



Práctica 1. Introducción

toc

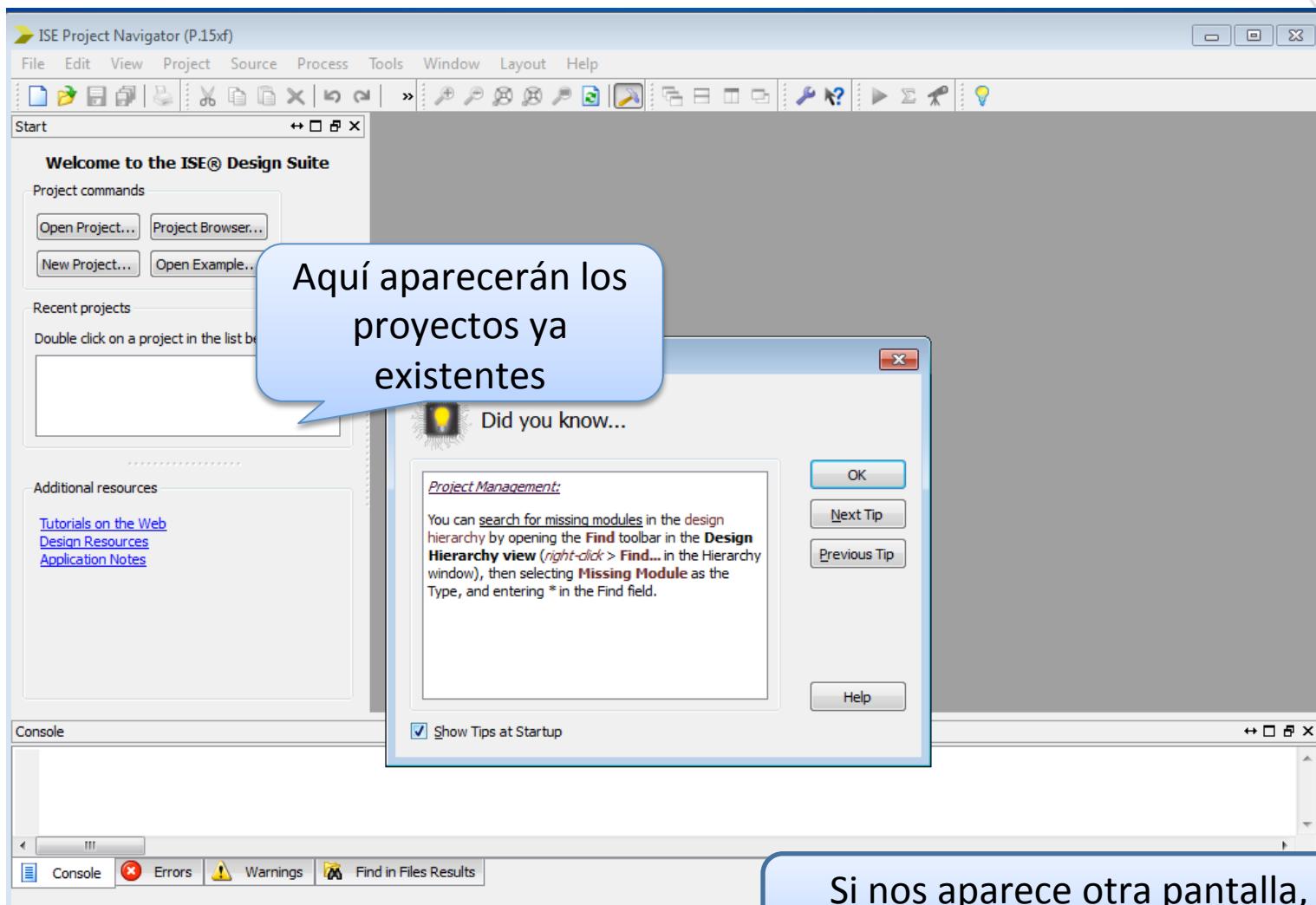


Objetivos

- Aprender a utilizar la herramienta Xilinx ISE 2014.1 para diseñar circuitos digitales.
- Diseño, simulación e implementación de circuitos sencillos.



Pantalla de bienvenida

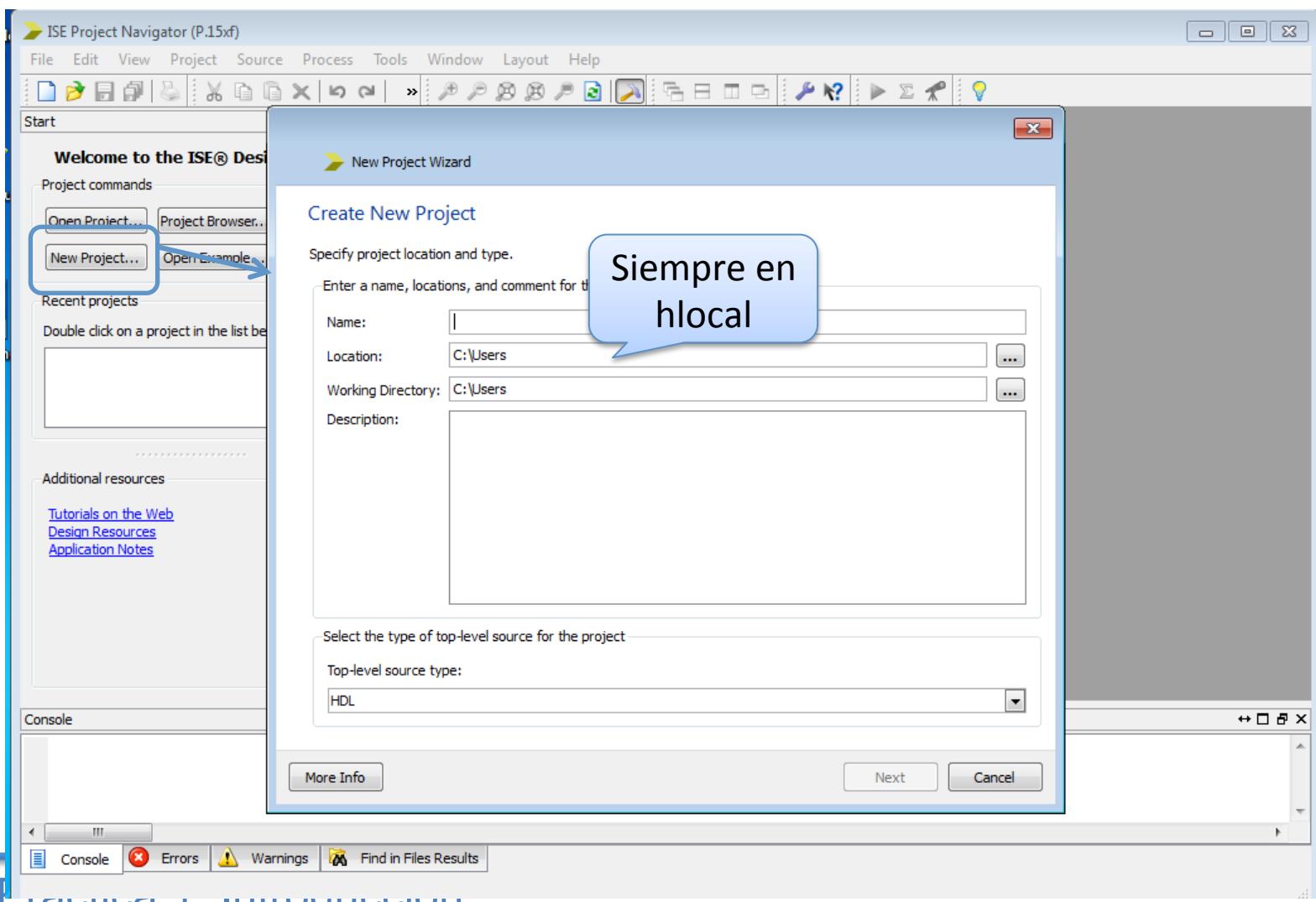


Aquí aparecerán los
proyectos ya
existentes

Si nos aparece otra pantalla,
ir a transparencia 12

Nombre y localización

- Desde la ventana de bienvenida



Configuración del proyecto



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
Family	Spartan3
Device	XC3S1000
Package	FT256
Speed	-5
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Pro	
Manual Compile Order	
VHDL Source Analysis Stand	
Enable Message Filtering	

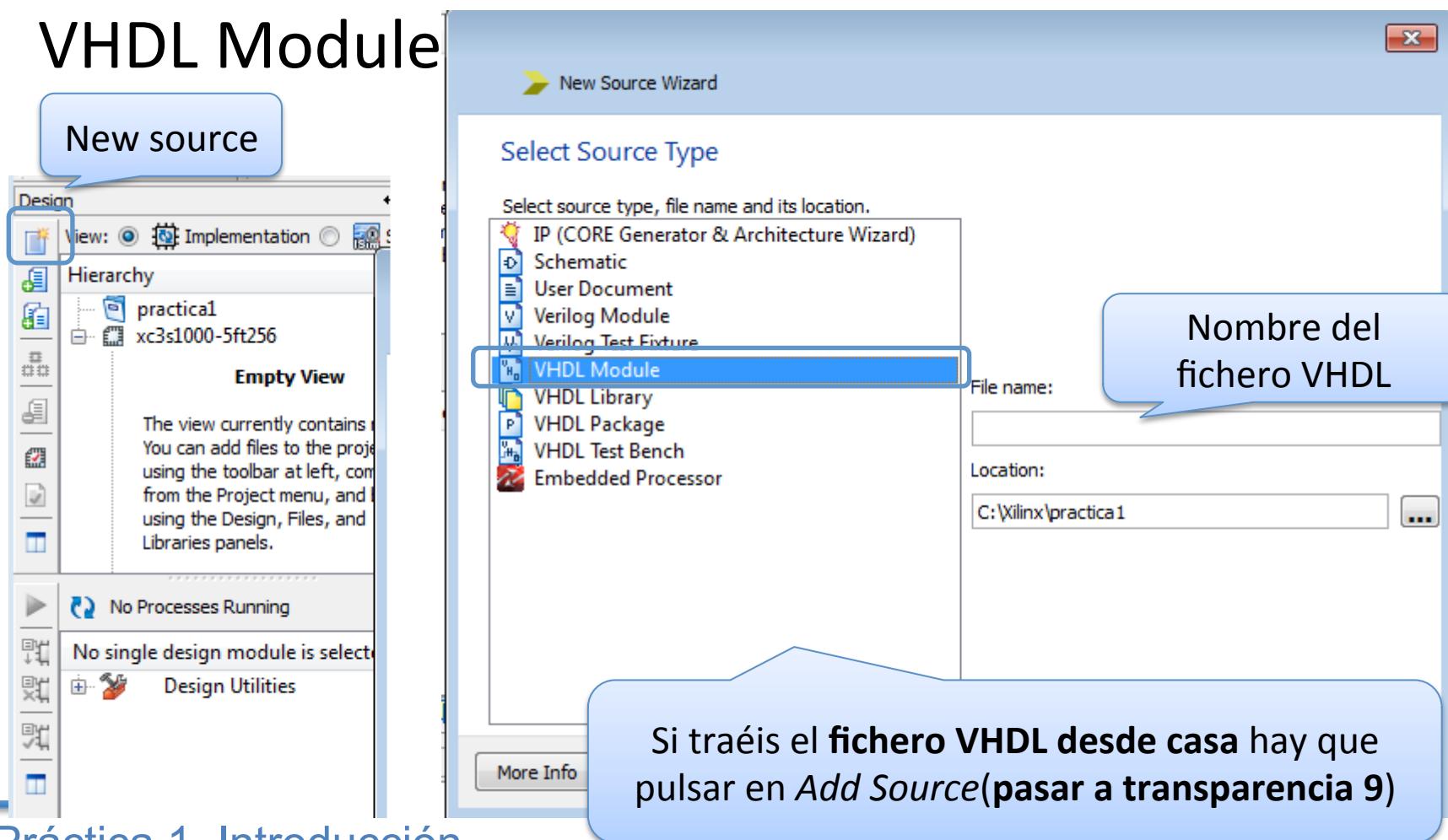
More Info Next Cancel

Seleccionar el entorno de simulación ISim

A blue callout bubble points to the "Simulator" row, which is highlighted with a blue border. A large blue arrow points from the "Simulator" row to a light blue rounded rectangle containing the selected values: Spartan 3, XC3S1000, FT256, and -5.

Crear un fichero VHDL (I)

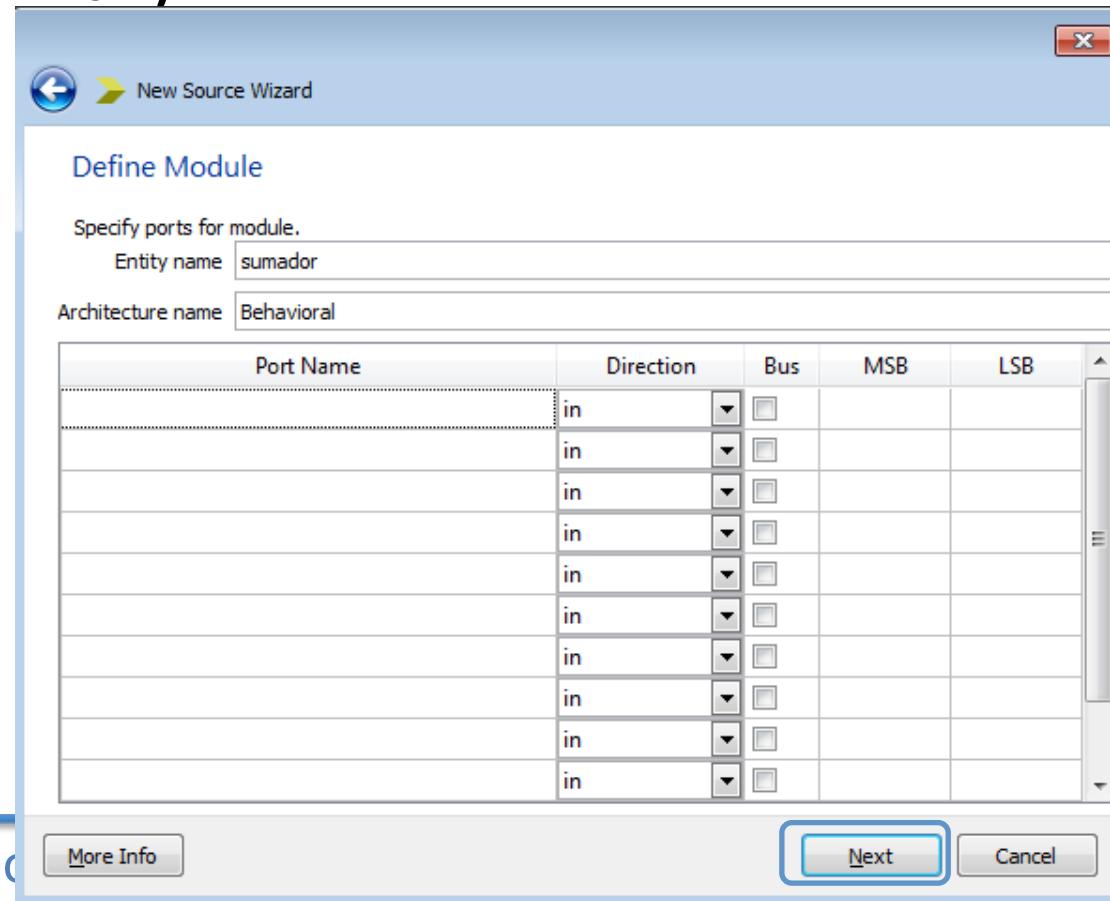
- Siempre se inicia creando un fichero tipo VHDL Module



Crear un fichero VHDL (II)



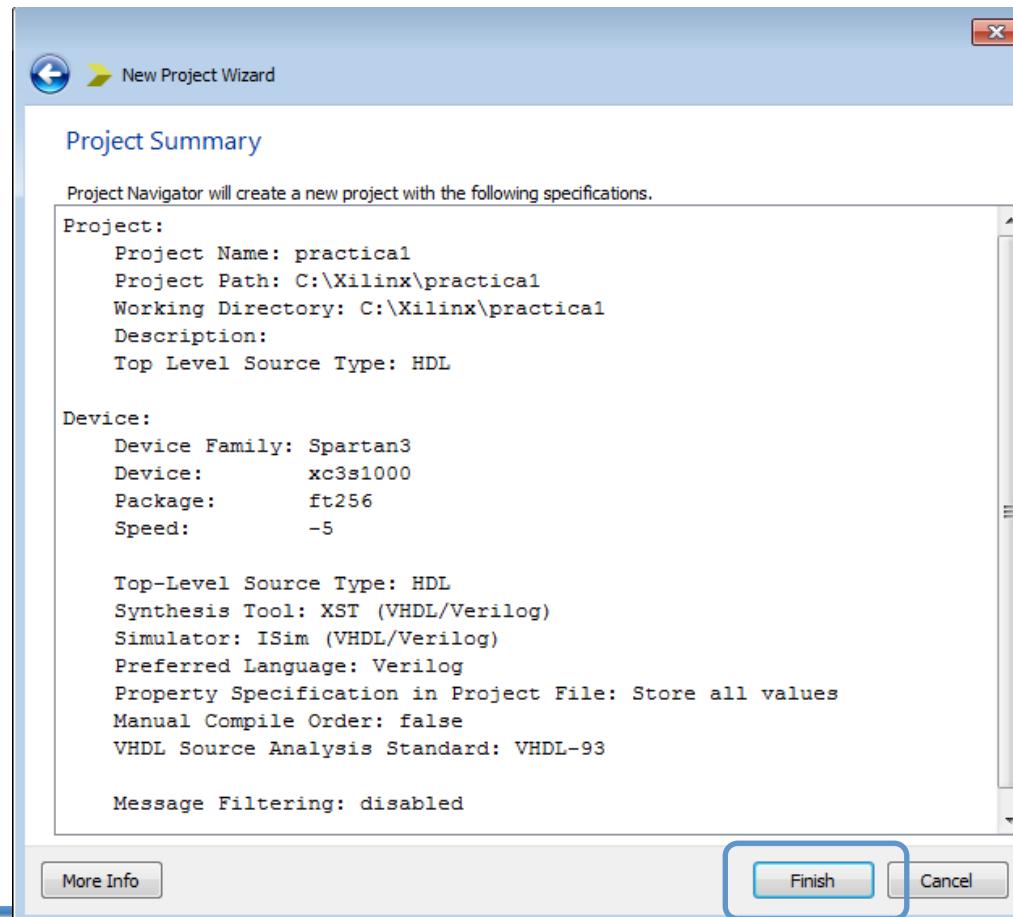
- Wizard para definir las señales de entrada y salida (no utilizar)



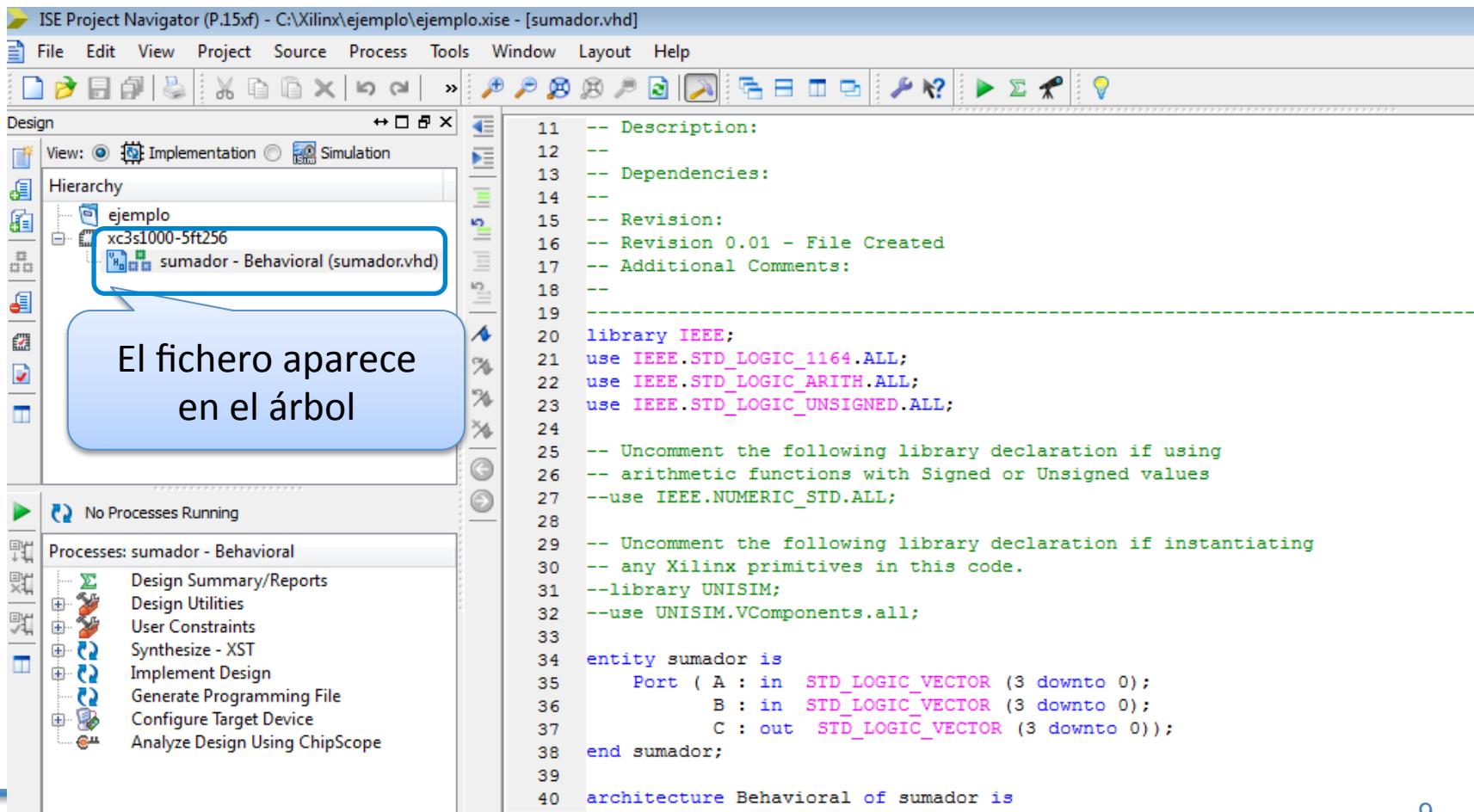
Crear un fichero VHDL (III)



- Todo ha salido correcto



■ Todo ha salido correcto



The screenshot shows the ISE Project Navigator interface. The title bar reads "ISE Project Navigator (P.15xf) - C:\Xilinx\ejemplo\ejemplo.xise - [sumador.vhd]". The menu bar includes File, Edit, View, Project, Source, Process, Tools, Window, Layout, and Help. The toolbar has various icons for file operations. The left pane shows a "Design" view with "Implementation" selected. A "Hierarchy" tree displays a project named "ejemplo" containing a device "xc3s1000-5ft256" and a behavioral entity "sumador - Behavioral (sumador.vhd)". A callout bubble points to this entity with the text "El fichero aparece en el árbol". The right pane is a code editor showing the VHDL source code for "sumador.vhd". The code includes comments about dependencies, revision, and library declarations, followed by the entity and architecture definitions.

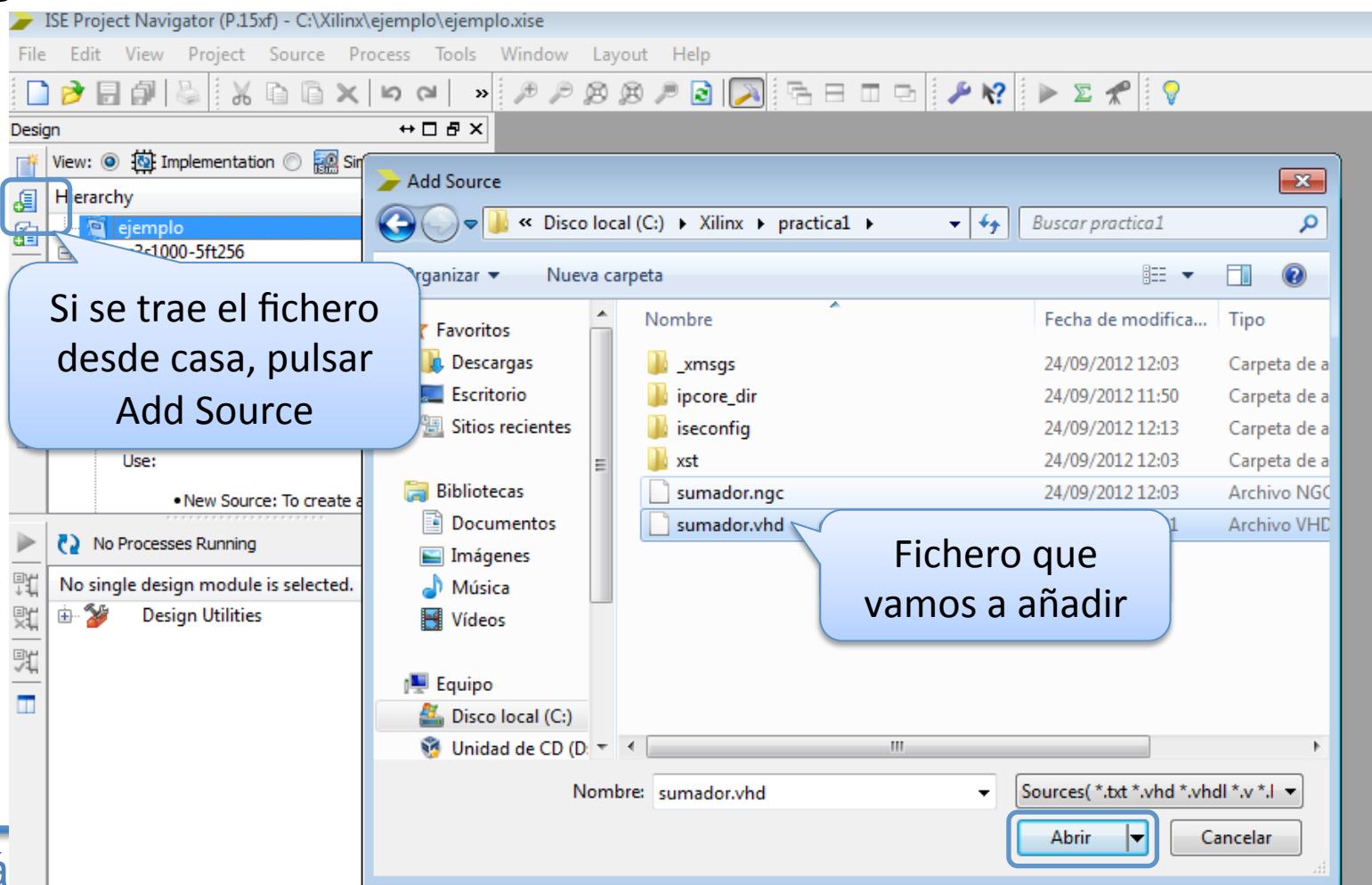
```
11 -- Description:  
12 --  
13 -- Dependencies:  
14 --  
15 -- Revision:  
16 -- Revision 0.01 - File Created  
17 -- Additional Comments:  
18 --  
19  
20 library IEEE;  
21 use IEEE.STD_LOGIC_1164.ALL;  
22 use IEEE.STD_LOGIC_ARITH.ALL;  
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;  
24  
25 -- Uncomment 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
```



Añadir ficheros VHDL (I)



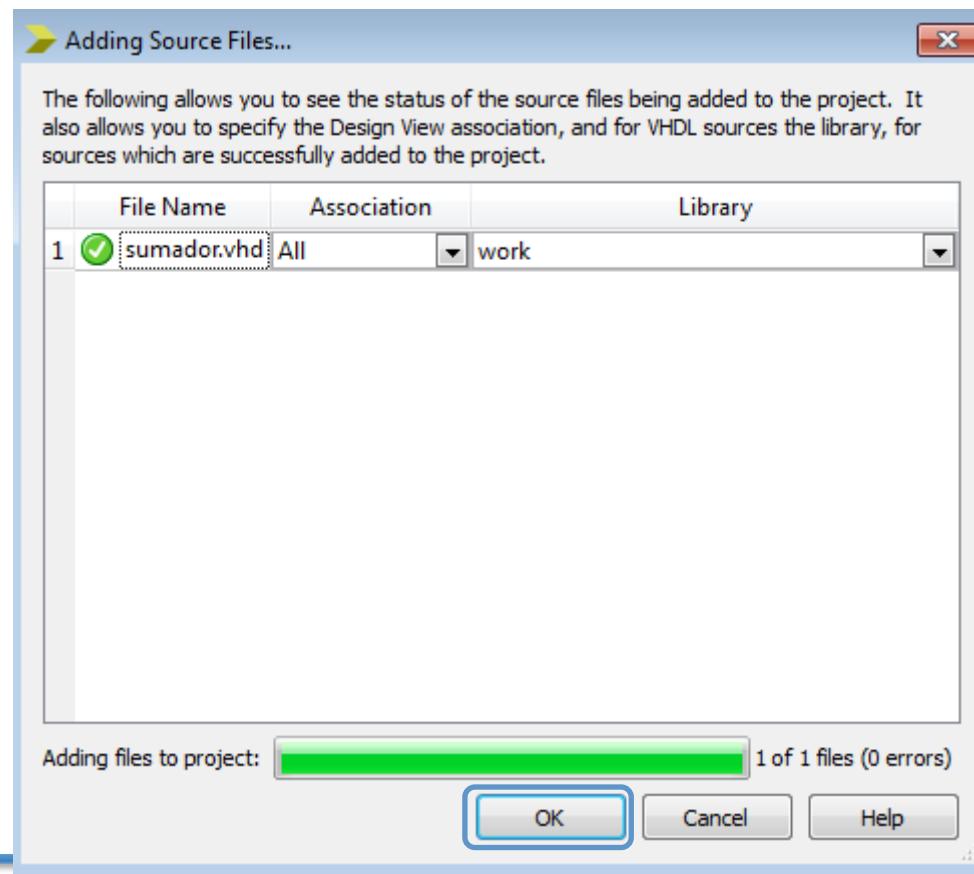
- Los ficheros a añadir deben ser previamente grabados en el directorio hlocal



Añadir ficheros VHDL (II)



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



Añadir ficheros VHDL (III)



- El fichero se ha añadido correctamente

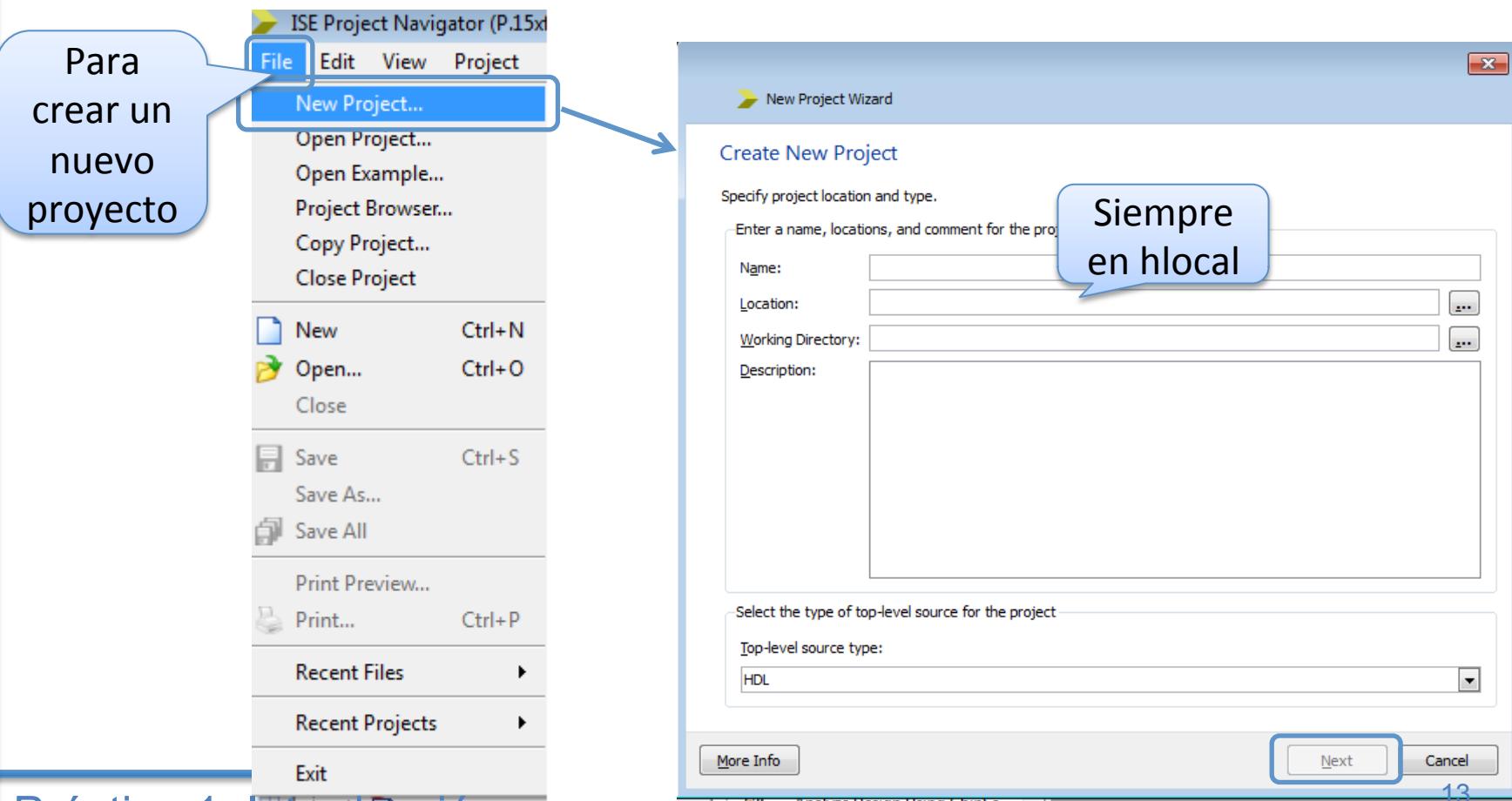
The screenshot shows the ISE Project Navigator interface. In the top left, the title bar reads "ISE Project Navigator (P.15xf) - C:\Xilinx\ejemplo\ejemplo.xise - [sumador.vhd]". The menu bar includes File, Edit, View, Project, Source, Process, Tools, Window, Layout, and Help. The toolbar has various icons for file operations. The "Design" tab is selected in the top-left panel, which shows a "Hierarchy" tree. The tree has a root node "ejemplo" with a child "xc3s1000-5ft256", and under that, a "sumador - Behavioral (sumador.vhd)" file is highlighted with a blue border. A callout bubble from this highlighted item contains the text "El fichero se ha incorporado al proyecto". The bottom-left panel shows a list of processes: "No Processes Running" and a list for "Processes: sumador - Behavioral" which includes "Design Summary/Reports", "Design Utilities", "User Constraints", "Synthesize - XST", "Implement Design", "Generate Programming File", "Configure Target Device", and "Analyze Design Using ChipScope". The main right-hand pane displays the VHDL code for "sumador.vhd".

```
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
```



Crear un proyecto

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



2. Gestión de un proyecto

Pantalla de desarrollo



The screenshot shows the Xilinx ISE Project Navigator interface. The main window displays a VHDL code for a sumador (adder). A callout bubble points to the code area with the text: "Aquí se escribe el código VHDL que nos interese".

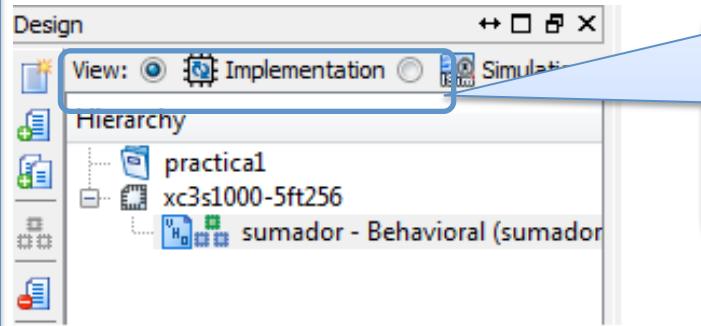
```
12 -- Dependencies:
13 -- Revision:
14 -- Revision 0.01 - File Created
15 -- Additional Comments:
16 --
17 -----
18
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
21
22 -- Uncomment the following library declaration if using
23 -- arithmetic functions with Signed or Unsigned values
24 --use IEEE.NUMERIC_STD.ALL;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

Sintetizar el diseño

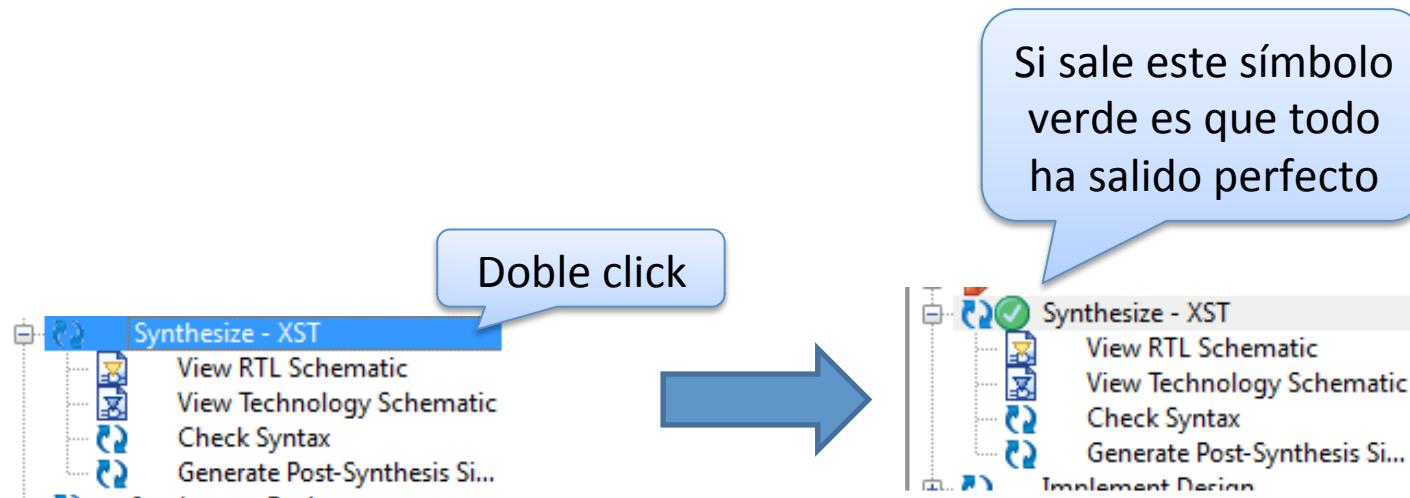


- Proceso para convertir el modelo del circuito en un *netlist*.

¿Cómo 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

¿Por qué simular?

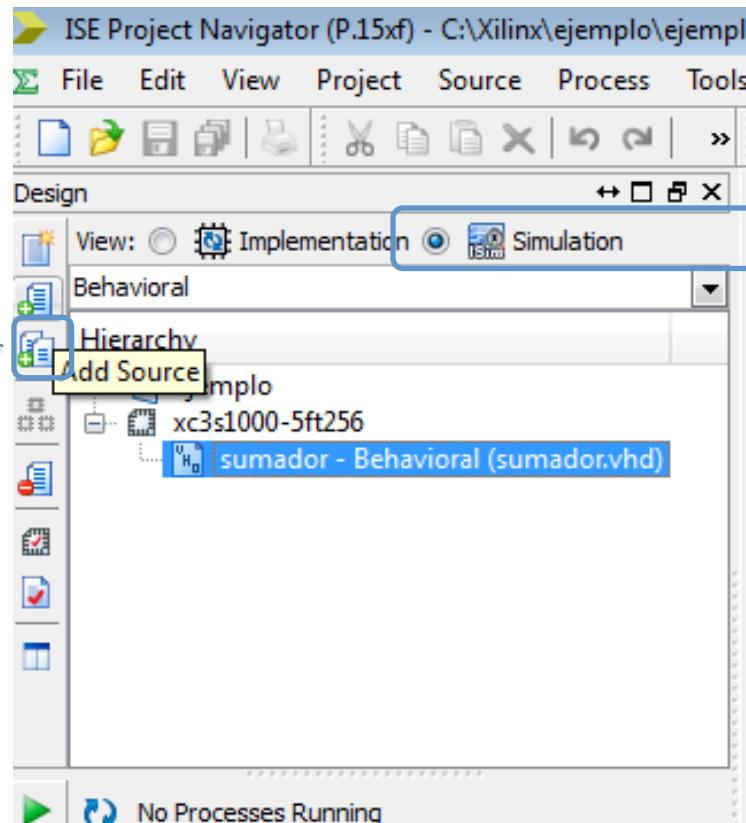


- Hay que verificar que el diseño es correcto.
- Para ello hay que simular su comportamiento:
 - Crear un *testbench* que genere las entradas del circuito.
 - Instanciar el circuito a verificar (*Device Under Test, DUT*) en el testbench
 - Simular el comportamiento del *testbench*

Añadir test (I)



- Añadir el fichero de simulación simsum.vhd



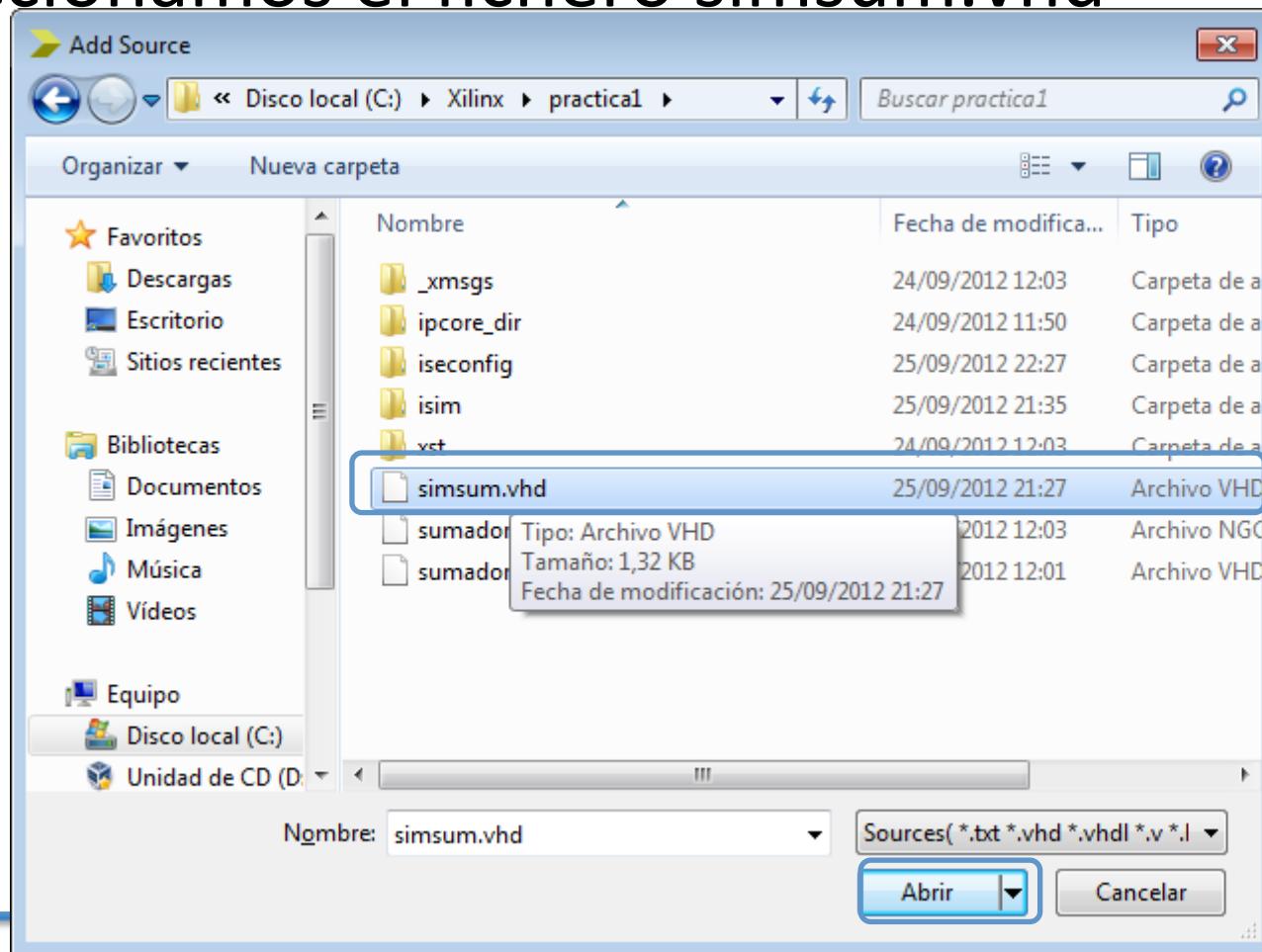
Marcamos la opción Simulation

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

Añadir test (II)



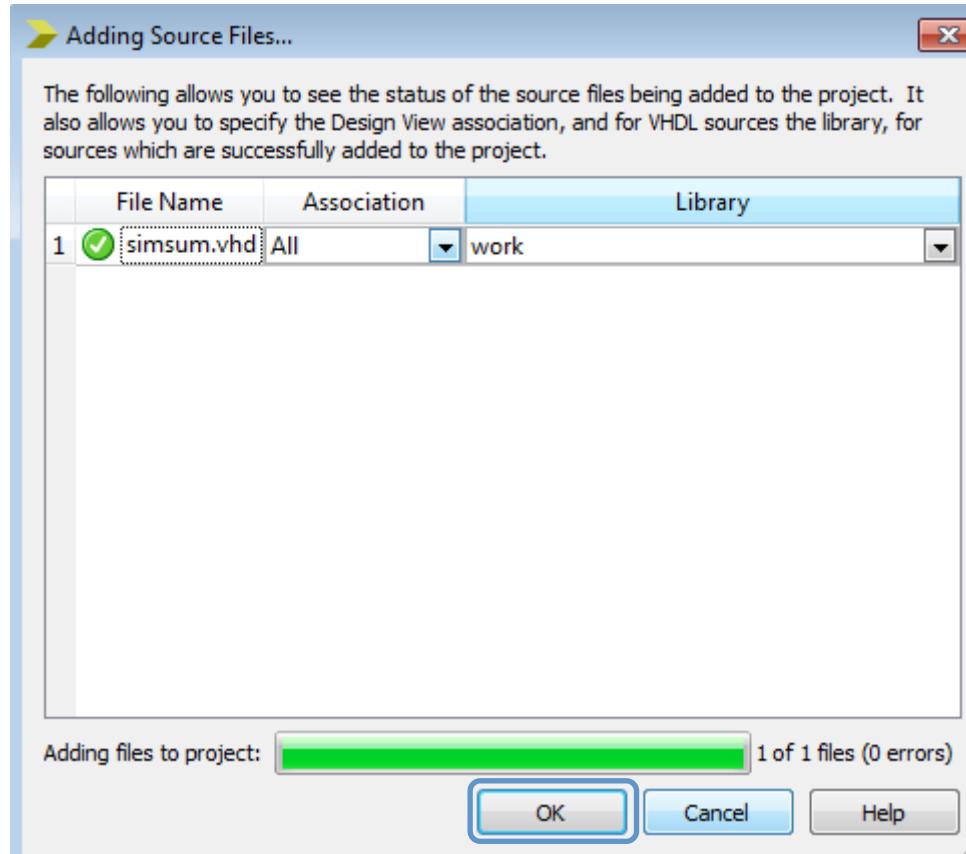
- Seleccionamos el fichero simsum.vhd



Añadir test (III)



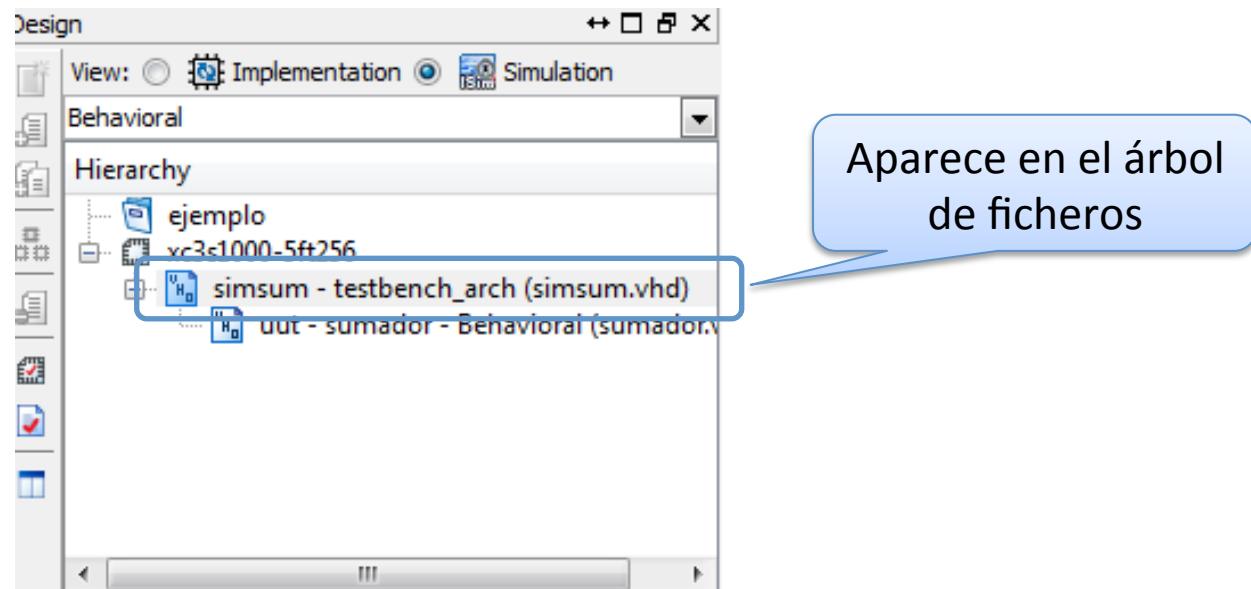
- El fichero se está añadiendo al proyecto



Añadir test (IV)



- El fichero se ha añadido correctamente



Añadir test (V)

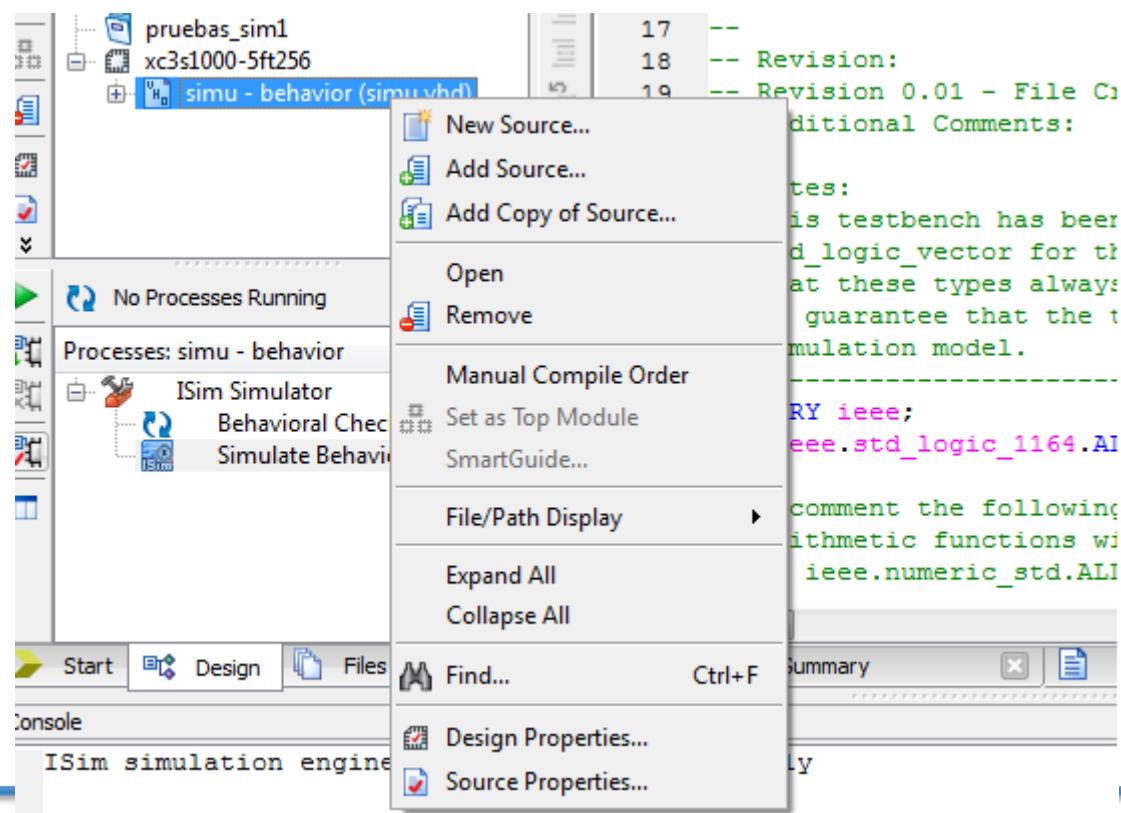


- Si el fichero de simulación se añade al proyecto no lo reconoce como fichero de simulación
- Asegurarse antes de hacer la implementación que se trata de un fichero de simulación



Añadir test (VI)

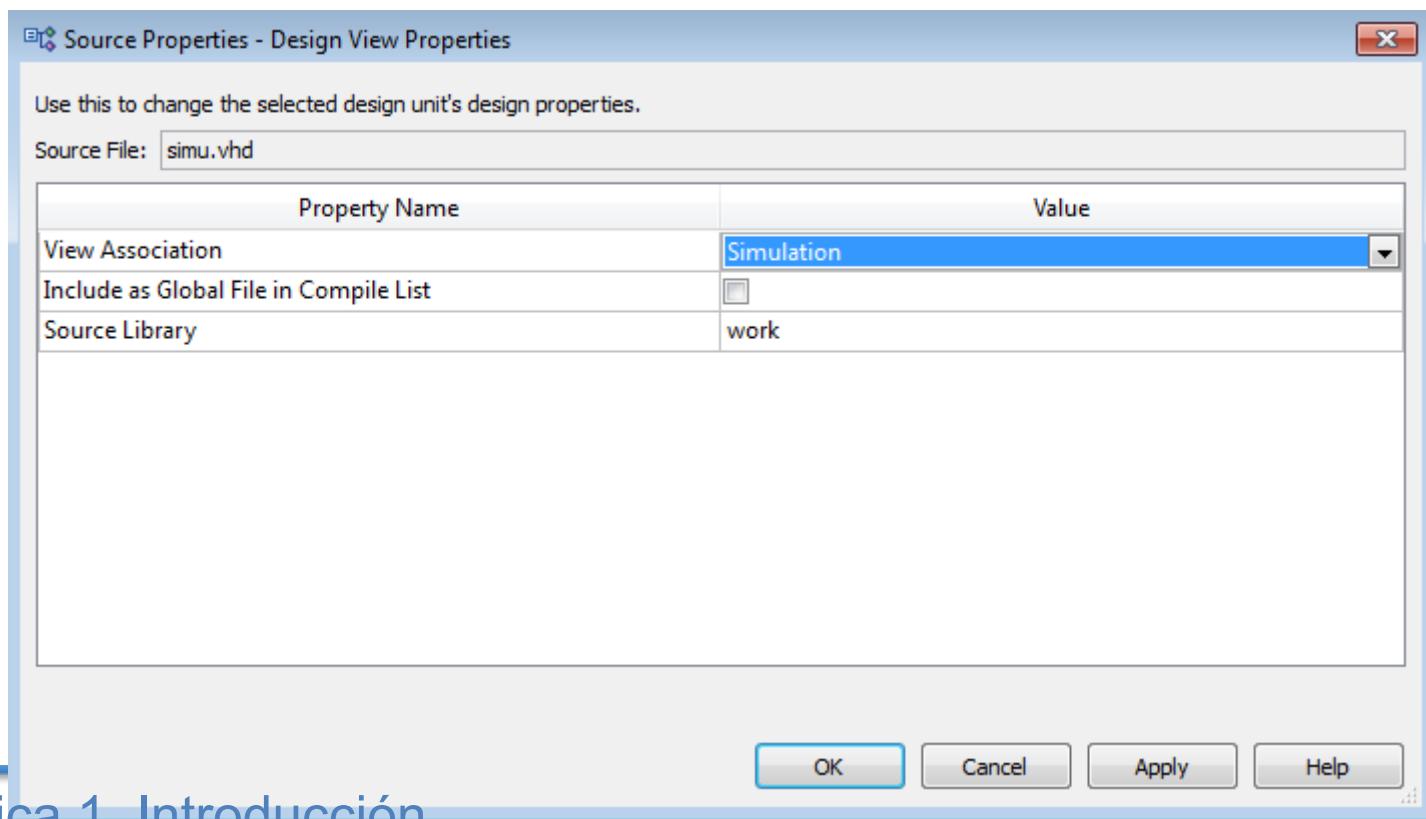
- Pulsar con el botón derecho sobre dicho fichero y elegir la opción *Source properties*



Añadir test (VII)



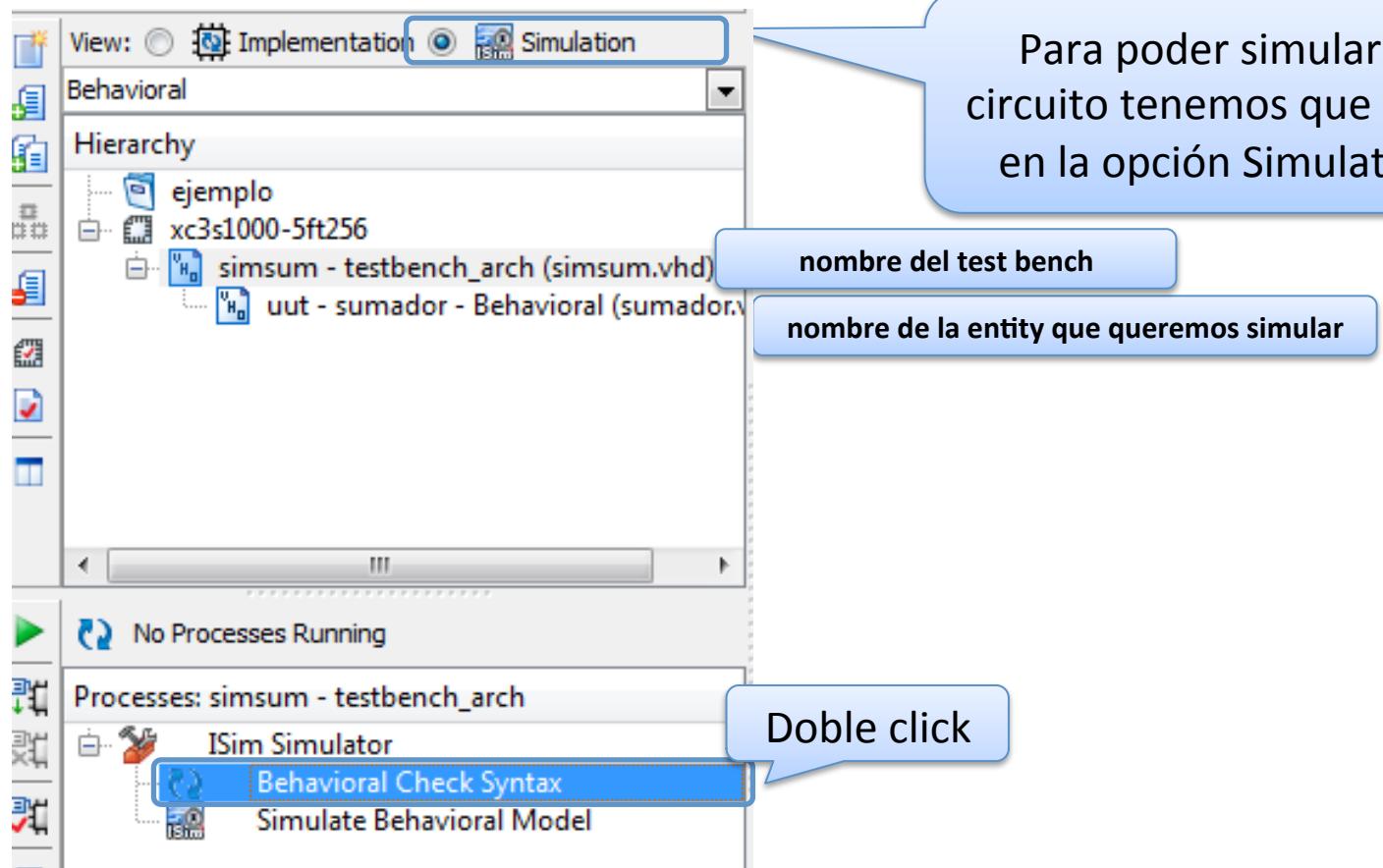
- En *View Association* aparecerá de tipo *All*, elegir tipo *Simulation*



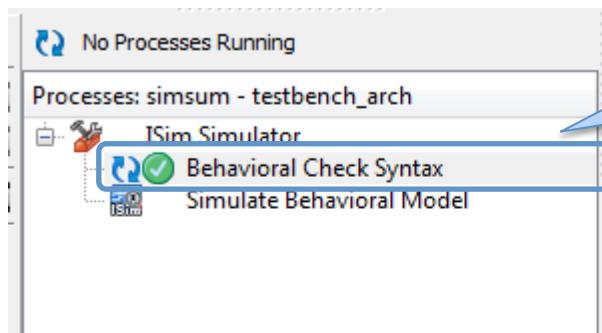
Simulación (I)



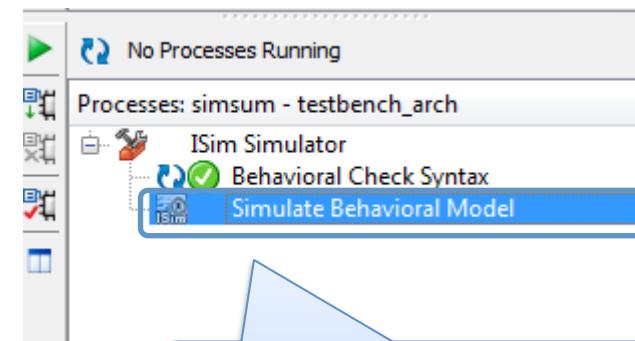
- Desplegar el menú de simulación



Simulación (II)



La simulación se ha ejecutado correctamente



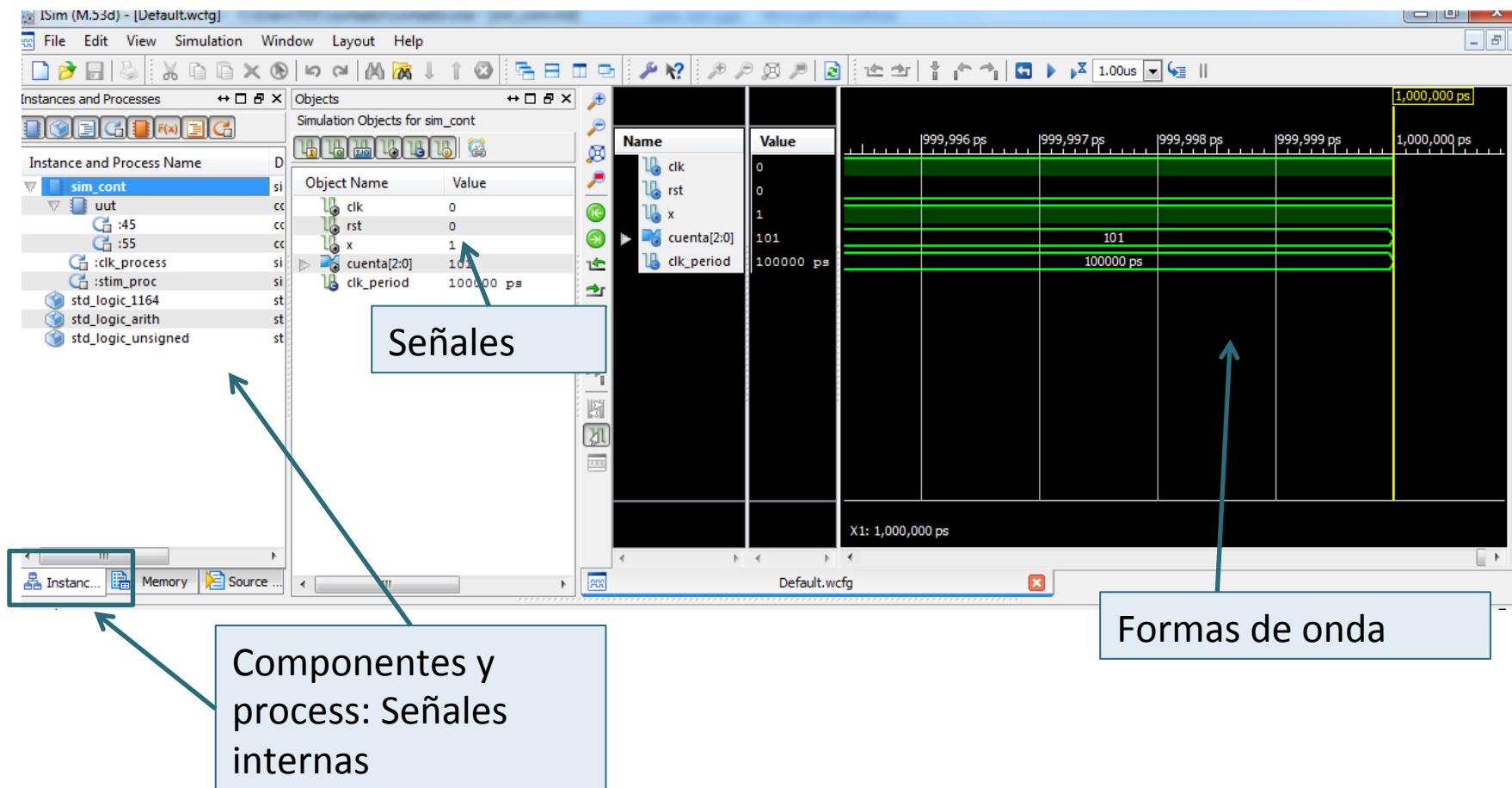
Necesitamos arrancar el ISIM para ver las formas de onda y comprobar que el circuito está bien diseñado

4. Simulación



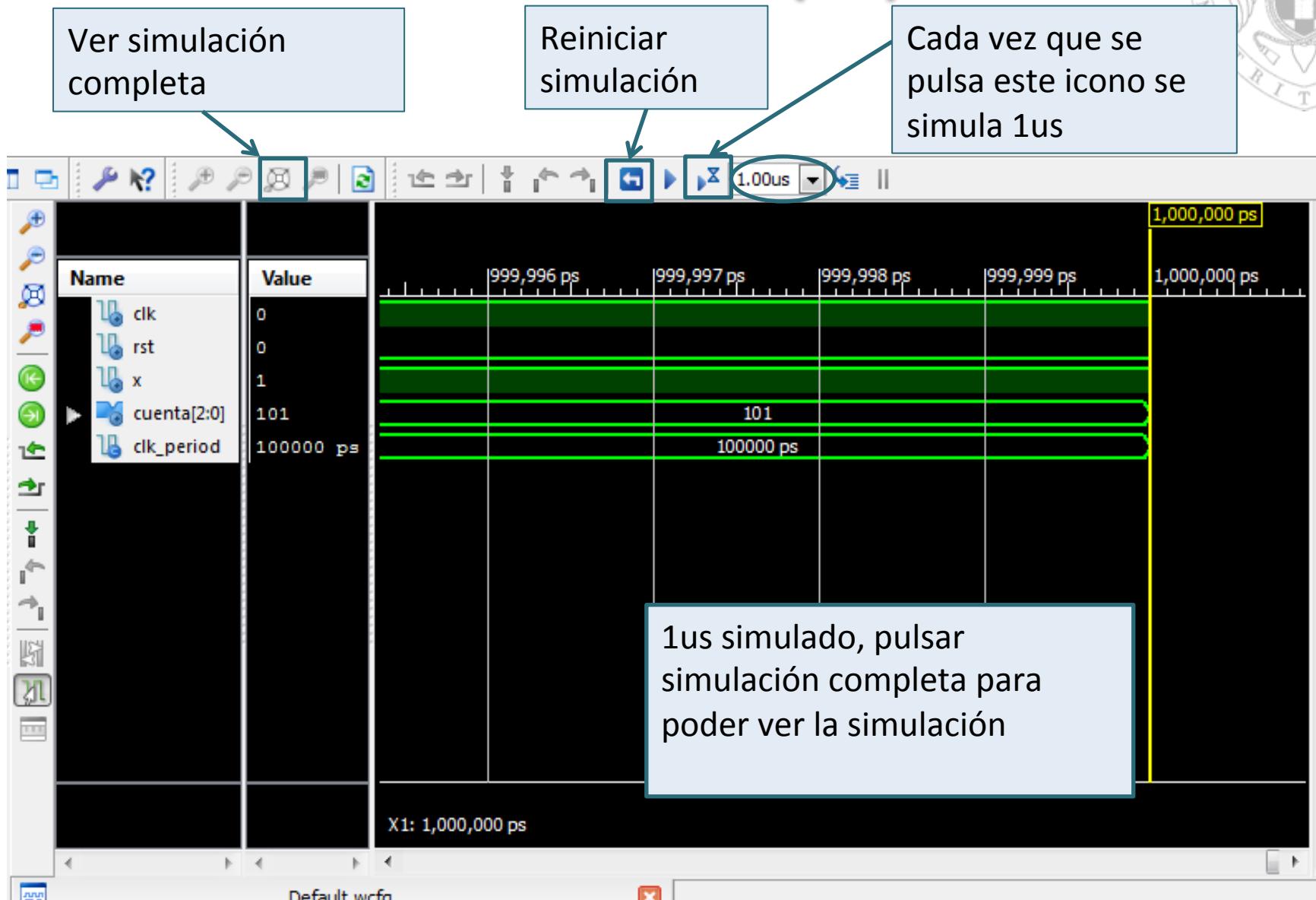
Simulación (III)

■ Ventana simulador ISim



toc

Simulación (IV)

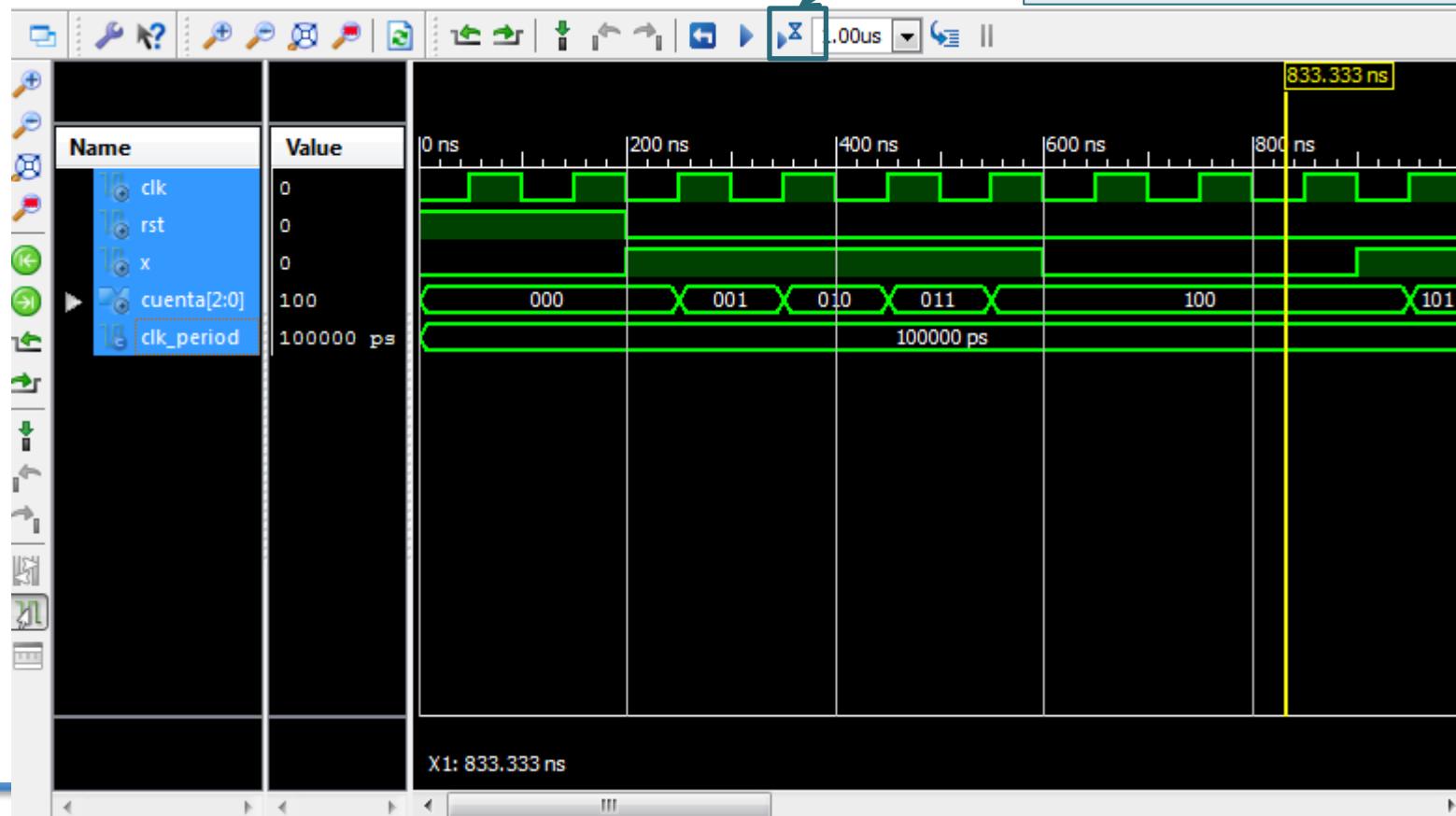


4. Simulación

Simulación (V)



Si queremos simular
más tiempo pulsar en este icono



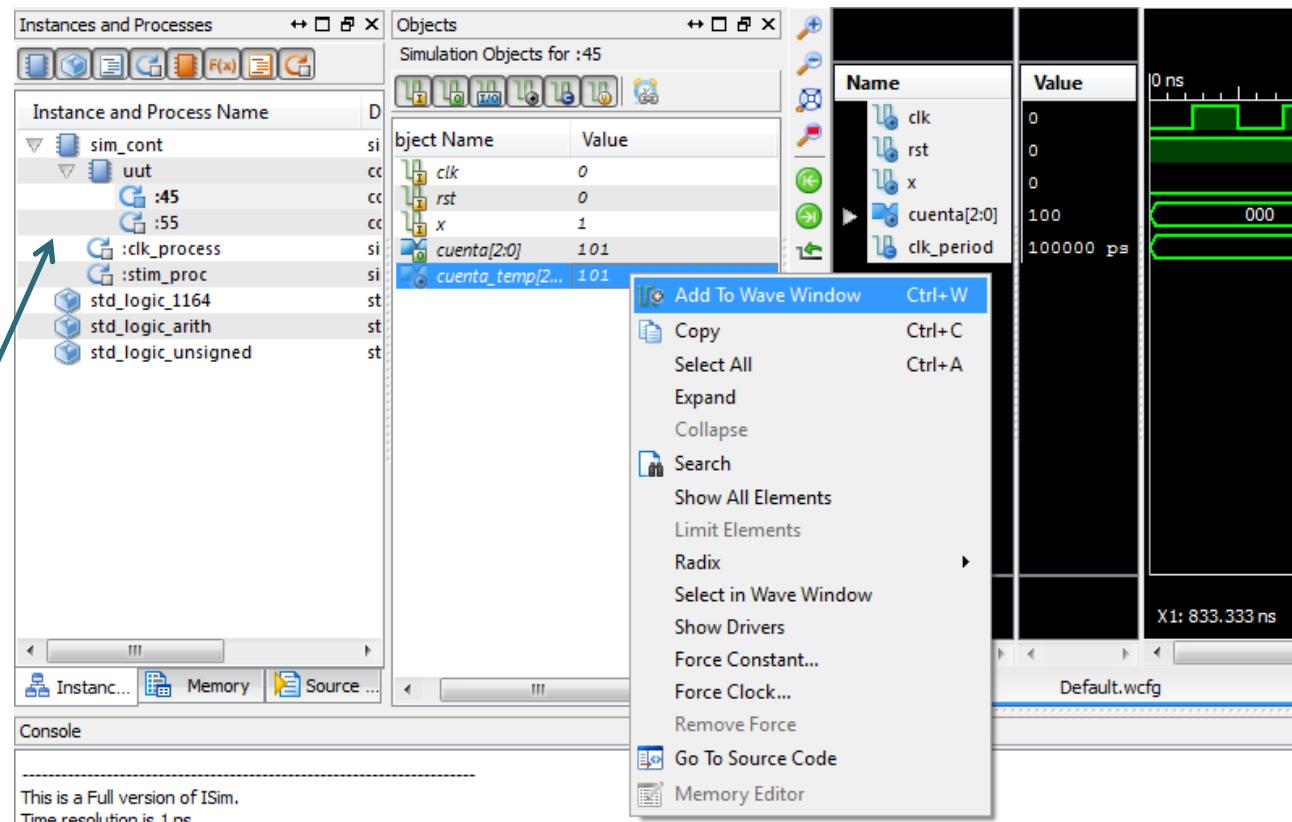
toc

Práctica 1. Introducción

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 componentes



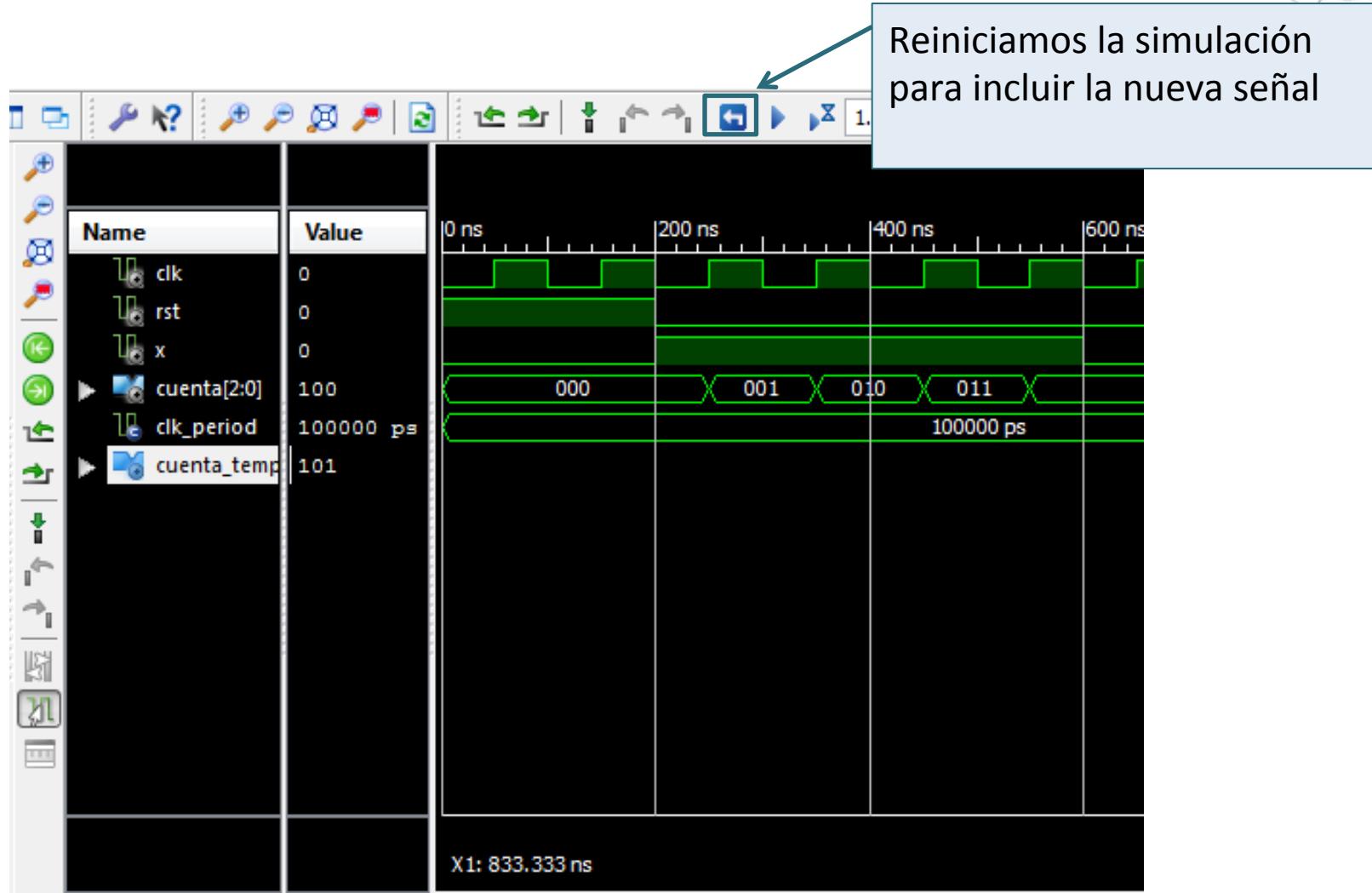
4. Simulación

toc

Simulación (VII)



- La señal se añadirá a la ventana de simulación



Implementación

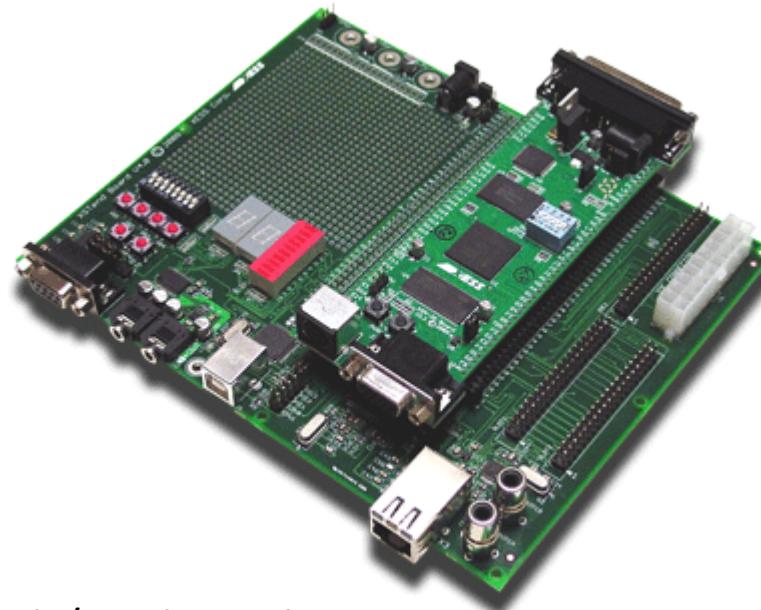


- Podemos comprobar que el circuito funciona perfectamente en el *mundo real*
 - El circuito se va a implementar sobre una FPGA
 - Hay que indicar de dónde se leen las entradas (switches) y en dónde se escriben las salidas (LEDs o display de 7 segmentos)
 - Hay que añadir un fichero UCF donde se le indica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

Plataforma



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



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



Definición pines E/S

■ 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
```

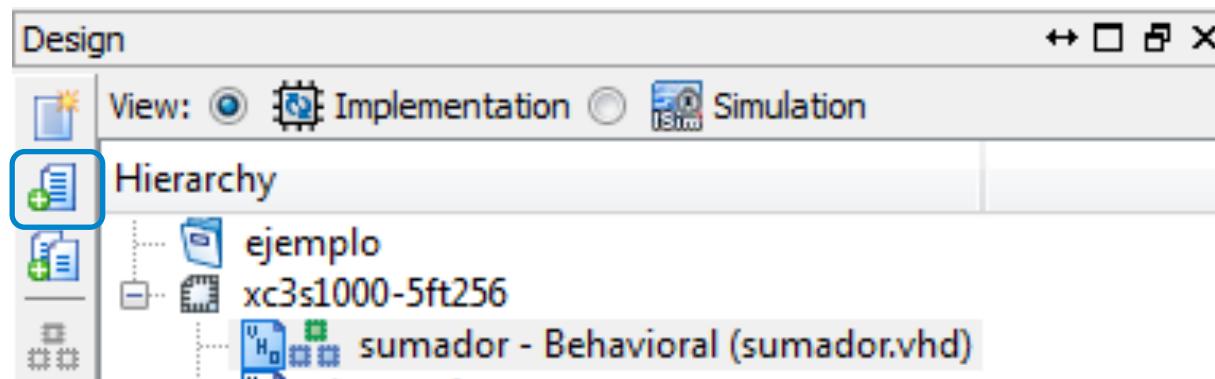
```
NET C<0> LOC=L5;  
NET C<1> LOC=N2;  
NET C<2> LOC=M3;  
NET C<3> LOC=N1;
```

- 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

Definición pines E/S



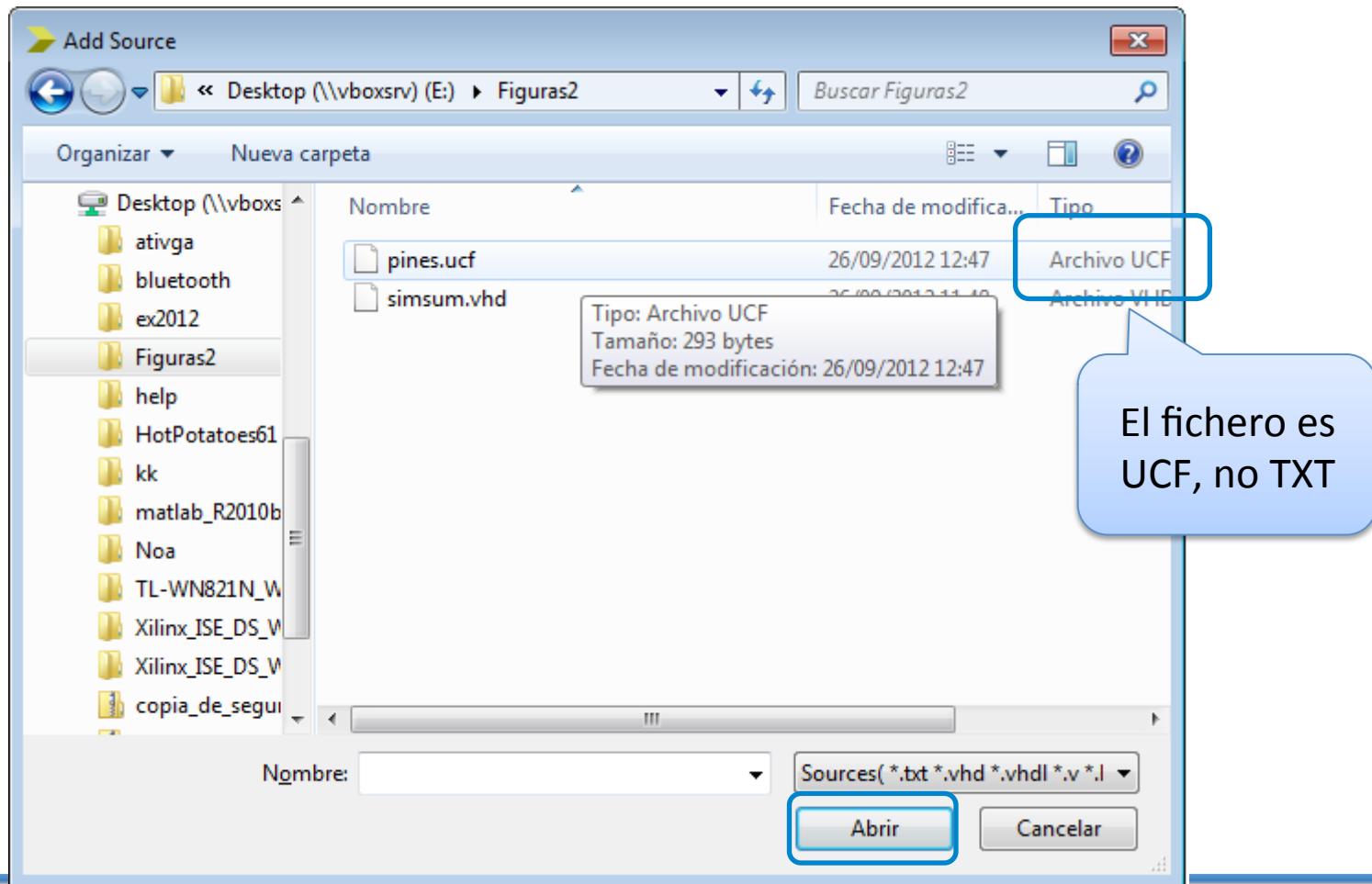
- El fichero pines.ucf se añade al proyecto mediante Add Source





Definición pines E/S

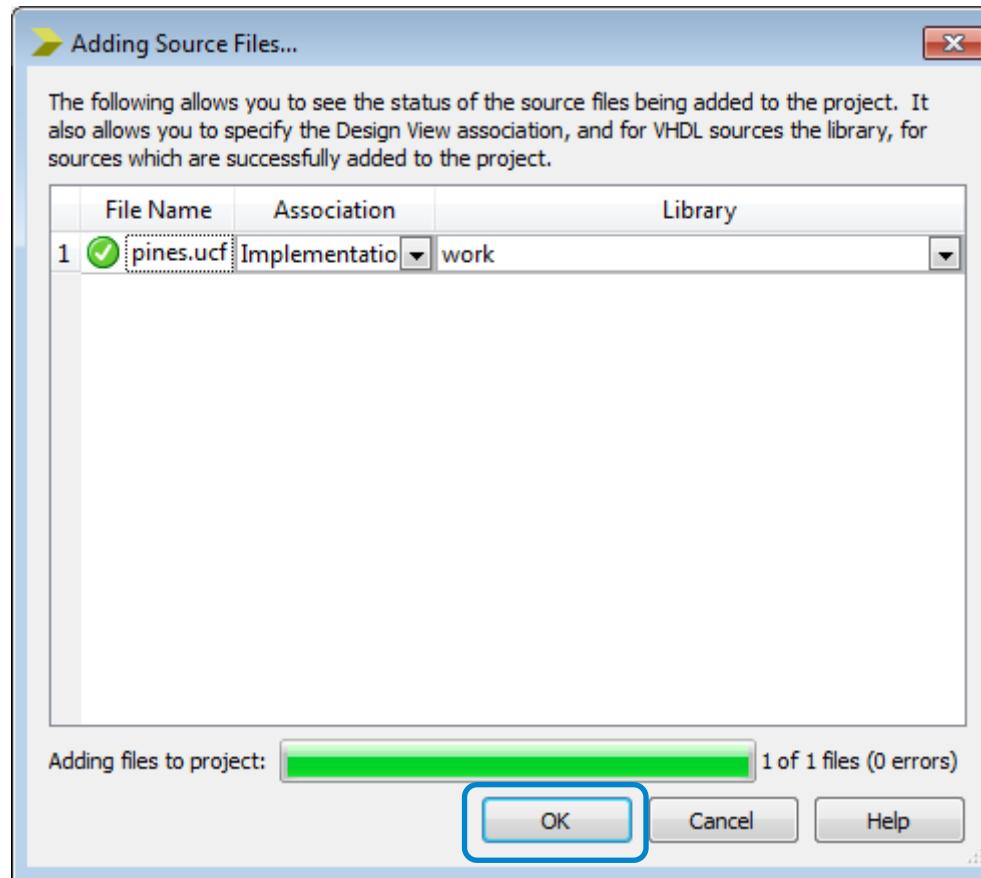
- Seleccionamos el fichero pines.ucf



Definición pines E/S



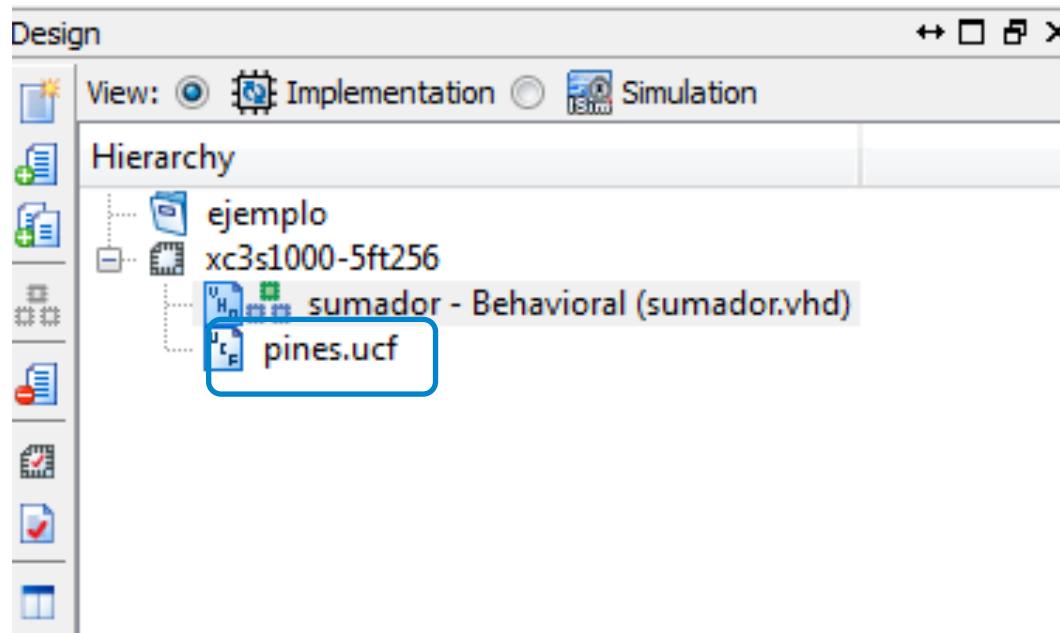
- El fichero se añade al proyecto



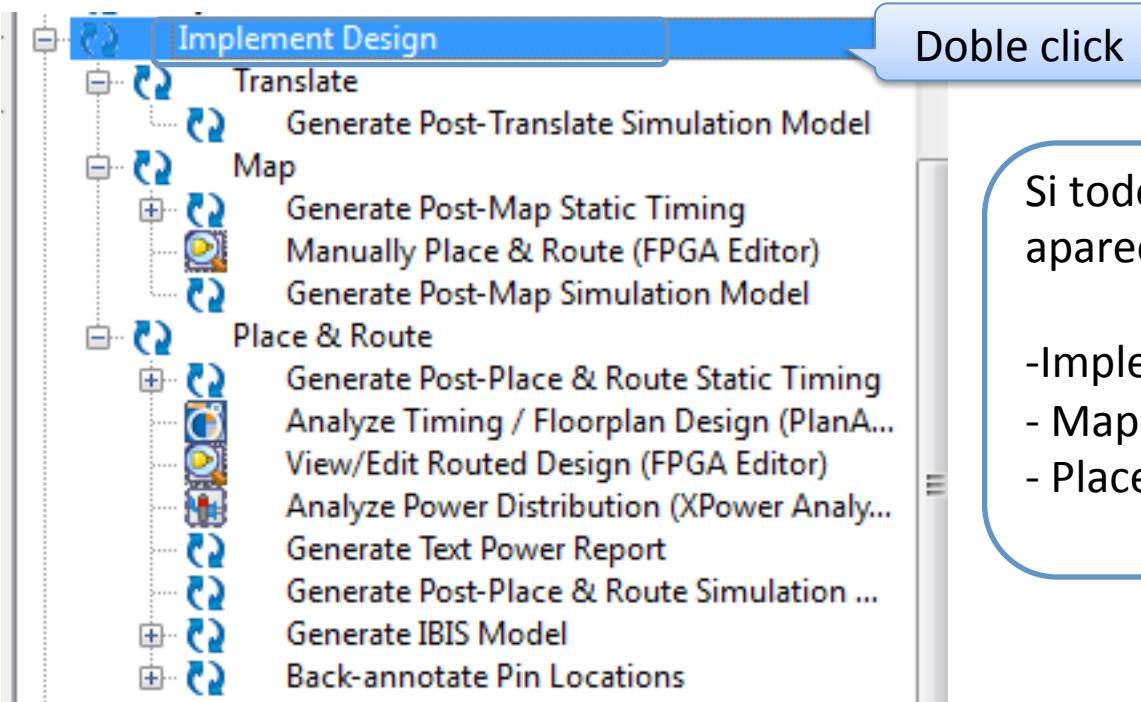
Definición pines E/S



- El fichero aparece en el árbol de jerarquía



Implementar el diseño (I)



Doble click

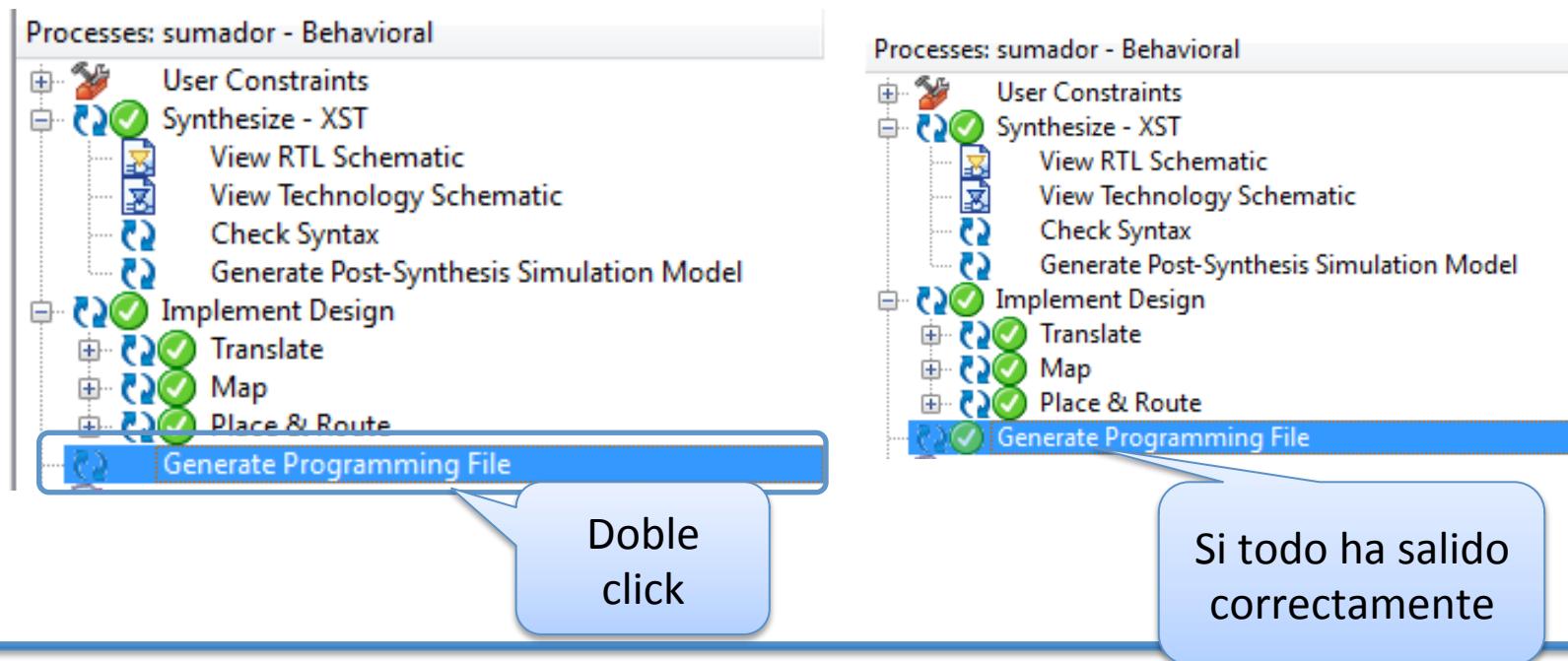
Si todo está correcto
aparecerá en:

- Implementation Design
- Map
- Place & Route

Implementar el diseño (II)



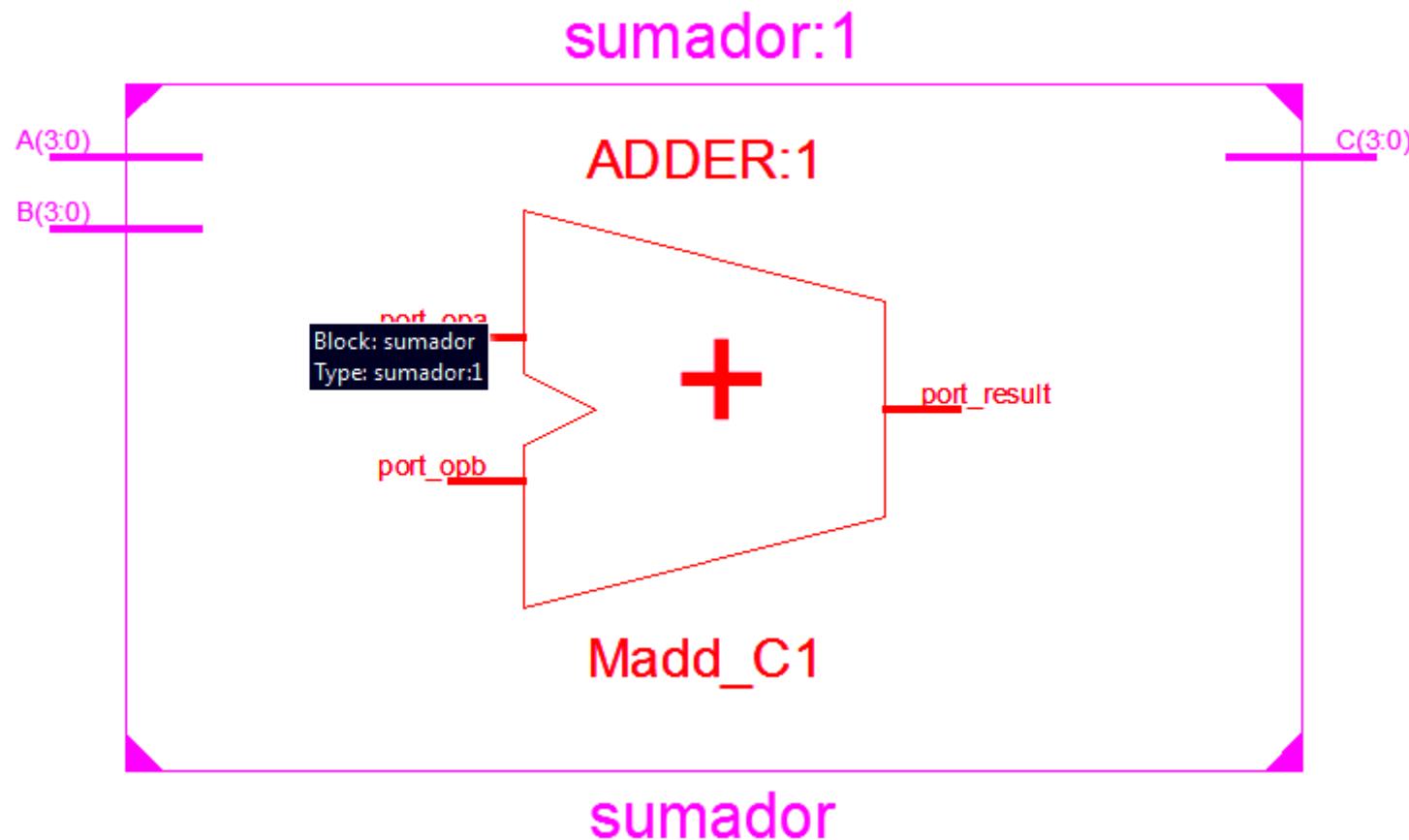
- Se genera un fichero de 0s y 1s que describe el circuito, se podrá encontrar en la carpeta del proyecto (nombre_proyecto.bit)



Implementar el diseño (III)



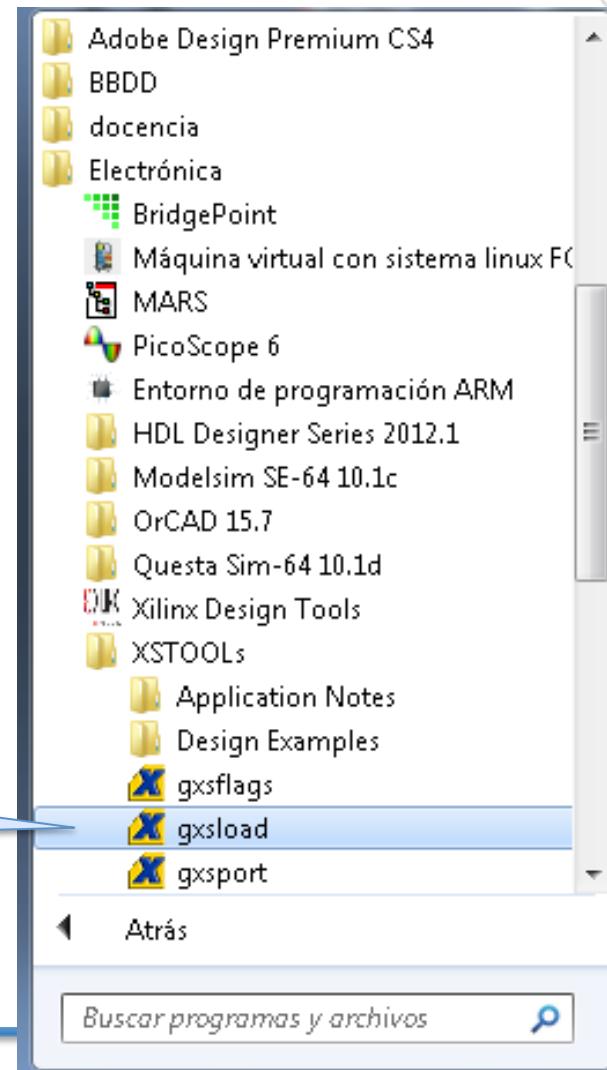
- Se puede visualizar el esquemático asociado a nuestro diseño VHDL



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”

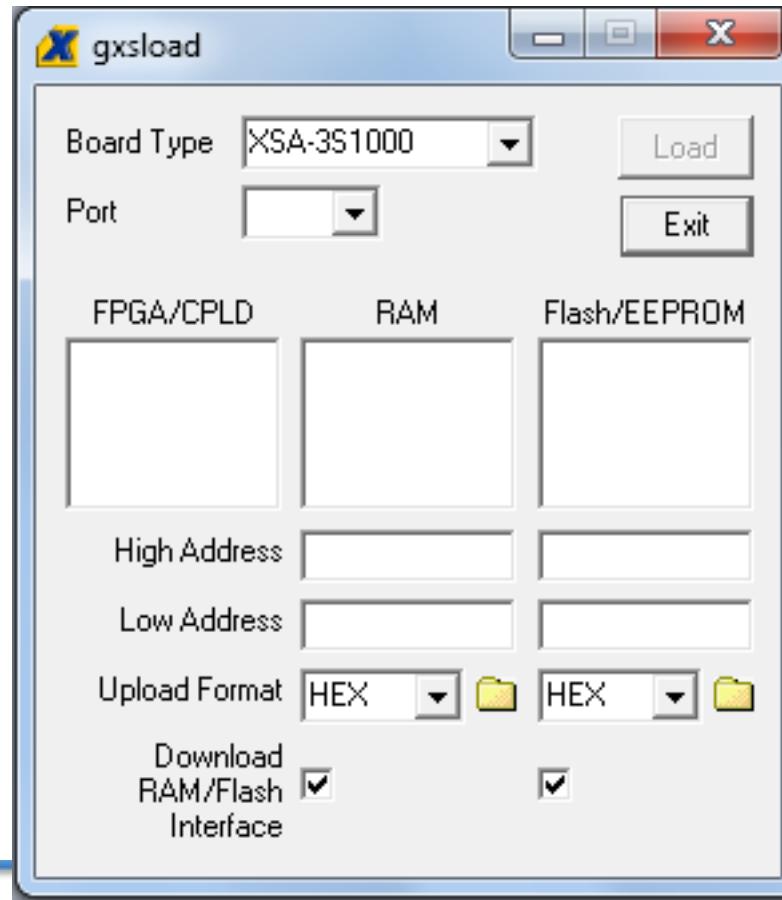
El programa es
accesible a través
de esta ruta



Descarga .bit sobre FPGA (II)



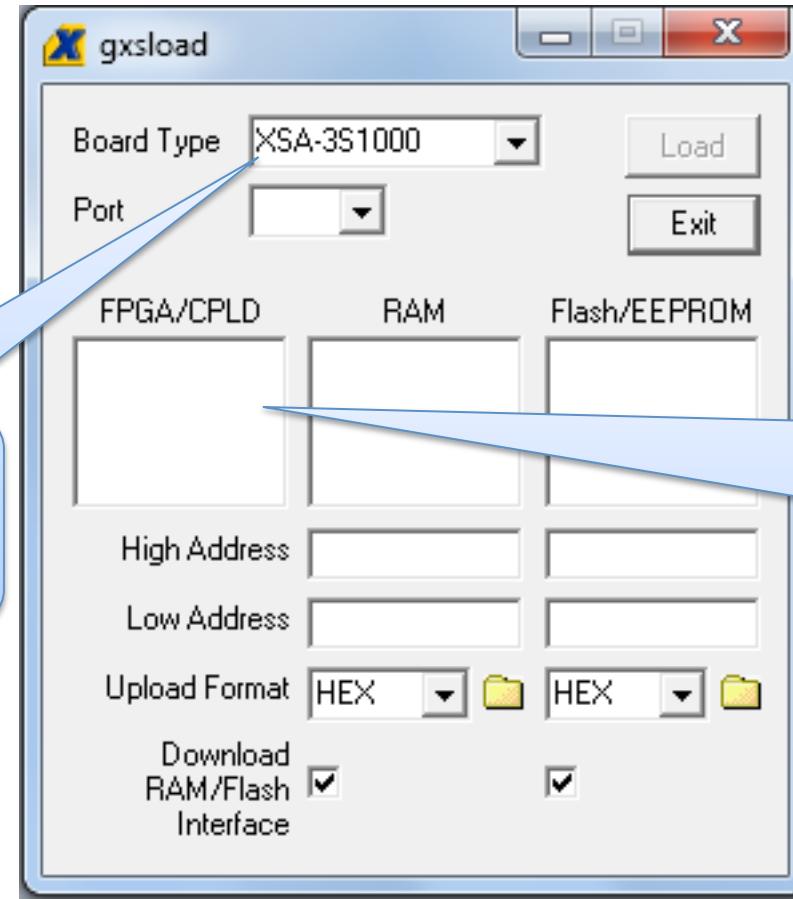
- Se abrirá la ventana de gxsload



Descarga .bit sobre FPGA (III)

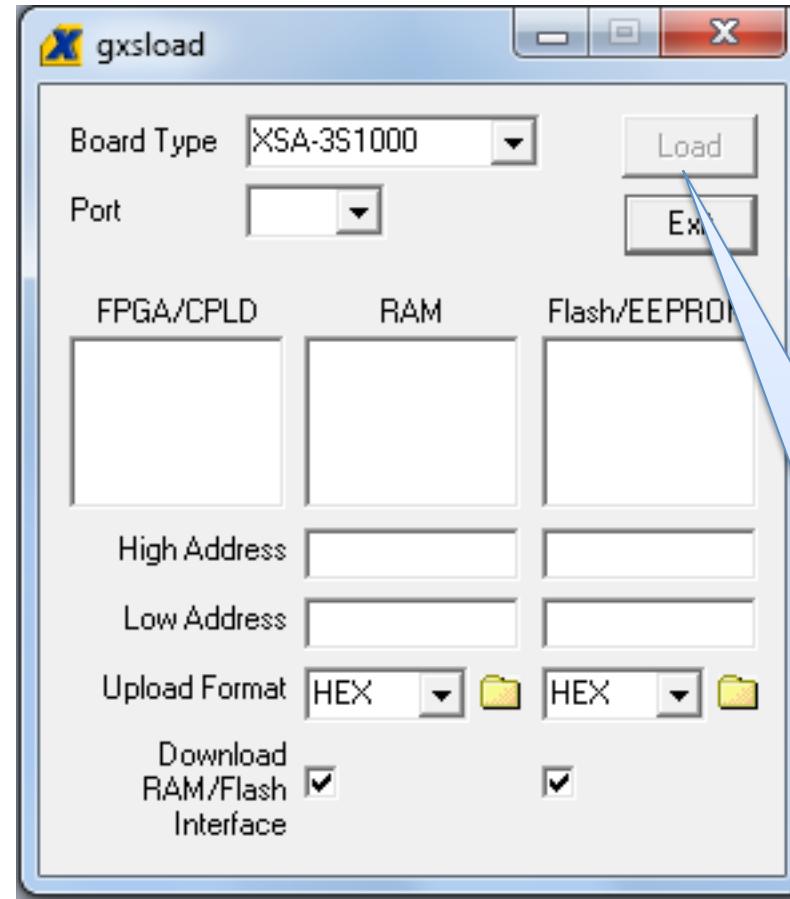


Aquí seleccionamos nuestro modelo de FPGA



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

Descarga .bit sobre FPGA (IV)



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

Descarga .bit sobre FPGA (V)



- 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

Práctica 1.a. Código



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sumador is
    port (op1 : in std_logic_vector (3 downto 0);
          op2 : in std_logic_vector (3 downto 0);
          res : out std_logic_vector (3 downto 0));
end sumador;

architecture rtl of sumador is
begin
    res <= std_logic_vector(unsigned(op1)+unsigned(op2));
end rtl;
```

Diseñar en VHDL un circuito sumador de números de 4 bits



Práctica 1.a. Testbench

```
-- Añadimos las librerías necesarias
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;

--entidad
entity tb_sumador is
end tb_sumador ;

--arquitectura
architecture beh of tb_sumador is
-- Declaración del componente que vamos a simular
component sumador
port( op1 : in std_logic_vector(3 downto 0);
      op2 : in std_logic_vector(3 downto 0);
      res : 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 res : std_logic_vector(3 downto 0);
```



Práctica 1.a. Testbench

```
begin
    -- Instanciacion de la unidad a simular
    dut: sumador port map (
        op1 => op1    ,
        op2 => op2    ,
        res => res
    );
    -- Proceso de estímulos
    p_stim: process
    begin
        op1<="0000";
        op2<="0000";
        wait for 100 ns;
        op1<="0101";
        op2<="0100";
        wait for 100 ns;
        op1<="0000";
        op2<="0111";
        wait for 100 ns;
        op1<="0011";
        op2<="1000";
        wait for 100 ns;
        op1<="1011";
        op2<="1111";
        wait for 100 ns;
        op1<="1001";
        op2<="0110";
        wait;
    end process p_stim;
end beh;
```



Práctica 1.b (I)

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



Práctica 1.b (II)

```
library ieee;
use ieee.std_logic_1164.all;

entity registro_siso is
    port (rst : in std_logic;
          clk : in std_logic;
          es  : in std_logic;
          ss  : out std_logic);
end registro_siso;

architecture rtl of registro_siso is
    signal clk_int : std_logic;
    signal data     : std_logic_vector(7 downto 0);

-----
-- Descomentar para implementacion
-----

component divisor is
    port(rst      : in std_logic;
          clk       : in std_logic;
          clk_1hz  : 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
54



Práctica 1.b (III)

```
begin
    -----
    -- Descomentar cuando se vaya a realizar la implementacion en la FPGA
    -----
    i_clk_int : divisor port map(
        rst      => rst,
        clk      => clk,
        clk_1hz => clk_int);
    -----
    -- Comentar cuando se vaya a realizar la implementacion en la FPGA
    -----
    clk_int <= clk;

    p_reg : process(clk_int, rst)
    begin
        if rst = '1' then
            data <= "00000000";
        elsif rising_edge(clk_int) then
            data(7)          <= es;
            data(6 downto 0) <= data(7 downto 1);
        end if;
    end process p_reg;

    ss <= data(0);
end rtl;
```

Para la implementación, necesitamos el divisor de frecuencia

En simulación el reloj es clk (100MHz) y en implementación es reloj (1Hz)



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 serie (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 reseteé el circuito no funcionará correctamente



Práctica 1.c

- Implementar a partir de la práctica 1.b un registro con entrada serie y salida paralelo de 8 bits. Suponer que el primer bit que entra en el registro es el más significativo

