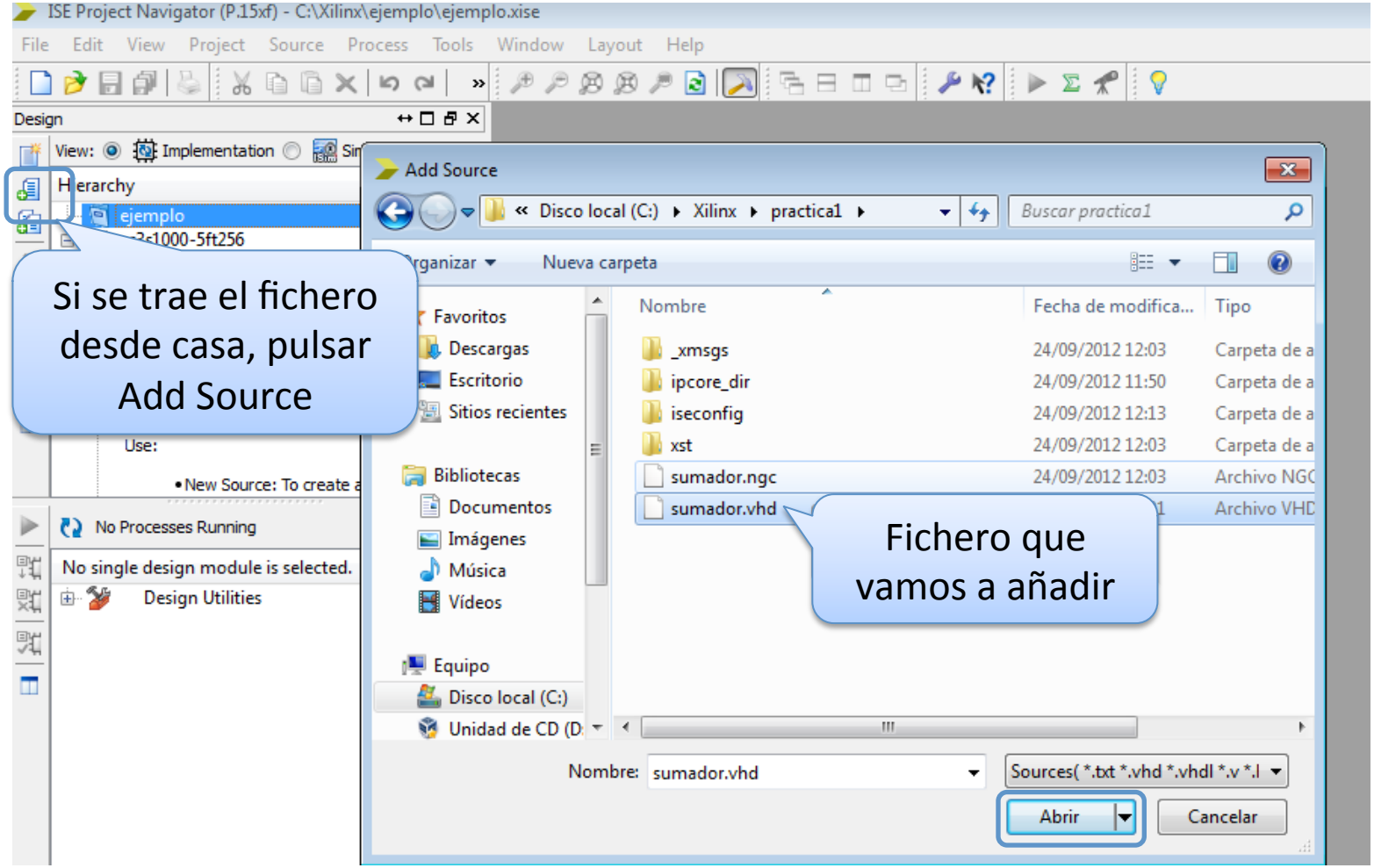
Crear un archivo VHDL (III)

- Todo ha salido correcto



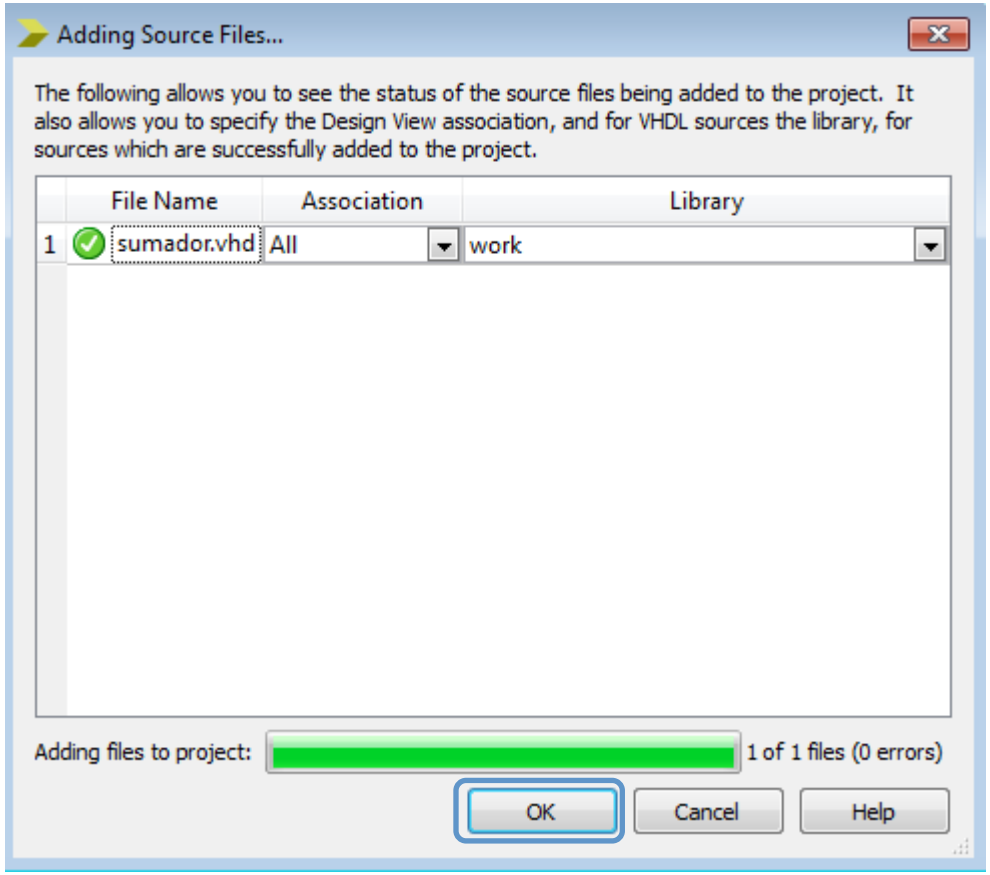
Añadir fuentes a nuestro proyecto

- Se pueden añadir otros ficheros fuente a nuestro proyecto
- Los ficheros a añadir deben ser previamente grabados en el directorio hlocal



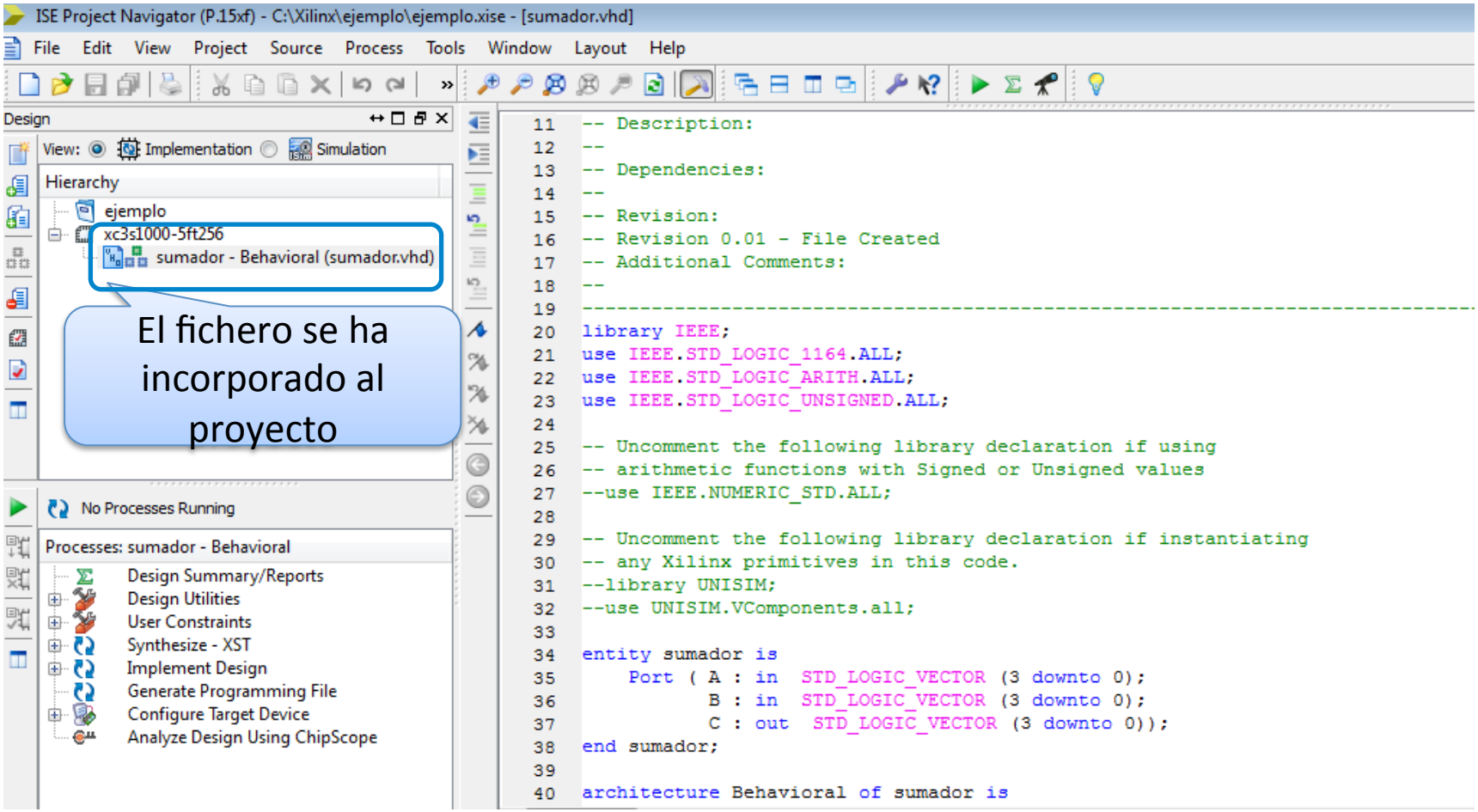
Añadir fuentes a nuestro proyecto

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



Añadir fuentes a nuestro proyecto

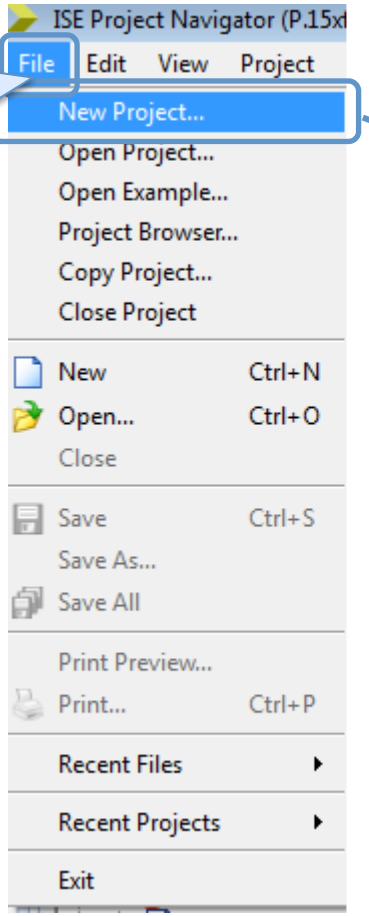
- El fichero se ha añadido correctamente

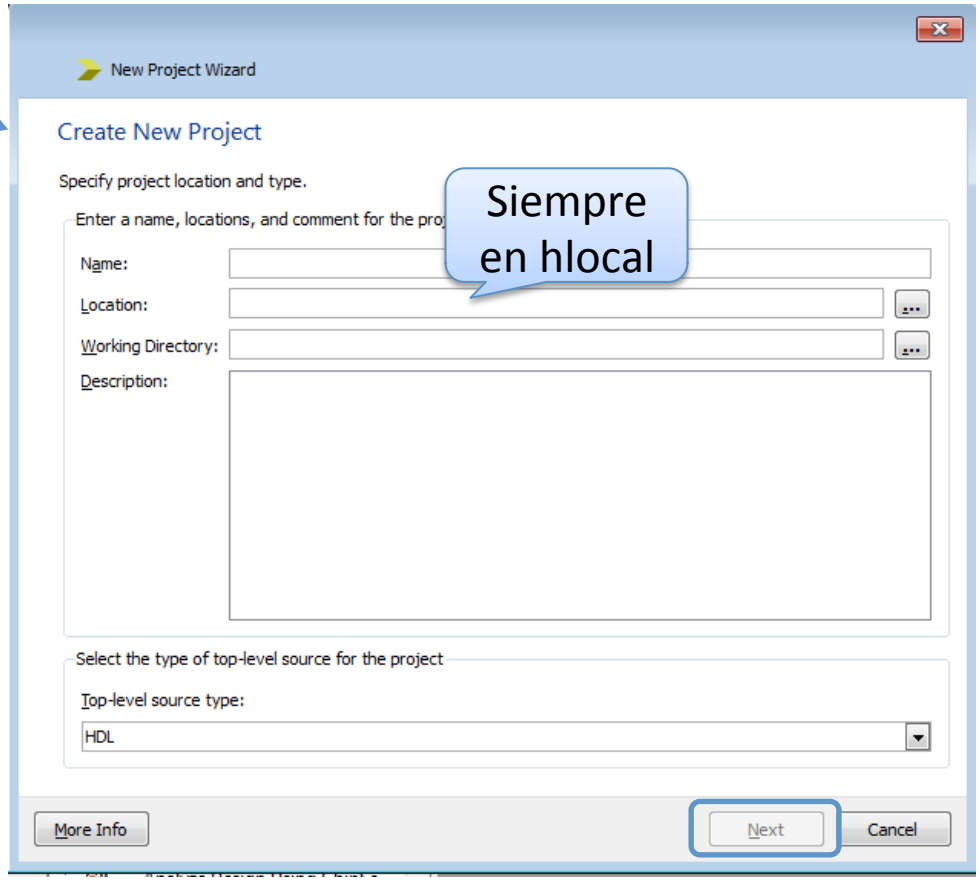


Crear un proyecto (otra vez)

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

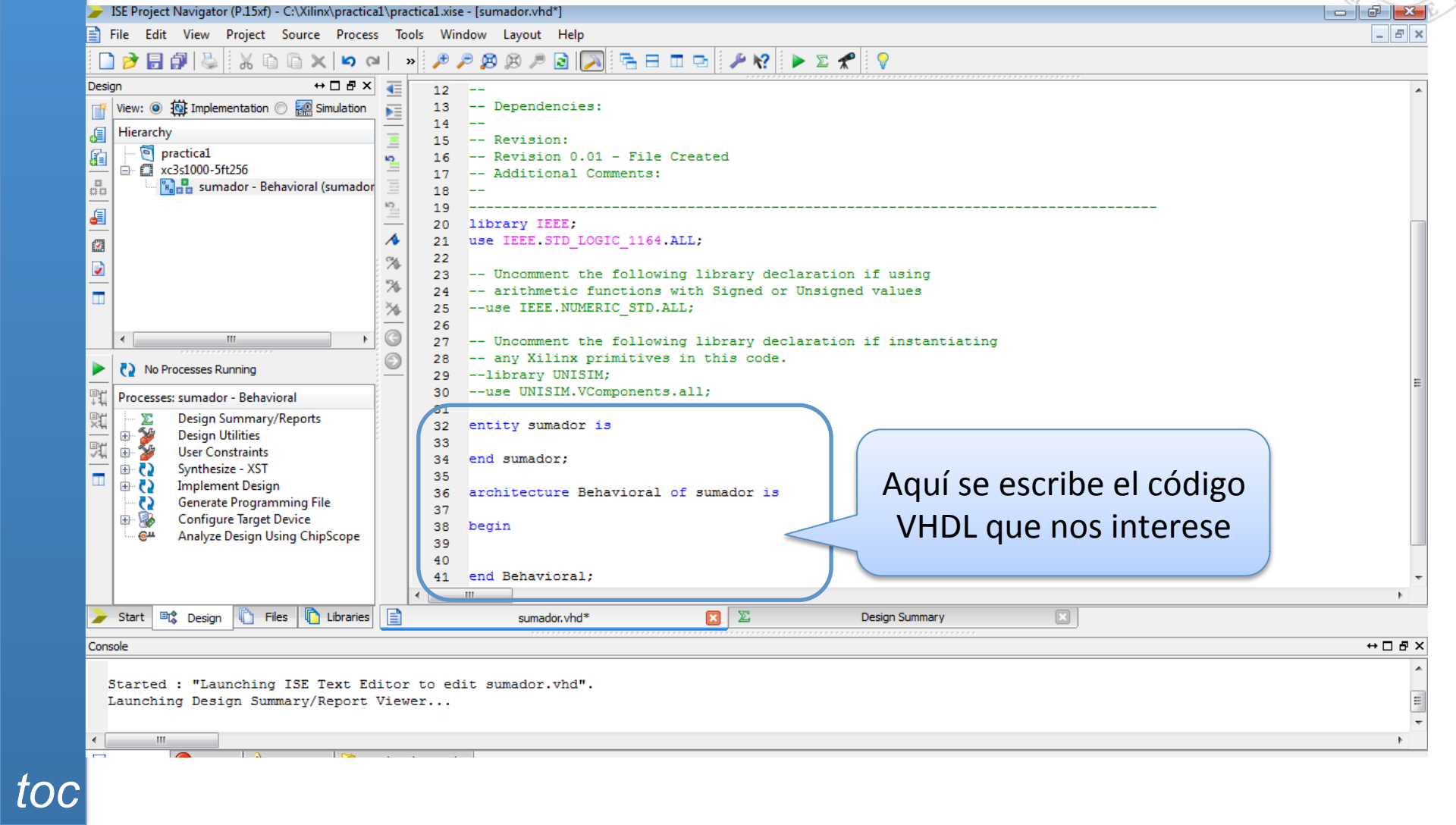
Para crear un nuevo proyecto





Anotaciones

Pantalla de desarrollo



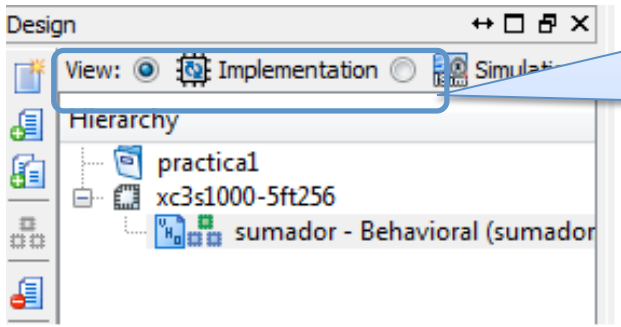
Práctica 1.a

```
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

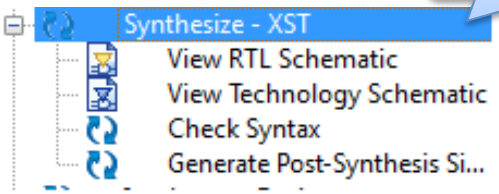
```
entity sumador is
port (A, B: in std_logic_vector(3 downto 0);
         C: out std_logic_vector(3 downto 0));
end sumador;

architecture beh of sumador is
-- No hace falta definir señales intermedias
begin

C <= A + B;

end beh;
```

Sintetizar el diseño

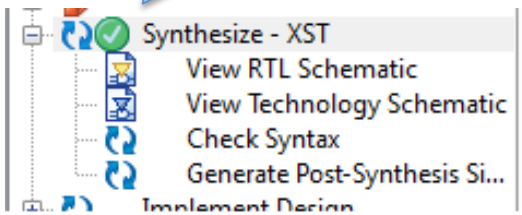


Para poder implementar el circuito tenemos que estar en la opción Implementation

Doble click



Si sale este símbolo verde es que todo ha salido perfecto

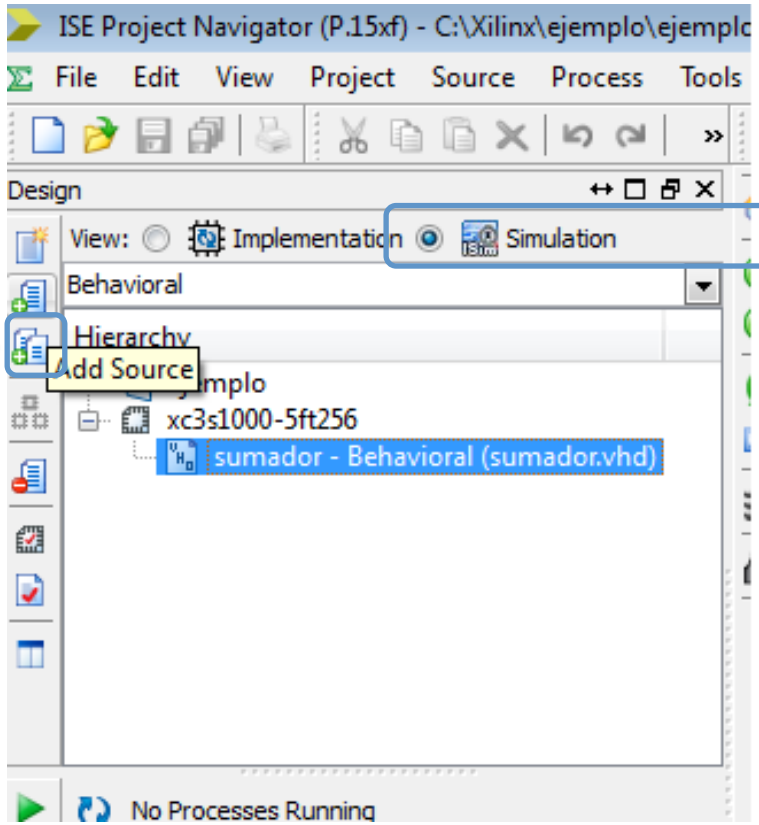


Comprobar que el Diseño es Correcto

- Para realizar está 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

Test de simulación (I)

- Añadir el fichero de simulación simsum.vhd



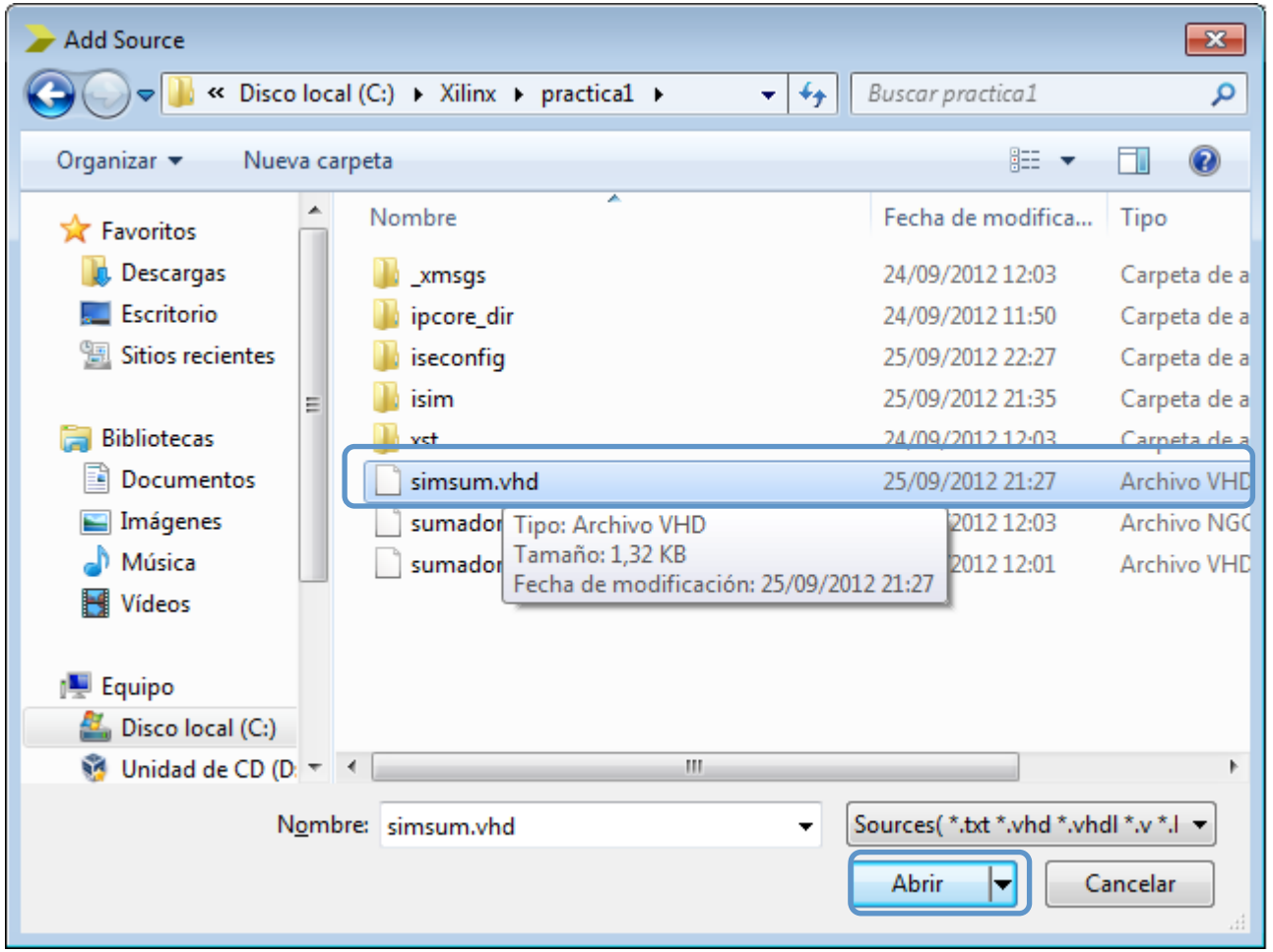
Pulsamos sobre Add Source

Marcamos la opción Simulation

Importante: El fichero simsum.vhd se debe grabar en hlocal antes de añadirlo al proyecto

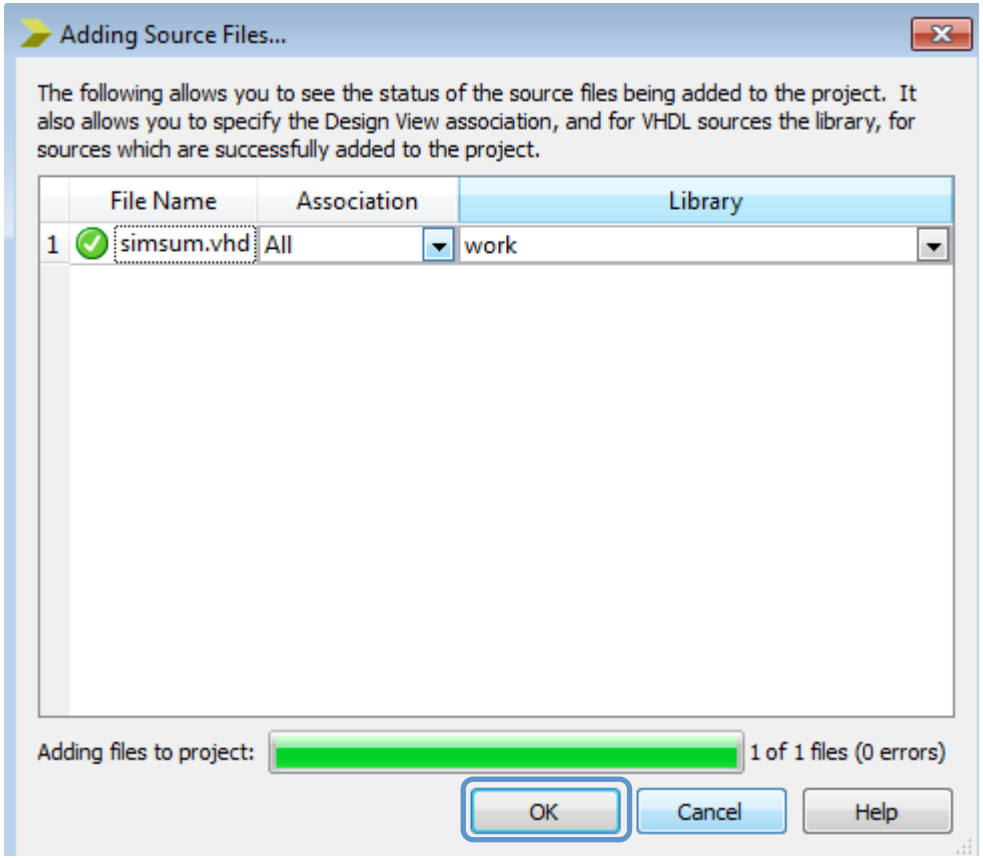
Test de simulación (II)

- Seleccionamos el fichero simsum.vhd



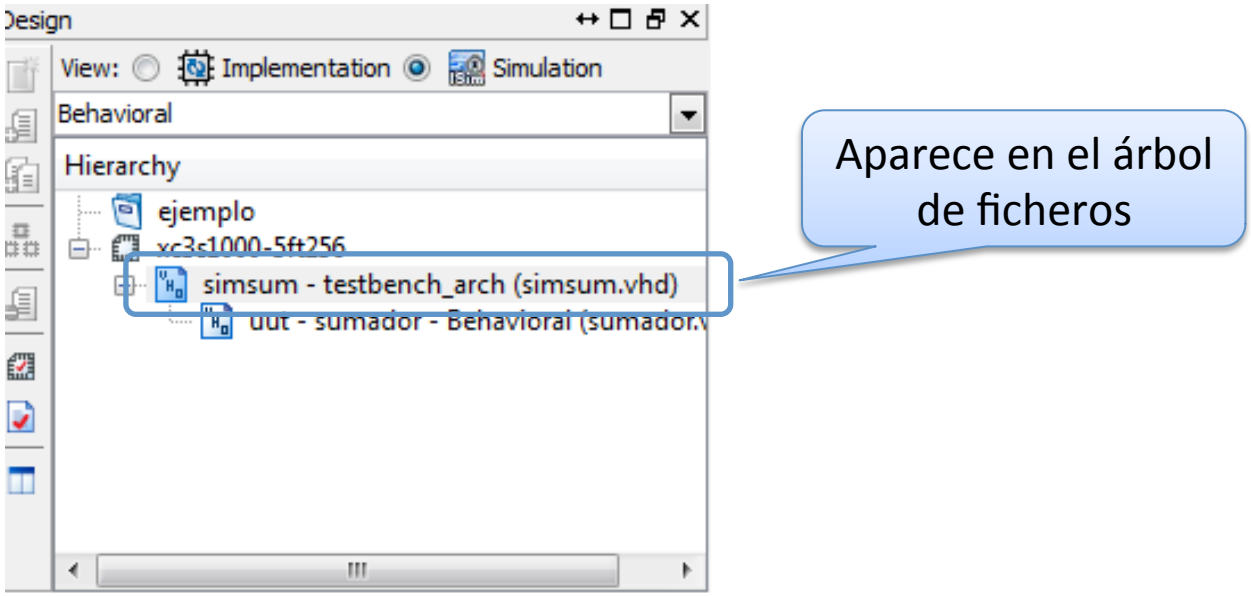
Test de simulación (III)

- El fichero se está añadiendo al proyecto



Test de simulación (IV)

- El fichero se ha añadido correctamente

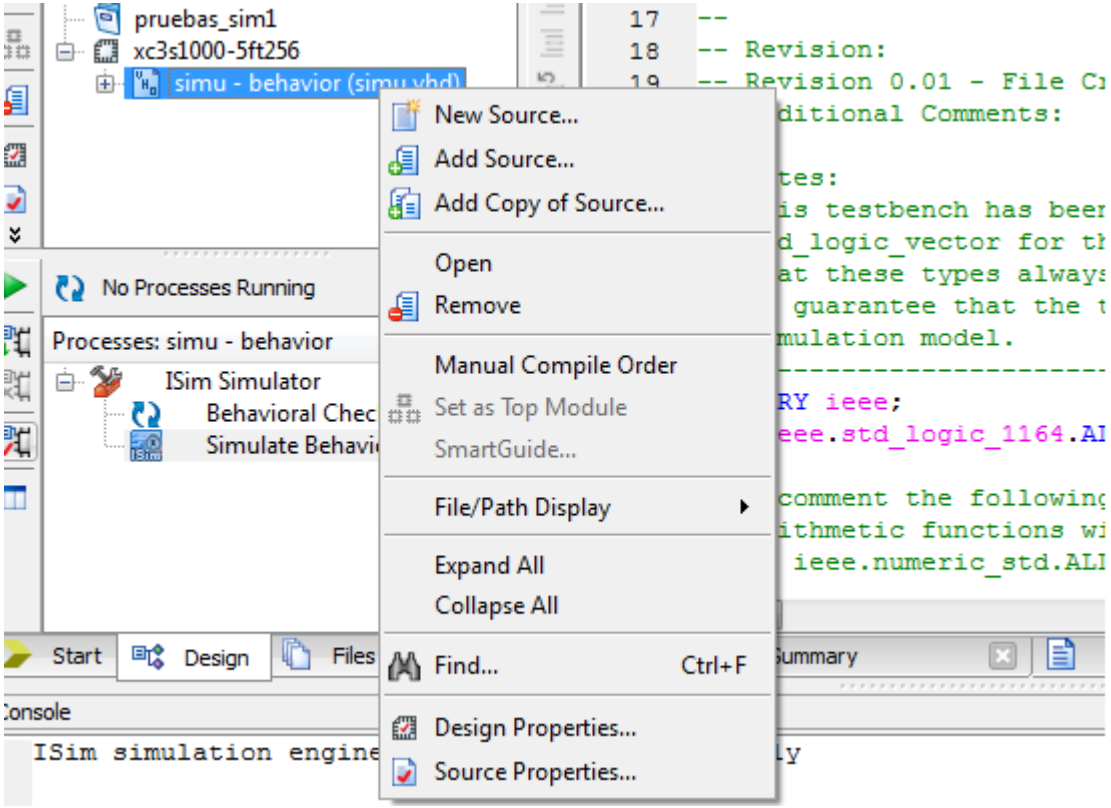


Fichero de simulación (I)

- 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

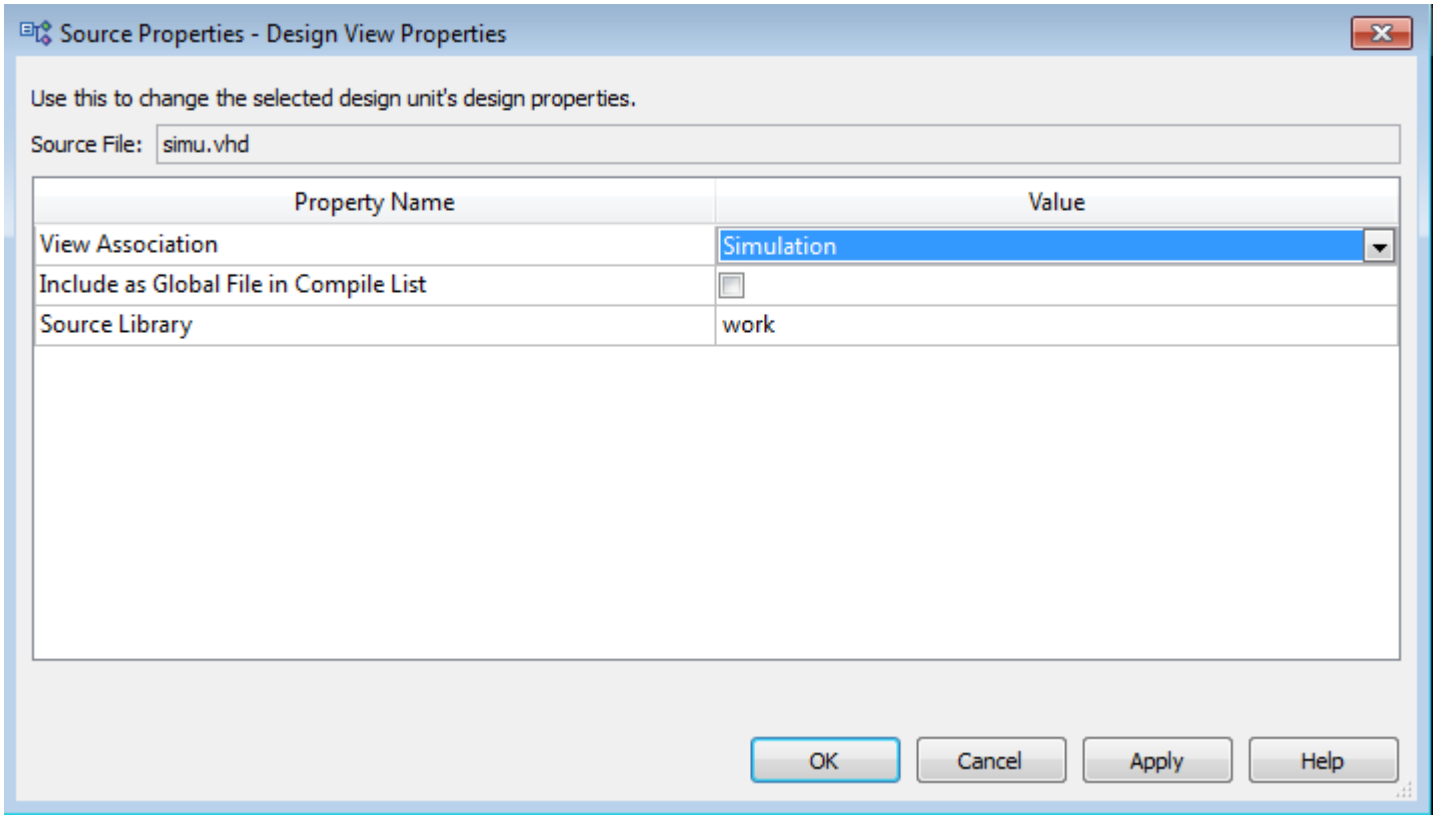
Fichero de simulación (II)

- 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 (lo tenéis en el CV)

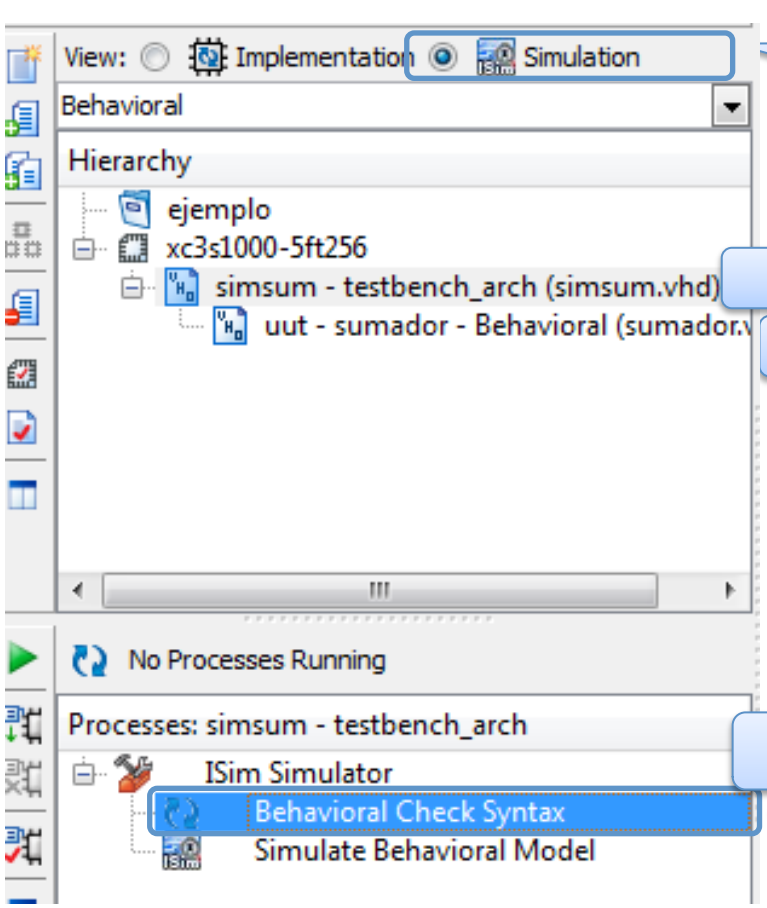
```
-- Añadimos las librerias necesarias
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;

--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 (lo tenéis en el CV)

```
Begin
-- Instanciacion de la unidad a simular
   uut: sumador port map (
          A => op1   ,
          B => op2   ,
          C => result
        );
   -- Proceso de estimulos
   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

- Desplegar el menú de simulación



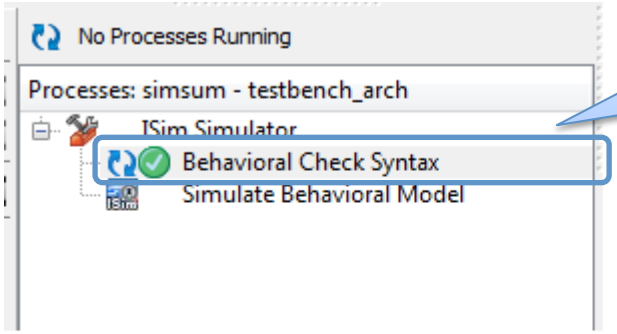
Para poder simular el circuito tenemos que estar en la opción Simulation

nombre del test bench

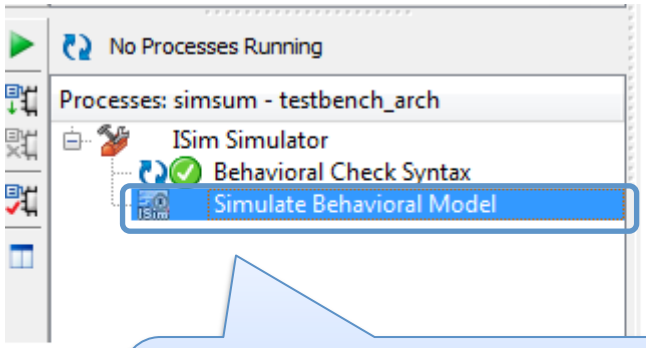
nombre de la entity que queremos simular

Doble click

Simulación



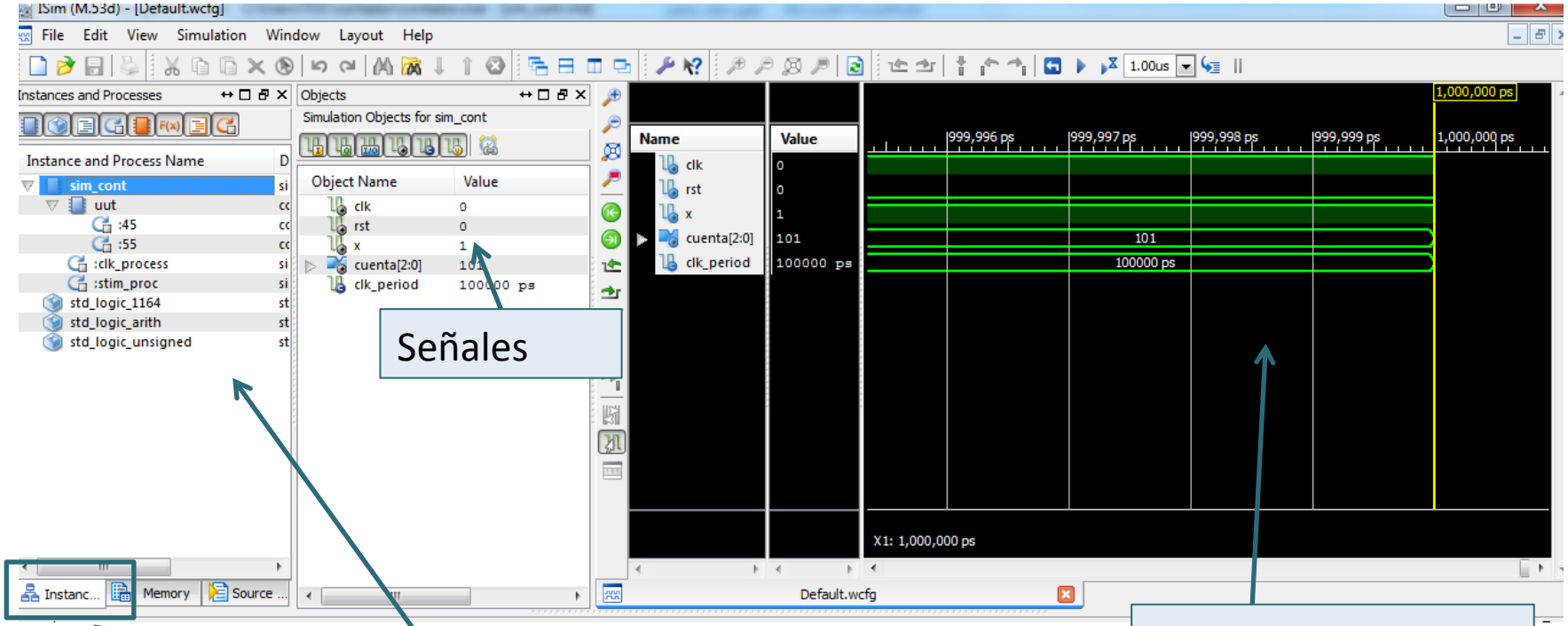
La simulación se ha ejecutado correctamente



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

Simulación

- Ventana simulador ISim

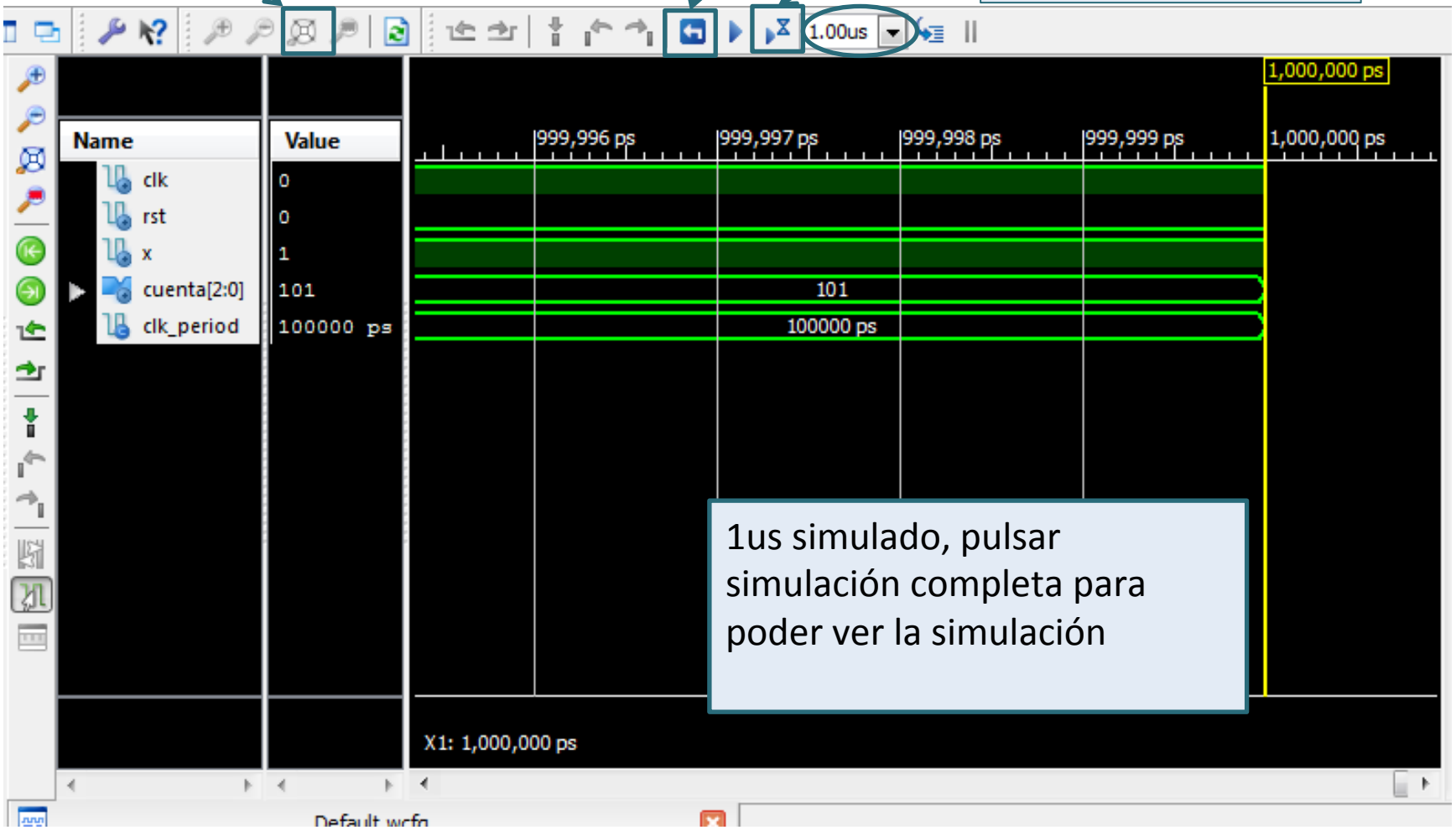


Simulación

Ver simulación completa

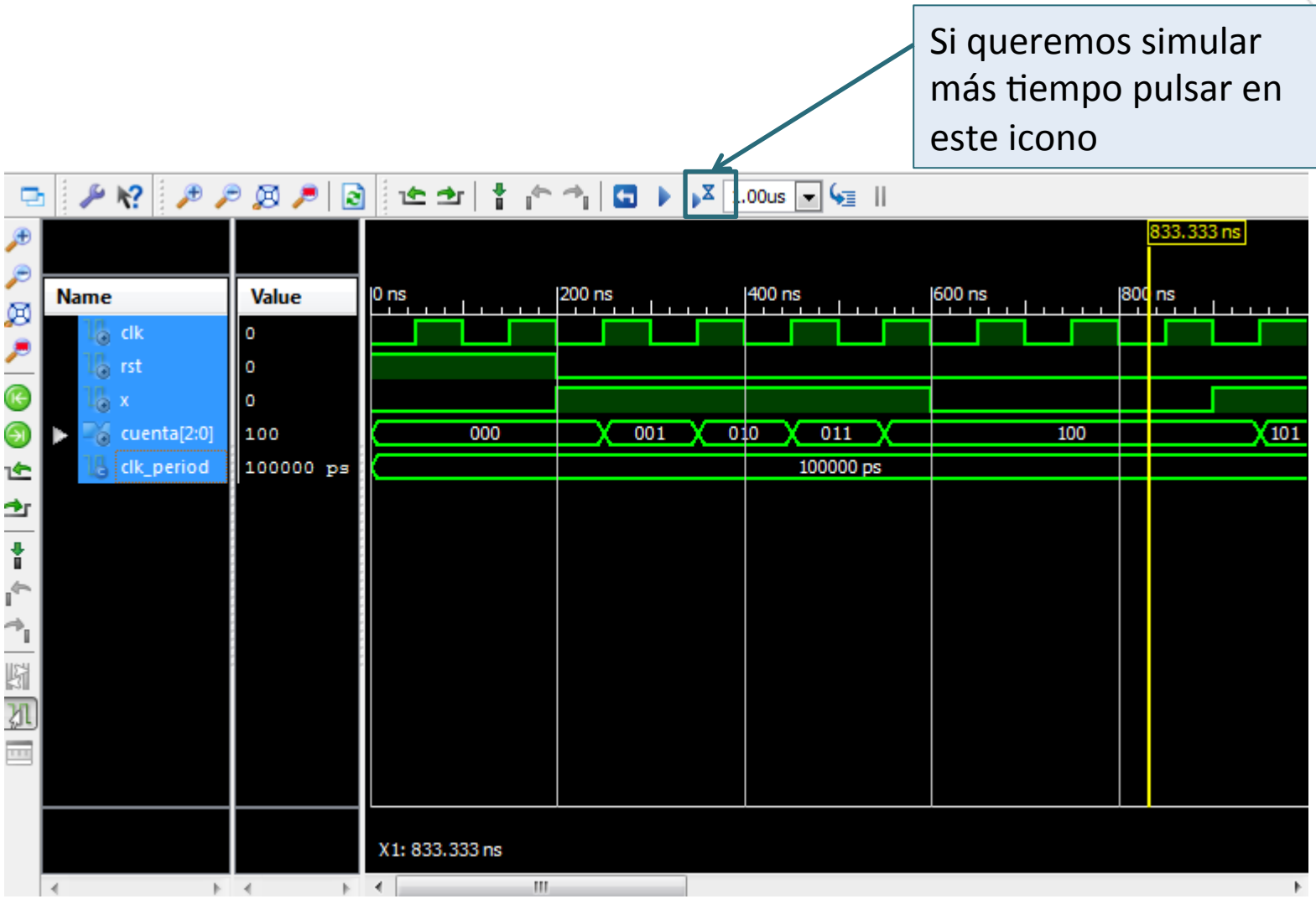
Reiniciar simulación

Cada vez que se pulsa este icono se simula 1us



1us simulado, pulsar simulación completa para poder ver la simulación

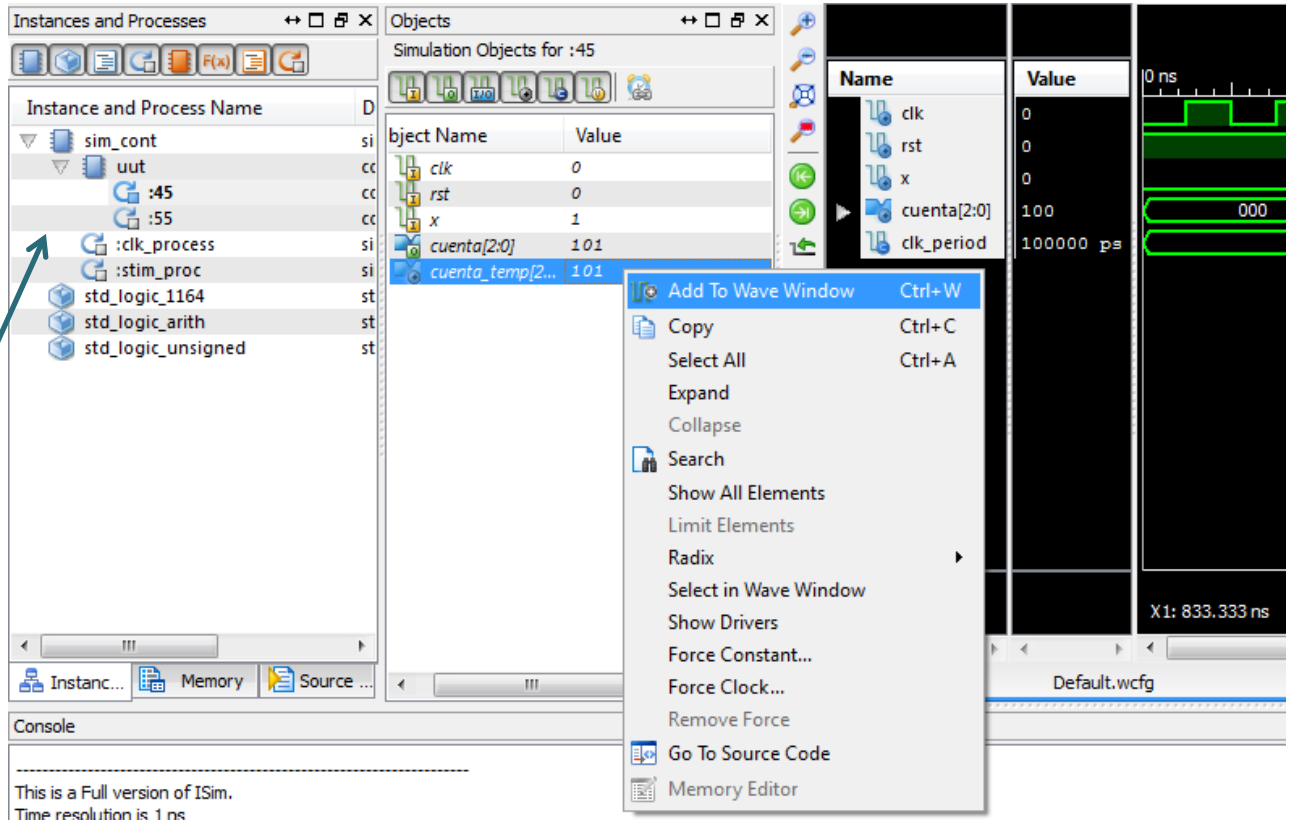
Simulación


Si queremos simular más tiempo pulsar en este icono
.00us
833.333 ns
Name
Value
clk
rst
x
cuenta[2:0]
clk_period
0
0
0
100
100000 ps
0 ns
200 ns
400 ns
600 ns
800 ns
000
001
010
011
100
101
100000 ps
X1: 833.333 ns

Simulación

- 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

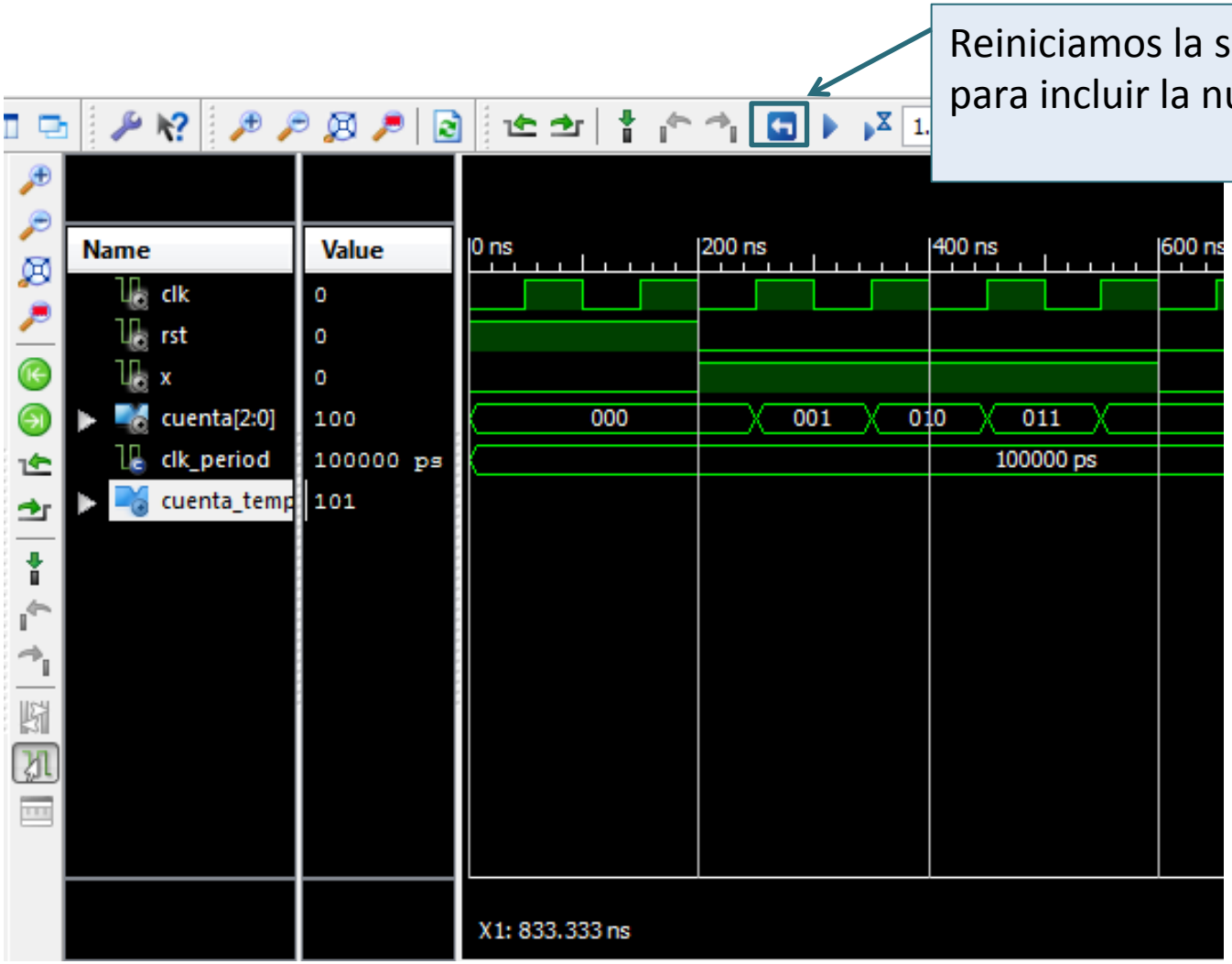
Buscar las señales internas, para añadirlas pulsar el botón derecho y elegir la opción marcada



toc

Simulación

- La señal se añadirá a la ventana de simulación



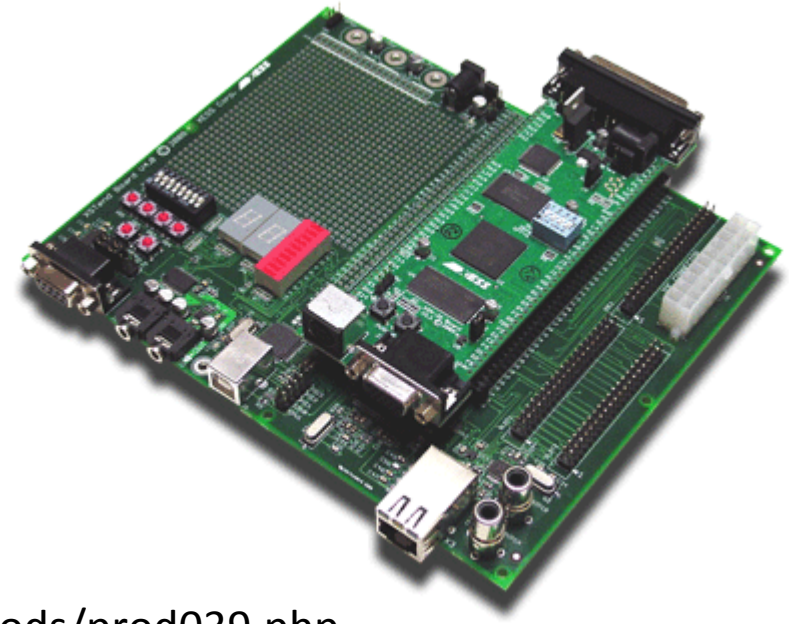
Anotaciones

Implementación

- 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 dónde se le indica a la herramienta lo anterior

Implementación

- 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

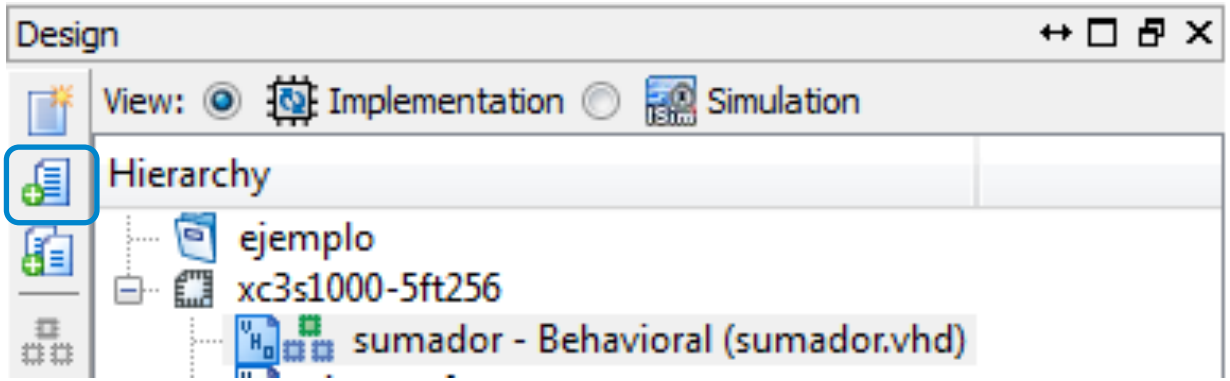
- Fichero UCF
  - En el Campus Virtual encontraréis el fichero UCF completo
  - Aquí aparecen los pines particulares que se necesitan para la práctica 1.a

```
#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 entrada/salida de la entity
- El puerto A<0> significa “puerto A, bit 0”
- A cada puerto de la entity se le asigna un pin físico 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

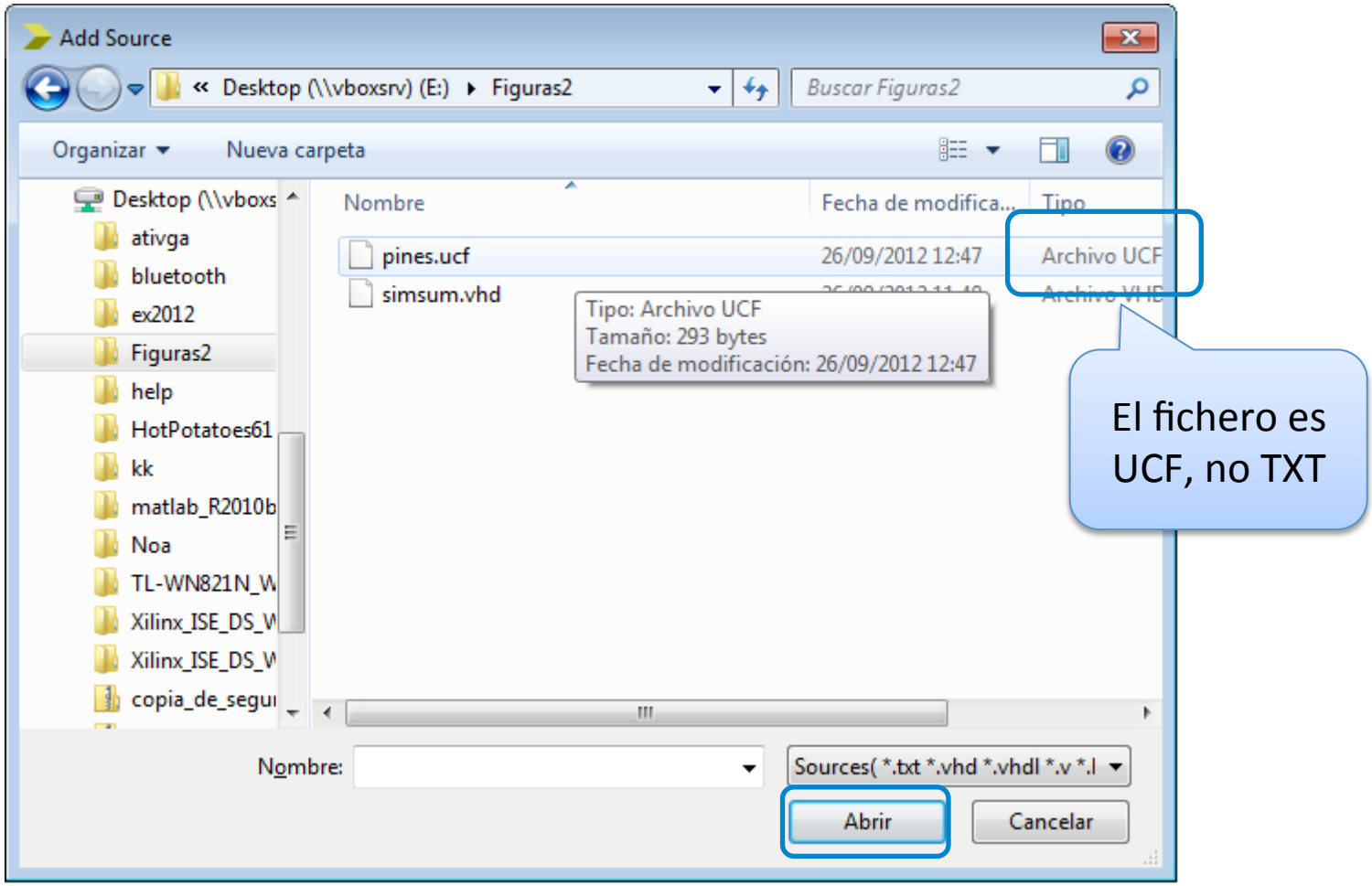
Implementación

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



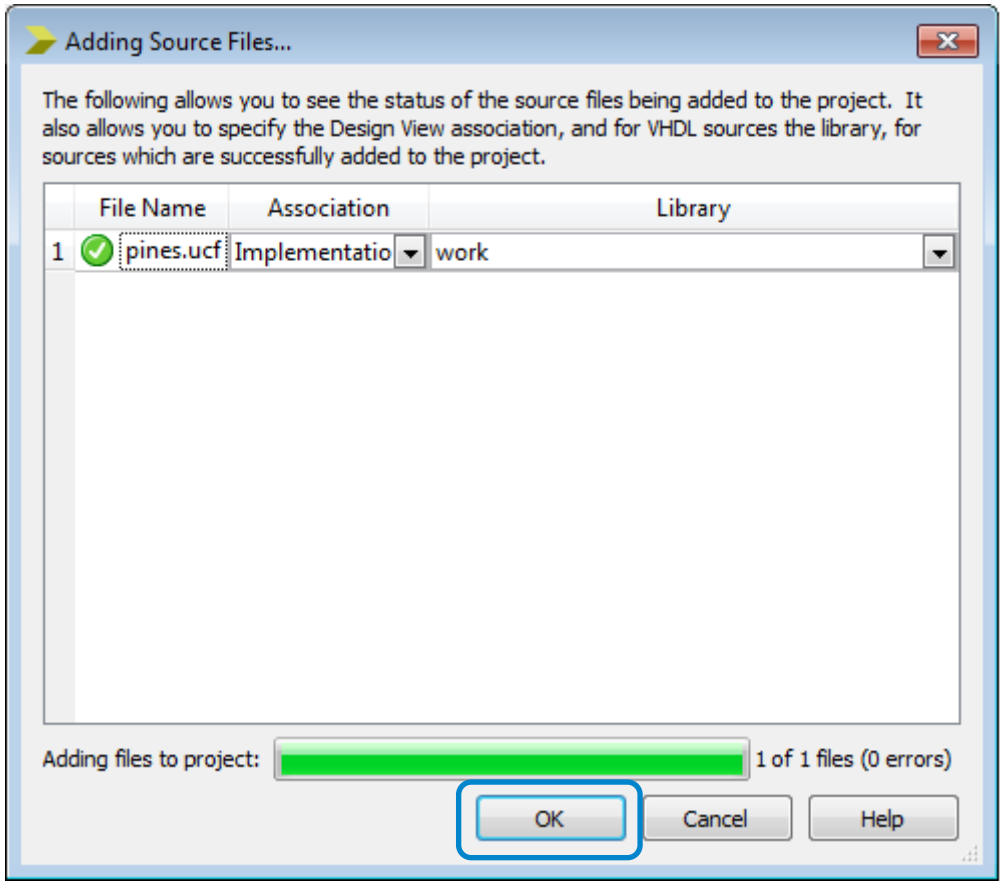
Implementación

- Seleccionamos el fichero pines.ucf



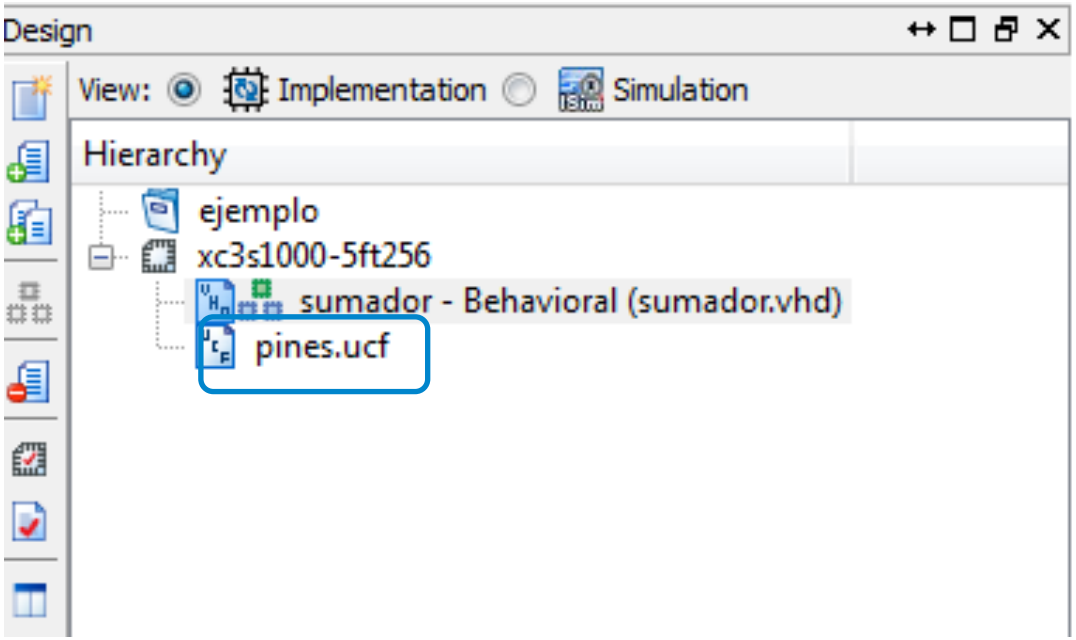
Implementación

- El fichero se añade al proyecto

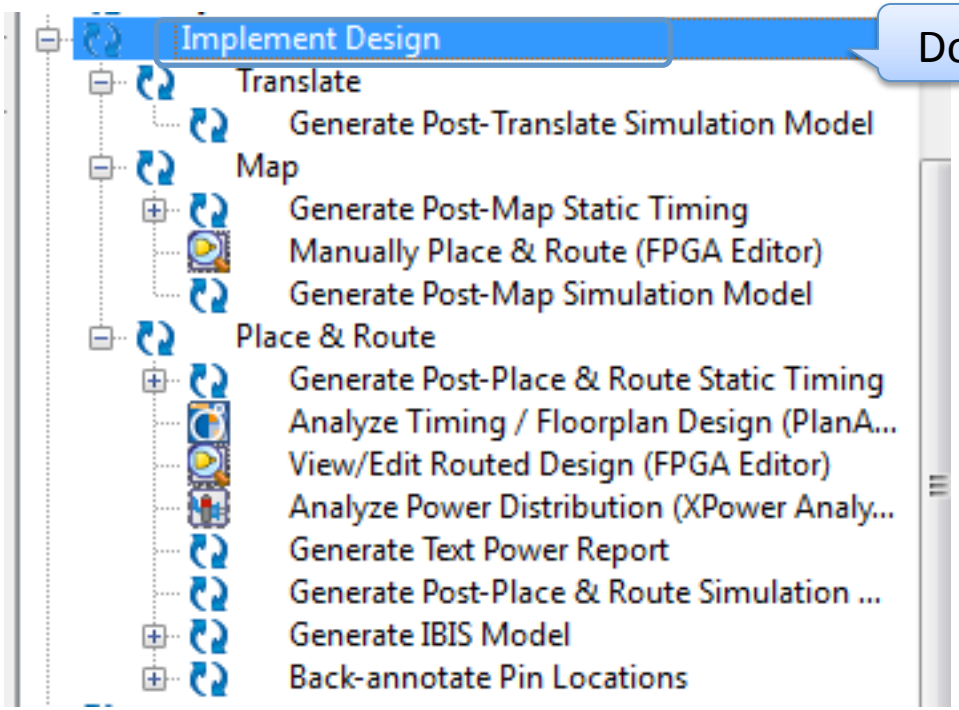


Implementación

- El fichero aparece en el árbol de jerarquía



Implementación



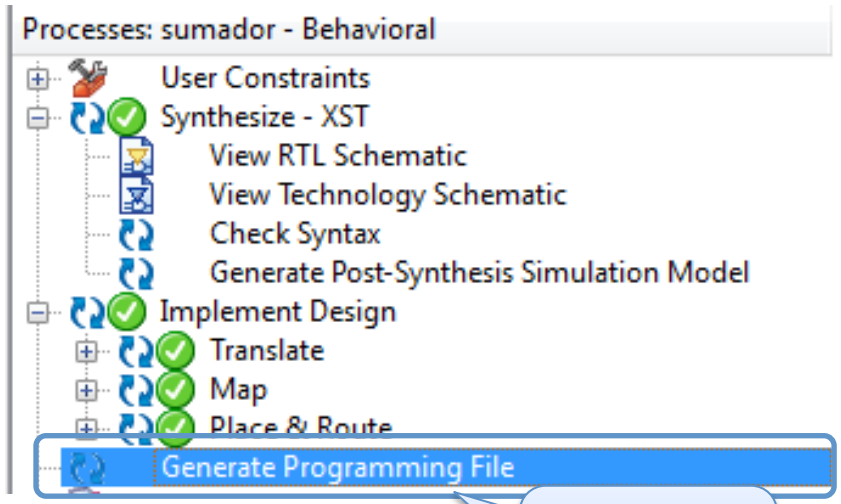
Doble click

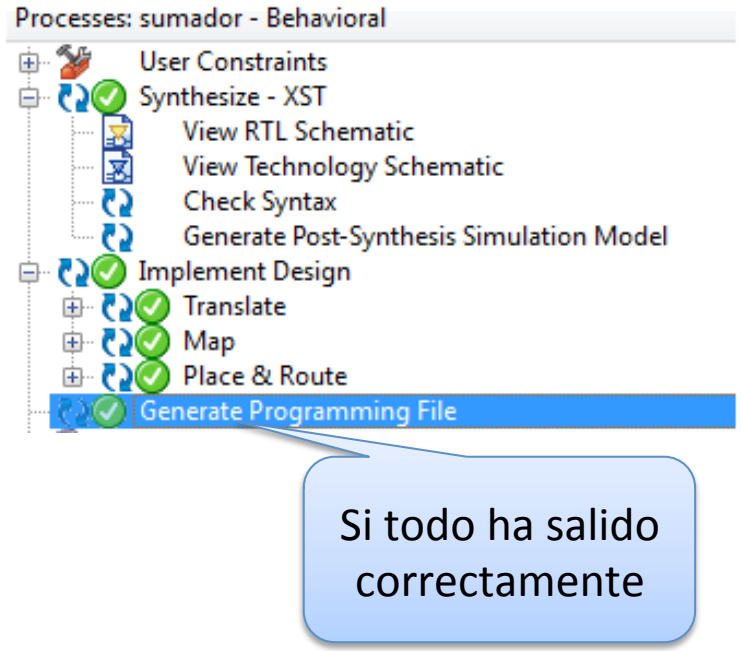
Si todo está correcto aparecerá en:

- Implementation Design
- Map
- Place & Route

Implementación

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

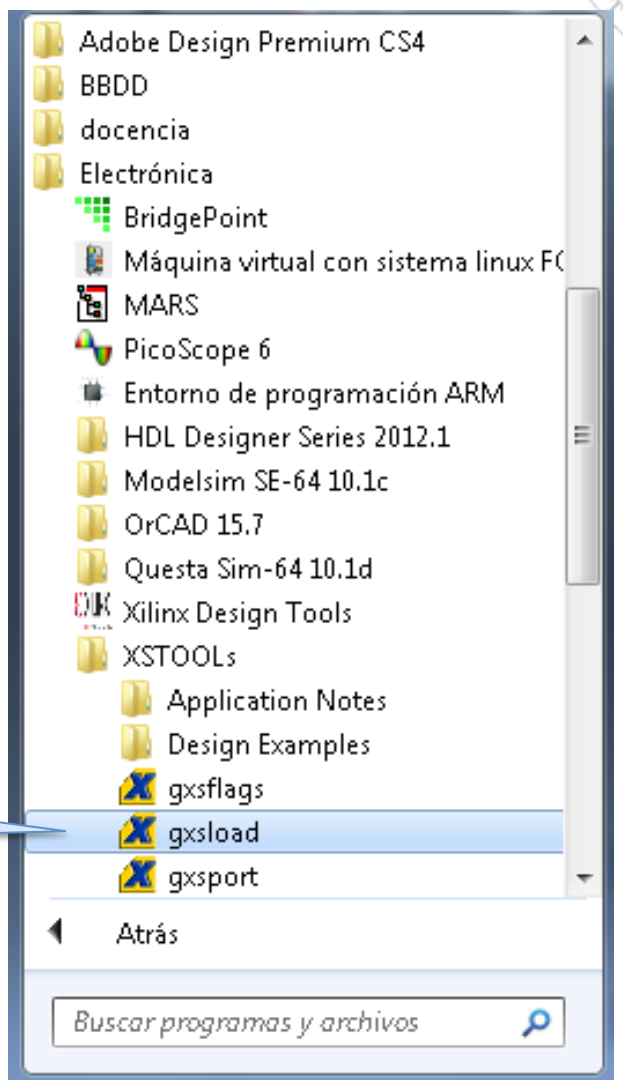




Descarga .bit sobre FPGA

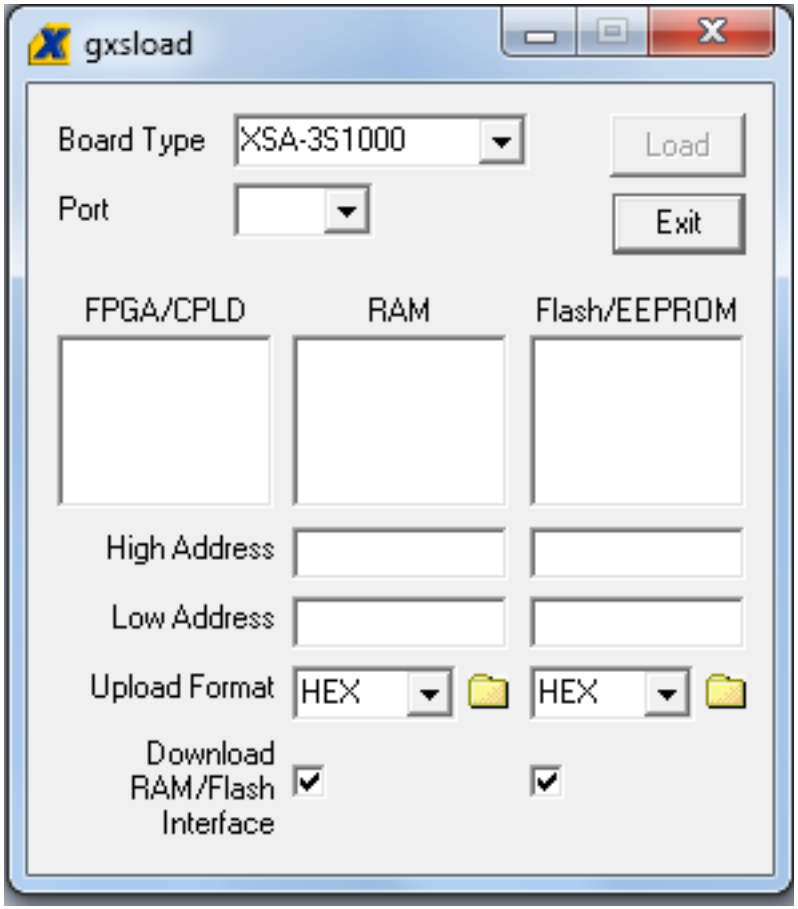
- Comprobar jumper J9 de la FPGA en la posición XS, de no ser así solicitad otra FPGA
- Para poder volcar el fichero.bit a la FPGA, abrir el programa gxsload, que se encuentra en “Inicio->Electrónica->XSTools”

El programa es accesible a través de esta ruta

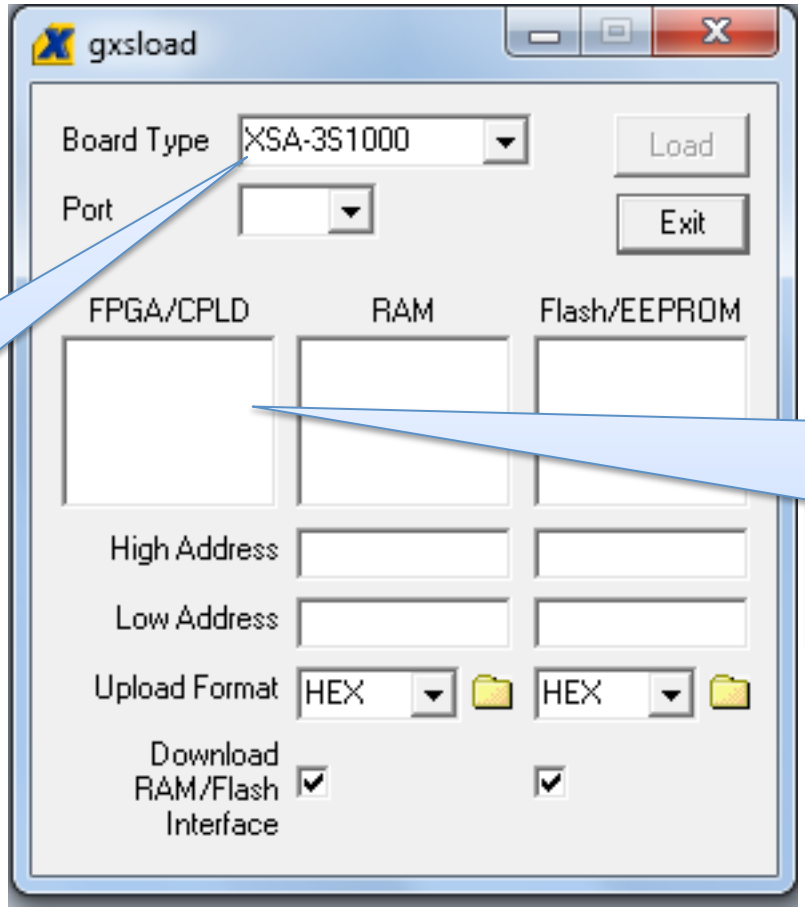


Descarga .bit sobre FPGA

- Se abrirá la ventana de gxsload



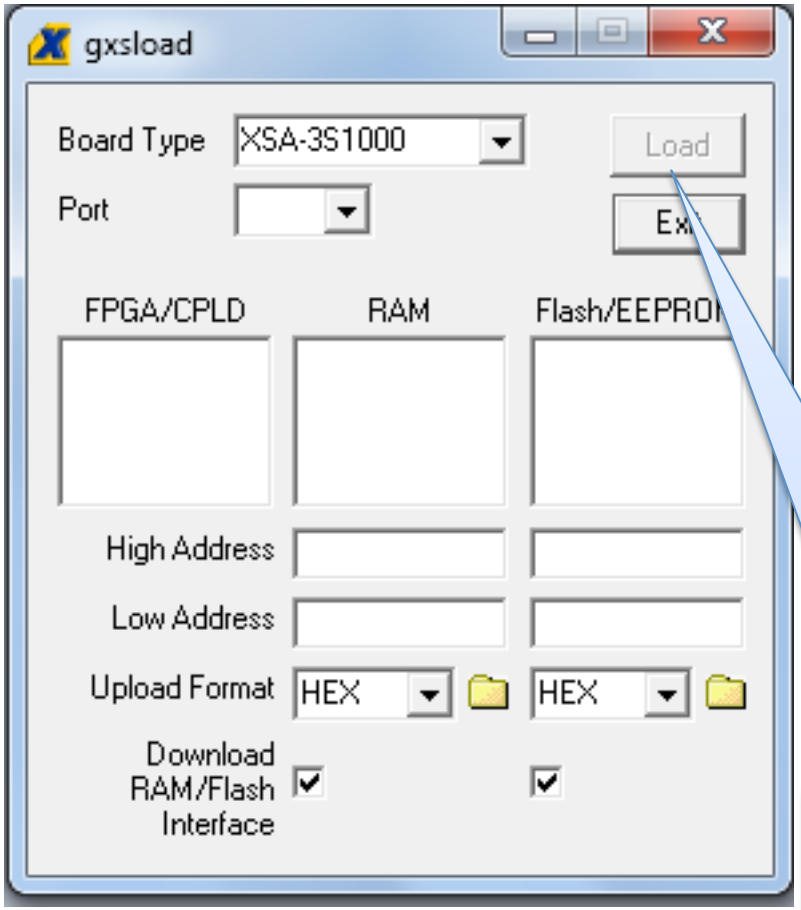
Descarga .bit sobre FPGA



Aquí seleccionamos nuestro modelo de FPGA

En esta ventana arrastramos y soltamos el fichero .bit que acabamos de crear

Descarga .bit sobre FPGA



A continuación, pulsamos el botón "Load" y la carga comienza

Descarga .bit sobre FPGA

- 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

- Simular e implementar un registro de entrada/salida paralela

Práctica 1.b

```
----------------------------------------------------------------
-- registro de desplazamiento entrada-salida paralelo
----------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity reg_paralelo is
    port (rst, clk, 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
```

Fichero p1b.vhd

Práctica 1.b

```
Begin

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

Implementación Práctica 1.b

- Pin del reloj:

```
NET clk LOC=T9;
```

- Hasta que no se resetee el circuito no funcionará correctamente

Anotaciones

Calificación

- Seguiréis las instrucciones de las transparencias para simular e implementar los diseños de la práctica 1.a y 1.b.
- Debéis enseñar al profesor del laboratorio cada una de las partes funcionando sobre la FPGA (+0.2).
- Habrá un apartado OPCIONAL, que os permitirá sumar otro +0.2.

Calificación

- 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