

ÁREA ELECTRICIDAD Y ELECTRÓNICA



ELECTRÓNICA DIGITAL Y MICROCONTROLADORES



Profesor: Yeison Javier Montagut Ferizzola

LABORATORIO I: SISTEMAS DIGITALES EN FPGA

OBJETIVO DEL LABORATORIO: Implementar sistemas digitales en una FPGA.

Este documento servirá como tutorial para la realización de la práctica de laboratorio de la asignatura electrónica digital y microcontroladores; tiene como objetivo guiar a los estudiantes en la implementación de un sistema digital en una FPGA.

PROCEDIMIENTO:

1. Antes del laboratorio lea todo el tutorial y cree los archivos “Bitstream” para cada uno de los dos circuitos (combinacional y secuencial) del presente tutorial.
2. Durante el laboratorio implemente los desarrollos realizados (circuito combinacional y secuencial de esta guía) en el sistema de desarrollo Basys 3 y compruebe su funcionamiento.
3. Realice el ejercicio propuesto al final del tutorial.

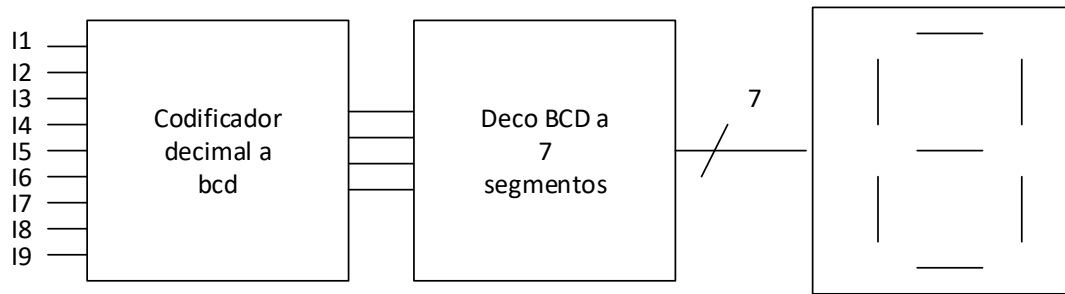
PROGRAMA:

- Vivado Design Suite WebPACK (lo pueden encontrar instalado en los computadores del laboratorio de sistemas eléctricos y electrónicos de la Universidad EIA o si lo prefieren lo pueden descargar de: <https://www.xilinx.com/products/design-tools/vivado/vivado-webpack.html>)

SISTEMAS COMBINACIONAL EN FPGA

Este documento muestra cómo se implementa el siguiente ejemplo:

Suponga que tiene nueve entradas conectadas a un codificador decimal a BCD el cual a su vez está conectado a un deco BCD a 7 segmentos, como lo muestra el siguiente diagrama de bloques:



La función del circuito es mostrar en el display 7 segmentos la entrada activa del codificador.

PASOS PARA IMPLEMENTAR UN SISTEMA COMBINACIONAL EN UNA FPGA

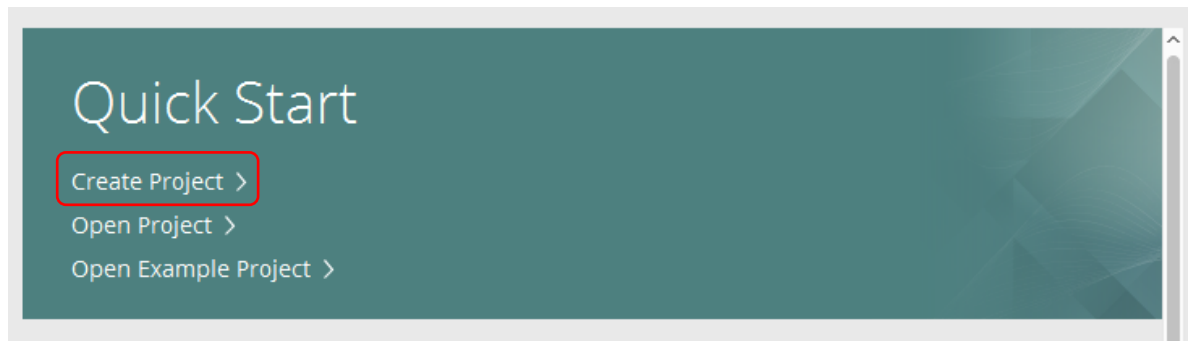
Para implementar el sistema combinacional del ejemplo en una FPGA se divide en dos subsistemas: codificador decimal a BCD y Deco BCD a 7 segmentos.

SUBSISTEMA CODIFICADOR DECIMAL A BCD

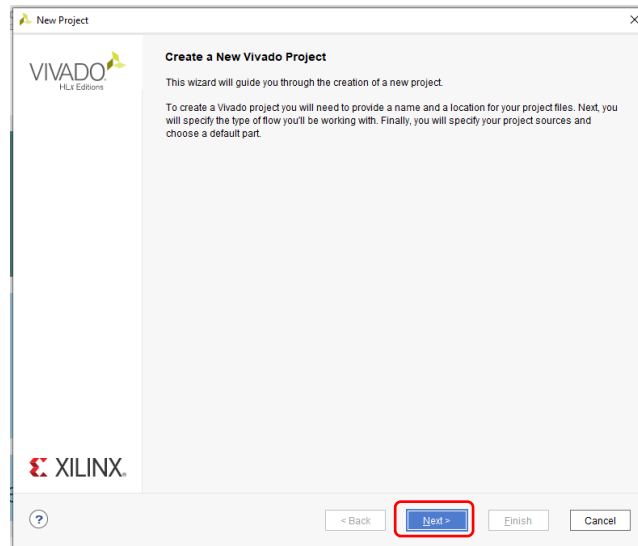
1. Iniciar el programa Vivado, dando doble clic en el icono



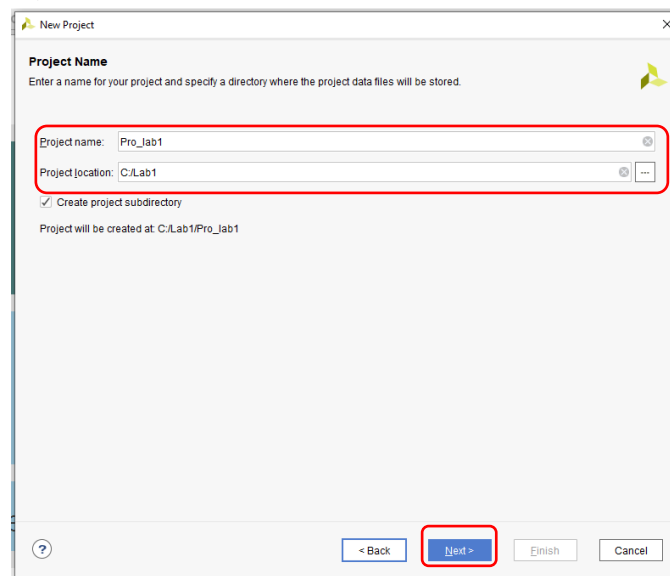
2. Clic en "Create Project"



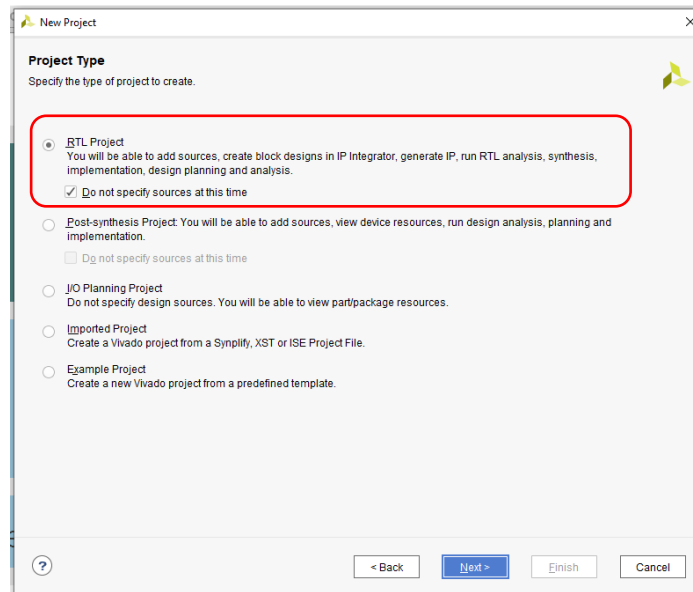
3. Se abre la siguiente ventana y clic en "Next"



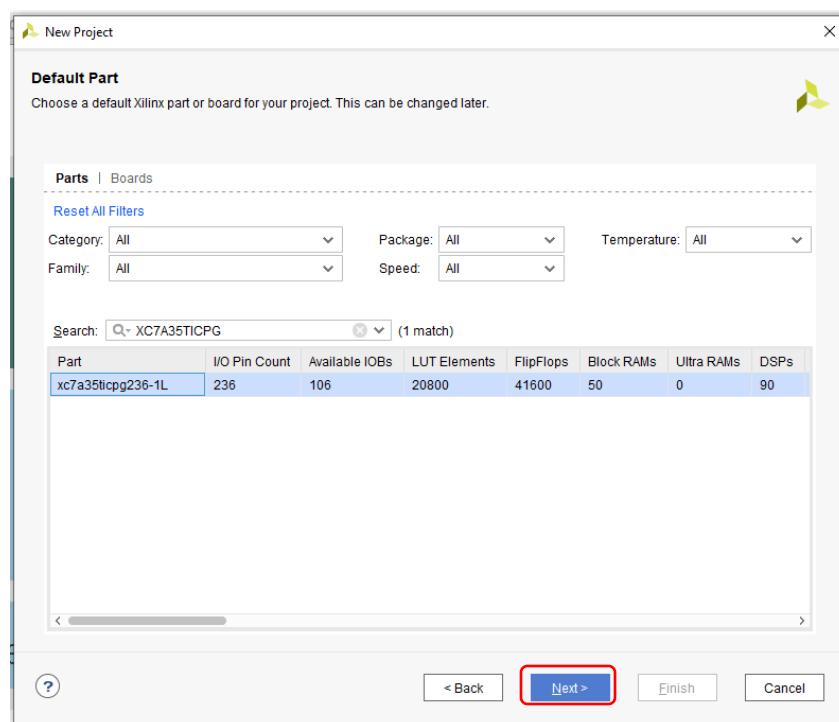
4. Dar un nombre y una ubicación al proyecto, evitar poner punto (.) en el nombre del proyecto. La ubicación debe ser fácil de encontrar preferiblemente en una carpeta del directorio raíz (C://). Clic en “Next”



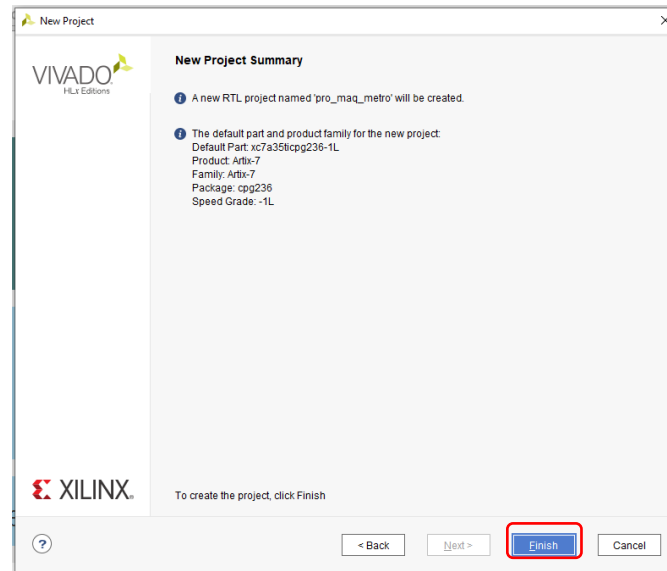
5. Seleccionar “RTL Project”



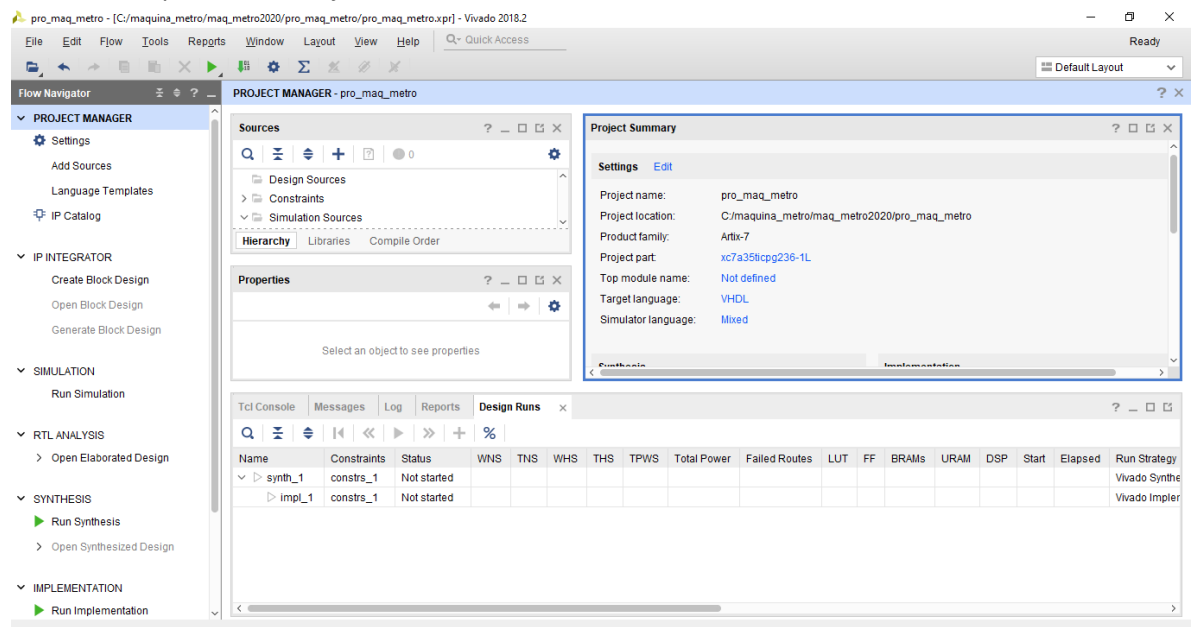
6. Seleccionar la FPGA con la que se va a trabajar, en este caso: xc7a35ticipg236-1L, clic en “Next”



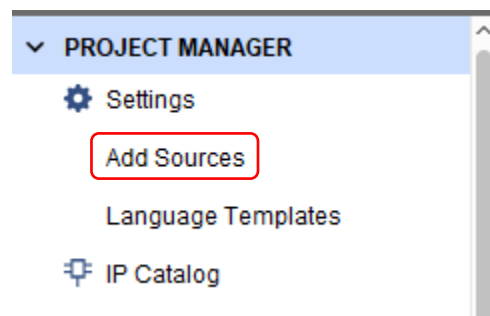
7. Aparece la siguiente ventana que indica que el proyecto se ha creado. Clic en “Finish”



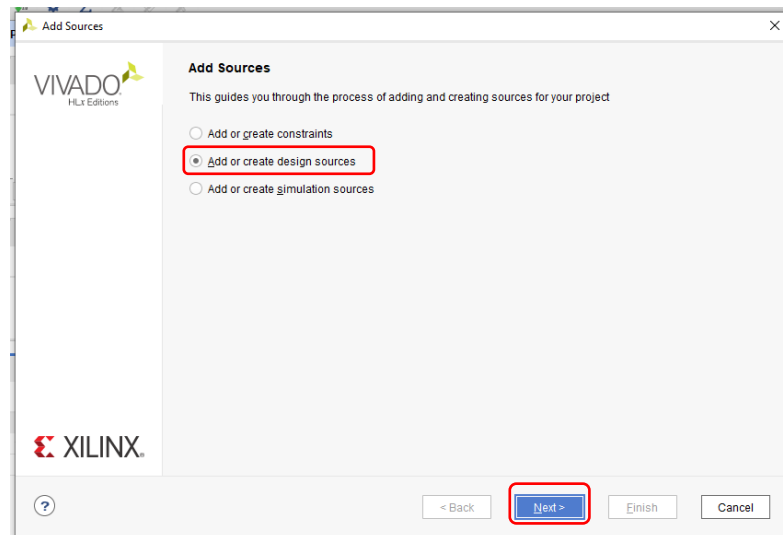
8. Se abre el espacio de trabajo de Vivado



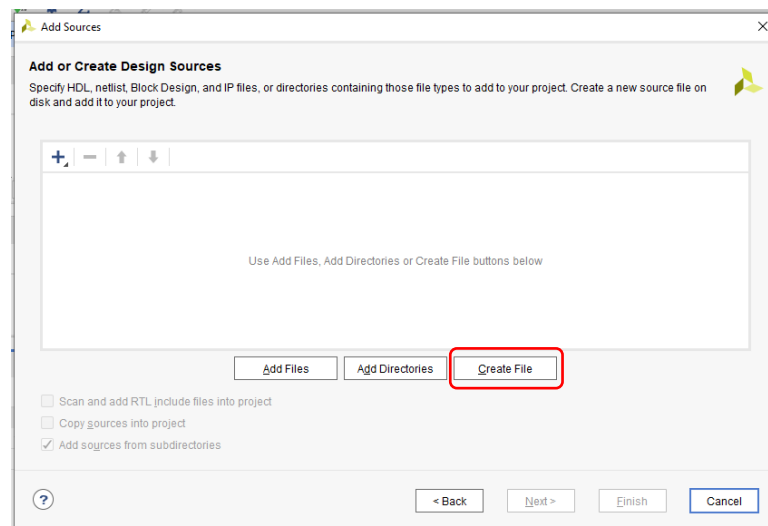
9. A continuación, se agrega el archivo vhd que permitirá generar el codificador, para ello clic en "Add Sources"



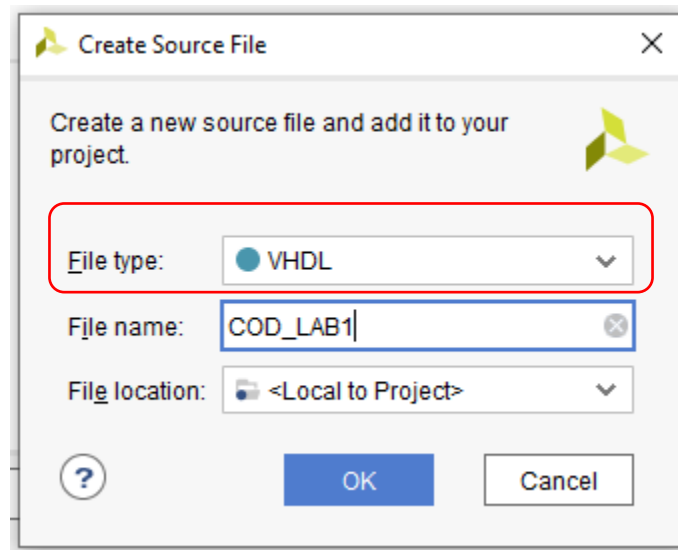
10. Aparece la siguiente ventana, en ella, seleccionar: "Add or create design sources" y clic en "Next"



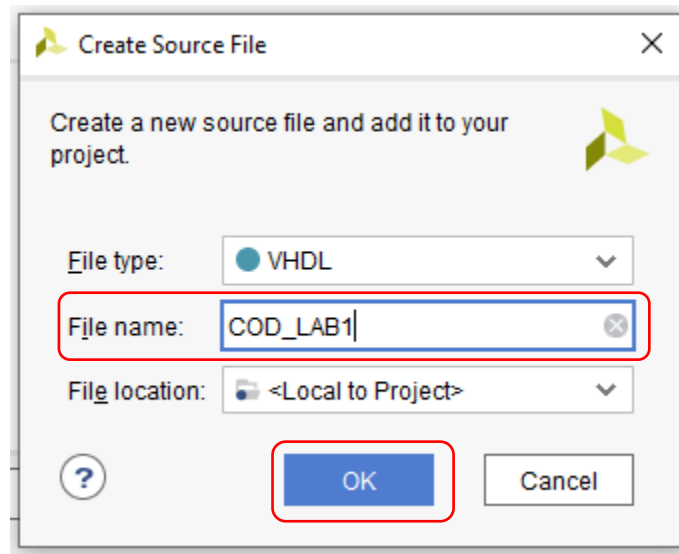
11. En la siguiente ventana clic en: “*Create Files*”



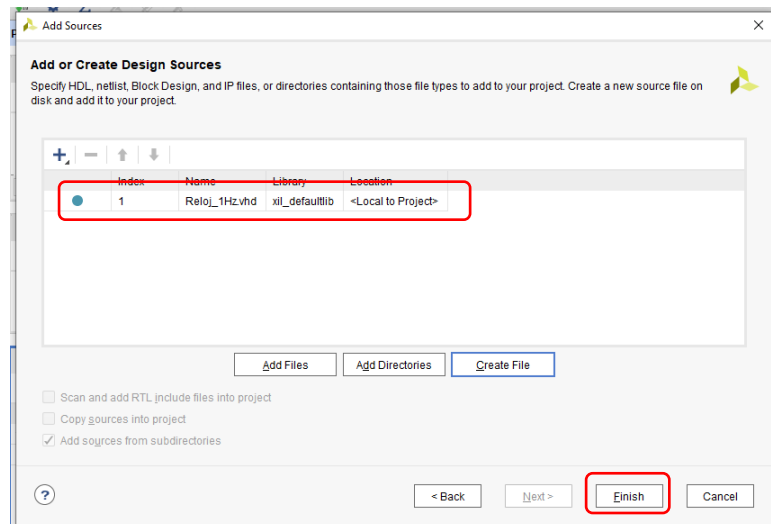
12. En la nueva ventana se selección “VHDL” en *File Type*



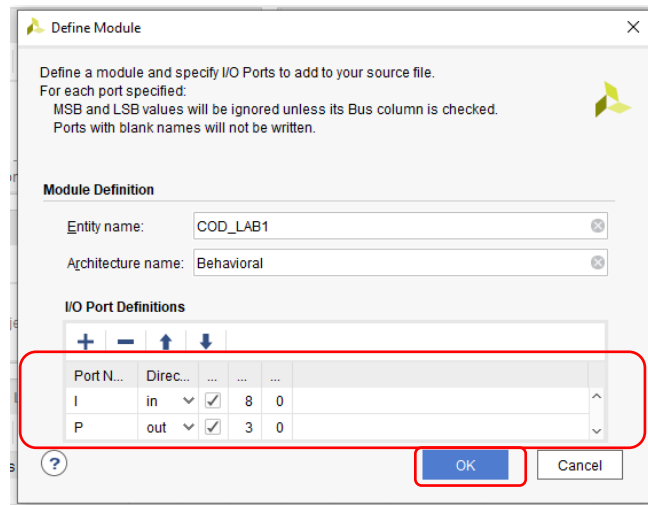
13. Se le asigna un nombre al archivo en *File name* y clic en “Ok”



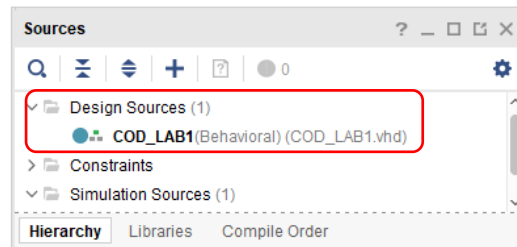
14. Revisar que aparezca el nuevo archivo creado y clic en “Finish”



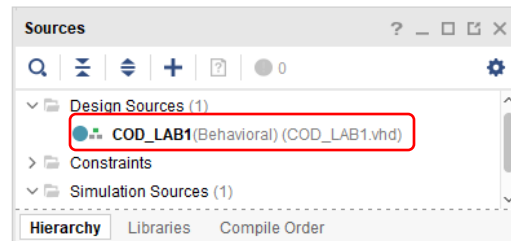
15. A continuación, crear las entradas y salidas que va a tener el codificador, en este caso una entrada (I) de 9 bits y una salida (P) de 4 bits y clic en “Ok”



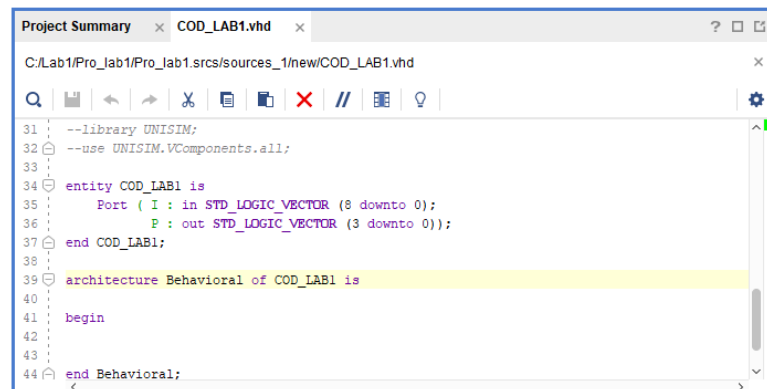
16. El nuevo archivo creado debe aparecer en:



17. Doble clic en el archivo creado



18. Se abre el archivo creado



19. En el archivo creado escribir el siguiente código:


```

33
34 entity COD_LAB1 is
35     Port ( I : in STD_LOGIC_VECTOR (8 downto 0);
36           P : out STD_LOGIC_VECTOR (3 downto 0));
37 end COD_LAB1;
38
39 architecture Behavioral of COD_LAB1 is
40     signal al : STD_LOGIC_VECTOR (3 downto 0);
41 begin
42     process (I)
43     begin
44         case (I) is
45             when "000000000" => al <= "0000";
46             when "000000001" => al <= "0001";
47             when "000000010" => al <= "0010";
48             when "000000100" => al <= "0011";
49             when "000001000" => al <= "0100";
50             when "000010000" => al <= "0101";
51             when "000100000" => al <= "0110";
52             when "001000000" => al <= "0111";
53             when "010000000" => al <= "1000";
54             when "100000000" => al <= "1001";
55             when others      => al <= "1111";
56         end case;
57     end process;
58     P <= al;
59
60 end Behavioral;
61

```

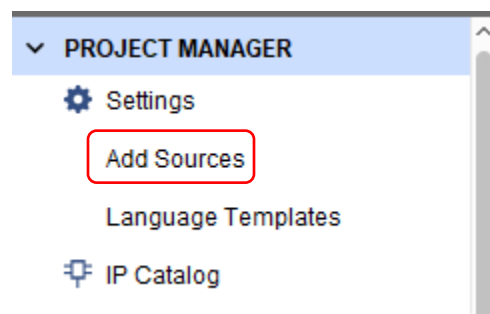
20. Clic en guardar



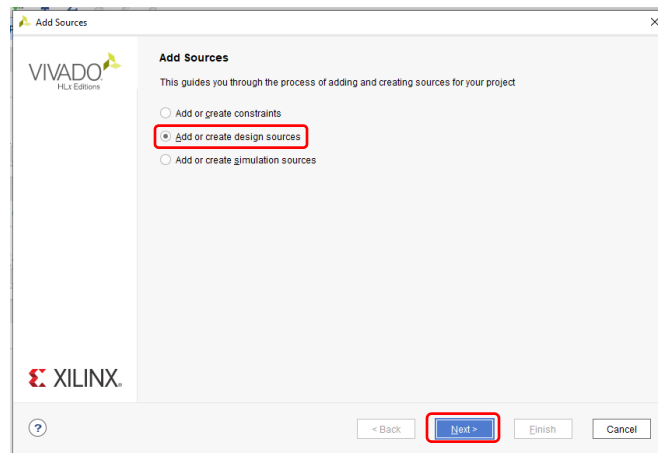
SUBSISTEMA DECODIFICADOR BCD A 7 SEGMENTOS

A continuación, se debe crear el archivo VHDL que deberá contener el decodificador BCD a 7 Segmentos.

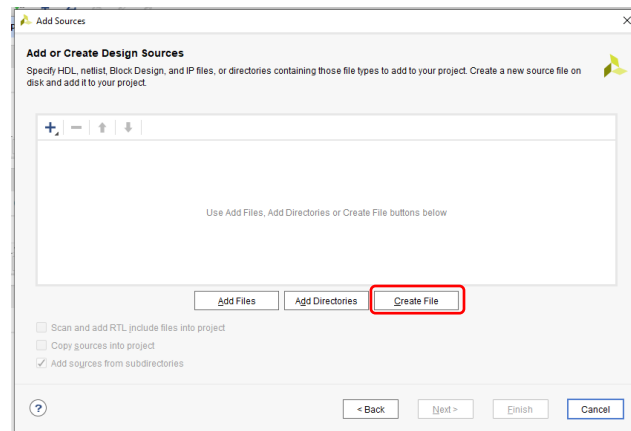
21. Clic en “Add Sources”



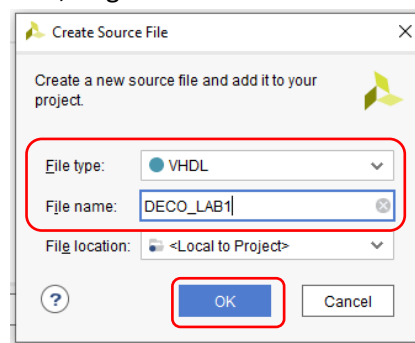
22. Seleccionar “Add or create design sources” y clic en “Next”



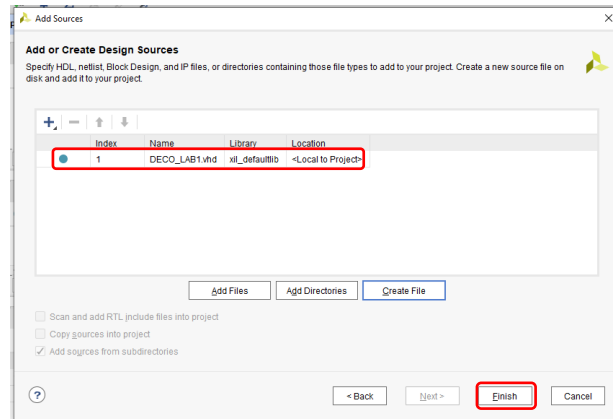
23. Clic en “Create File”



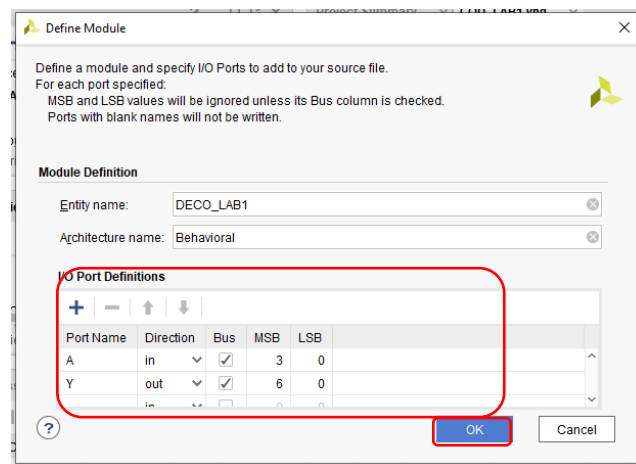
24. Seleccionar archivo tipo “VHDL”, asignar un nombre al nuevo archivo y clic en “OK”



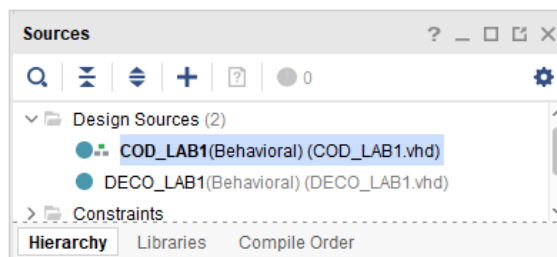
25. El archivo creado deberá aparecer en la siguiente ventana y clic en “Finish”



26. Agregar las entradas y salidas, en este caso el decodificador tiene una entrada (A) de 4 bits y una salida (Y) de 7 bits. Clic en “OK”



27. El nuevo archivo creado aparece en



28. Doble clic en el nuevo archivo para crear el código en VHDL de la máquina del metro

```

34 entity DECO_LAB1 is
35     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
36           Y : out STD_LOGIC_VECTOR (6 downto 0));
37 end DECO_LAB1;
38
39 architecture Behavioral of DECO_LAB1 is
40     signal a2 : STD_LOGIC_VECTOR (6 downto 0);
41 begin
42     process (A)
43     begin
44         case (A) is
45             when "0000" => a2 <= "1000000";
46             when "0001" => a2 <= "1111001";
47             when "0010" => a2 <= "0100100";
48             when "0011" => a2 <= "0110000";
49             when "0100" => a2 <= "0011001";
50             when "0101" => a2 <= "0010010";
51             when "0110" => a2 <= "0000010";
52             when "0111" => a2 <= "1111000";
53             when "1000" => a2 <= "0000000";
54             when "1001" => a2 <= "0011000";
55             when others => a2 <= "1111111";
56         end case;
57     end process;
58     Y <= a2;
59
60 begin
61
62
63 end Behavioral;
64

```

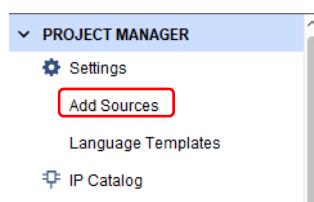
29. Clic en guardar



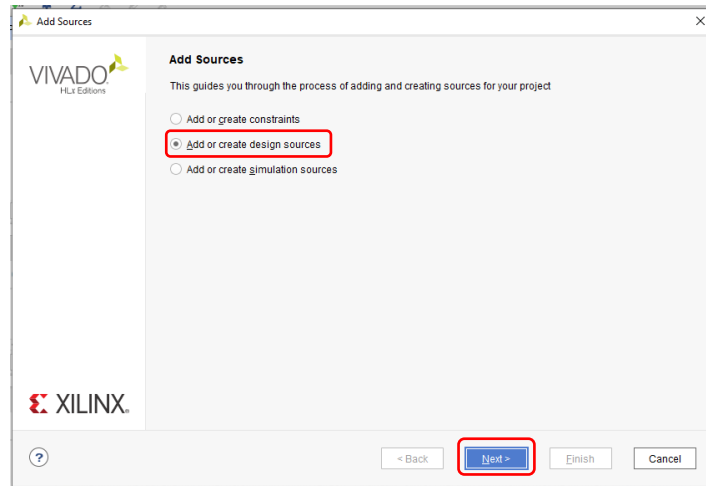
SISTEMA GENERAL

A continuación, se debe crear un archivo que contenga a todos los dos subsistemas creados, a ese archivo lo llamaremos archivo TOP.

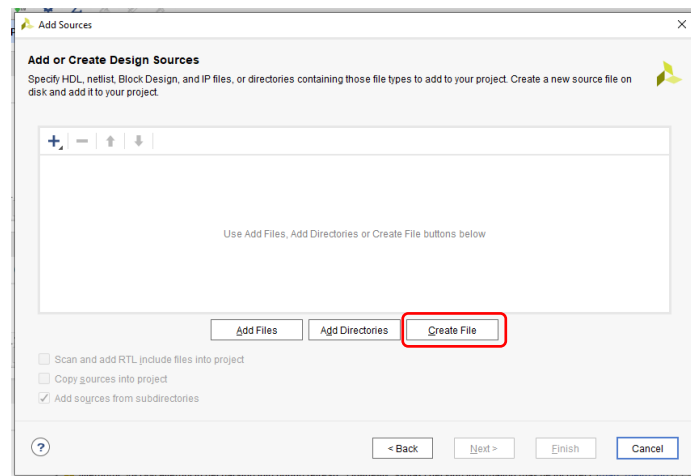
30. Clic en "Add Sources"



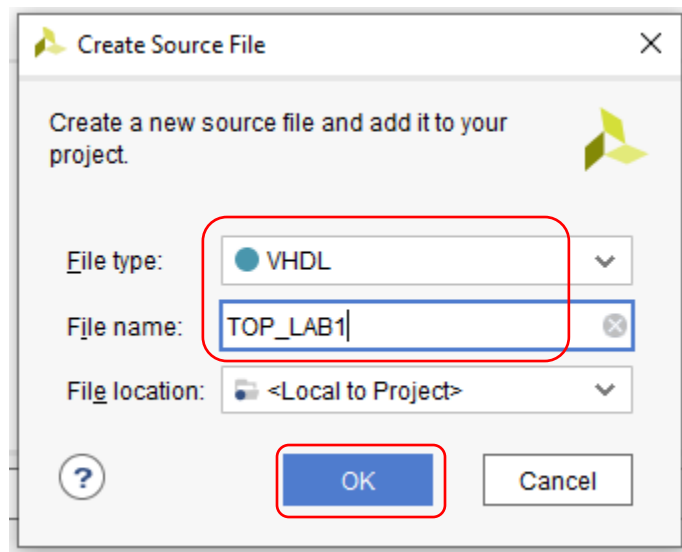
31. Seleccionar y clic en "Next"



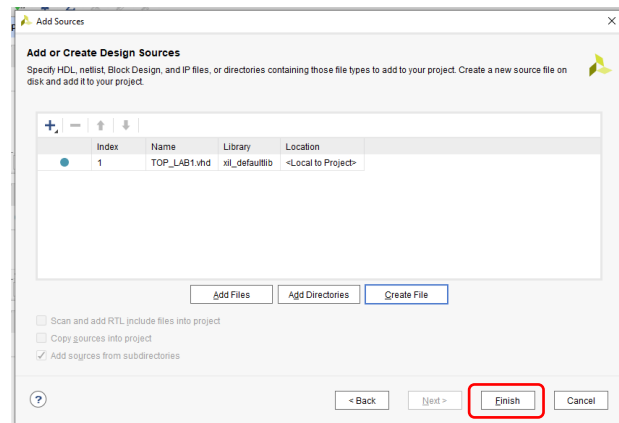
32. Clic en



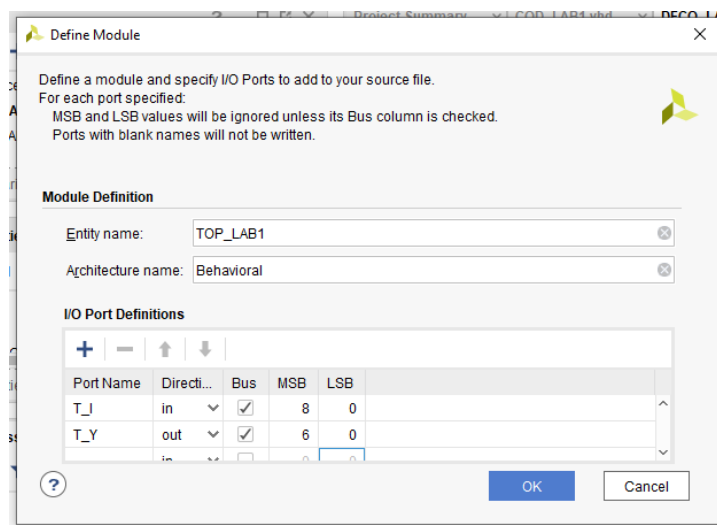
33. Nombre y tipo de archivo



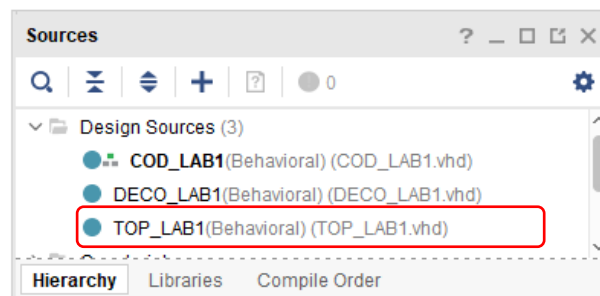
34. Clic en



35. Se crean las salidas y entradas del sistema que va a contener a los otros dos subsistemas y clic en “OK”



36. Aparece un nuevo archivo en



37. Doble clic en ese archivo

38. Se comienzan a agregar la entidad de los otros subsistemas como si fueran componentes del nuevo sistema

```

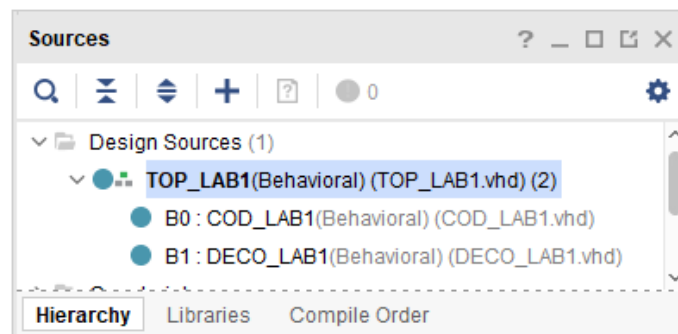
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity TOP_LAB1 is
35      Port ( T_I : in STD_LOGIC_VECTOR (8 downto 0);
36            T_Y : out STD_LOGIC_VECTOR (6 downto 0));
37  end TOP_LAB1;
38
39  architecture Behavioral of TOP_LAB1 is
40      component COD_LAB1 is
41          Port ( I : in STD_LOGIC_VECTOR (8 downto 0);
42                P : out STD_LOGIC_VECTOR (3 downto 0));
43      end component;
44      component DECO_LAB1 is
45          Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
46                Y : out STD_LOGIC_VECTOR (6 downto 0));
47      end component;
48
49      signal aux : STD_LOGIC_VECTOR (3 downto 0);
50  begin
51      B0 : COD_LAB1 PORT MAP(T_I,aux);
52      B1 : DECO_LAB1 PORT MAP(aux,T_Y);
53
54  end Behavioral;

```

39. Clic en guardar



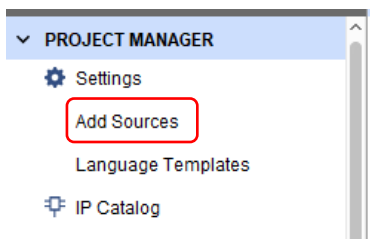
40. Observar el nuevo aspecto de



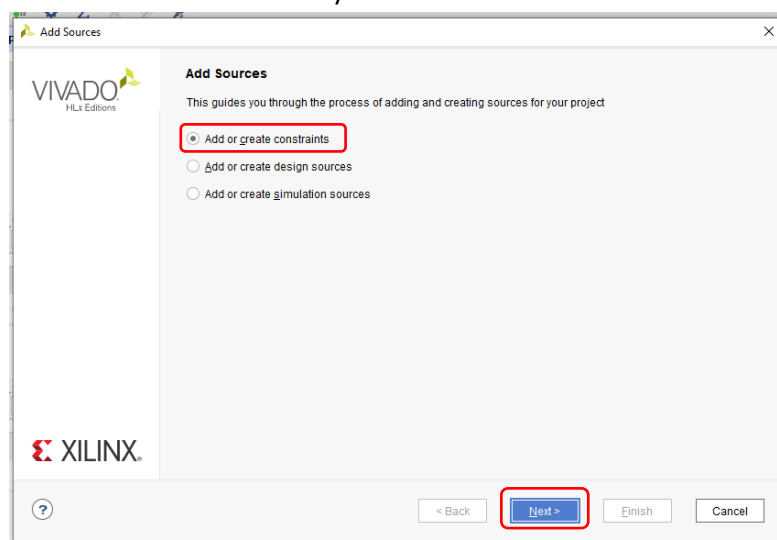
CONEXIÓN DE LAS ENTRADAS Y SALIDAS DEL SISTEMA DESARROLLADO CON LAS ENTRADAS Y SALIDA DE LA BASYS 3

Una vez creado el archivo de programación lo que viene a continuación es la conexión de las entradas y salida del sistema desarrollado con los pulsadores, señal de reloj y led de la placa de desarrollo Basys 3

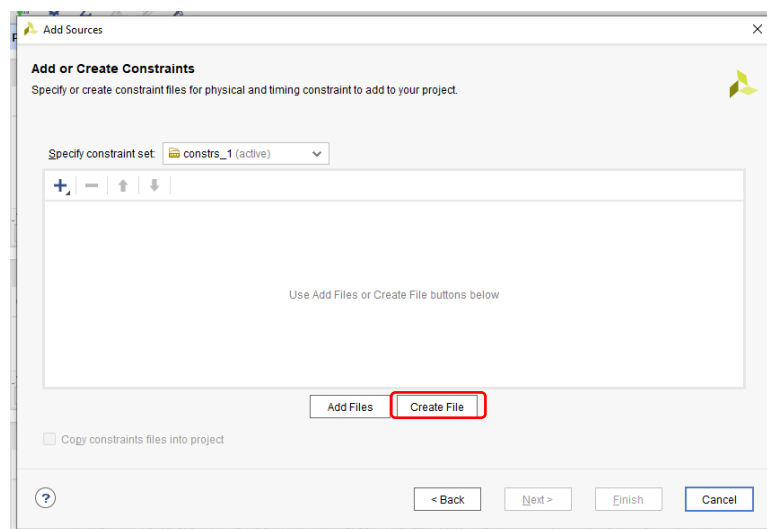
41. Clic en



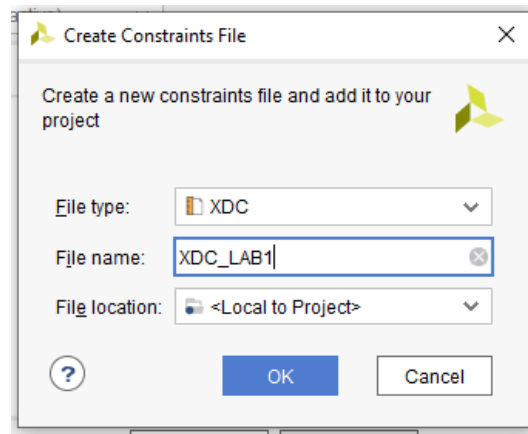
42. Seleccionar “Add or create constraints” y clic en “Next”



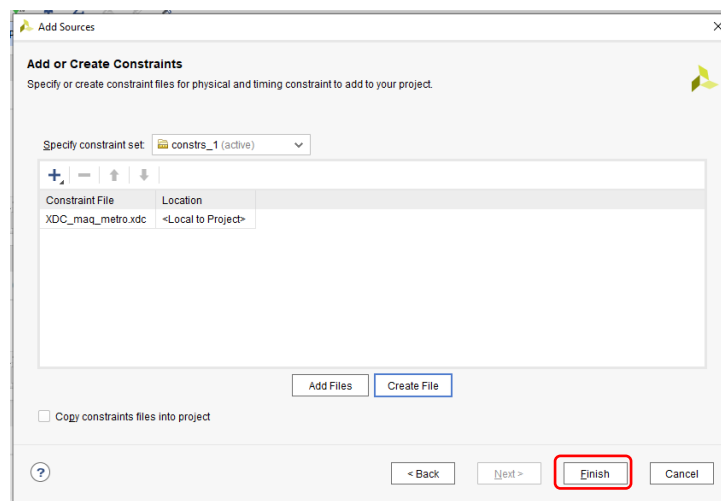
43. Clic en “Create File”



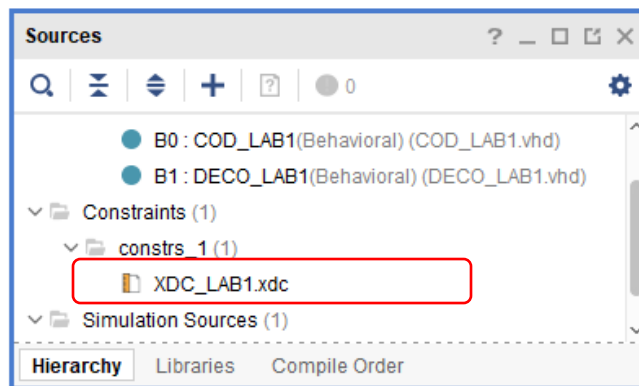
44. Se le asigna un nombre y clic en “OK”



45. Clic en “Finish”



46. El nuevo archivo creado aparece en



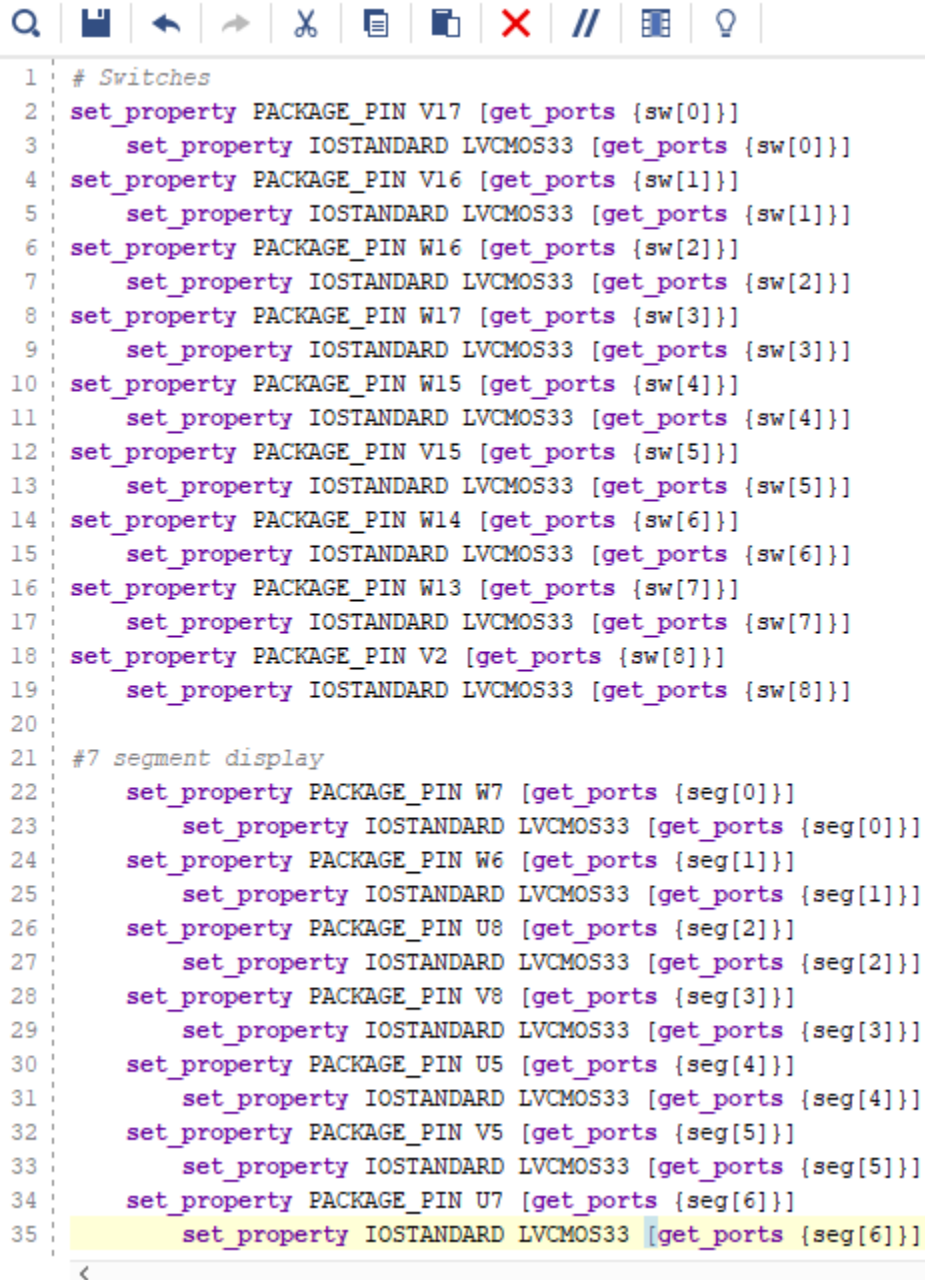
47. Doble clic sobre el nuevo archivo creado y es posible editarlo

48. Para editar el archivo creado se debe ingresar a la siguiente dirección:

https://github.com/Digilent/Basys3/blob/master/Projects/XADC_Demo/src/constraints/Basys3_Master.xdc

Copiar las entradas, salidas que se necesitan y pegarlos en el nuevo archivo creado.

49. El archivo creado deberá tener el siguiente aspecto:

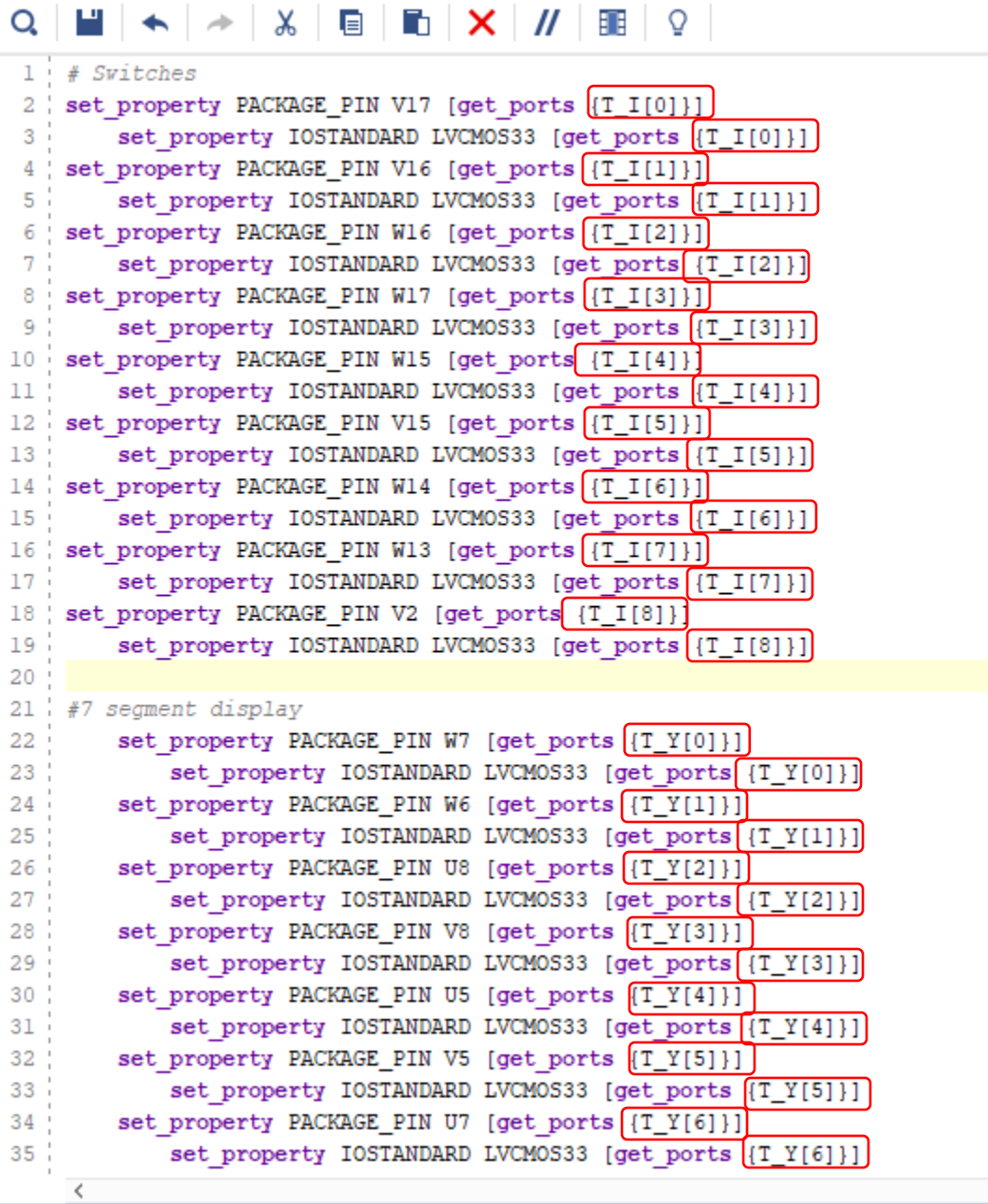


```

1  # Switches
2  set_property PACKAGE_PIN V17 [get_ports {sw[0]]}
3      set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]]}
4  set_property PACKAGE_PIN V16 [get_ports {sw[1]]}
5      set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]]}
6  set_property PACKAGE_PIN W16 [get_ports {sw[2]]}
7      set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]]}
8  set_property PACKAGE_PIN W17 [get_ports {sw[3]]}
9      set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]]}
10 set_property PACKAGE_PIN W15 [get_ports {sw[4]]}
11     set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]]}
12 set_property PACKAGE_PIN V15 [get_ports {sw[5]]}
13     set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]]}
14 set_property PACKAGE_PIN W14 [get_ports {sw[6]]}
15     set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]]}
16 set_property PACKAGE_PIN W13 [get_ports {sw[7]]}
17     set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]]}
18 set_property PACKAGE_PIN V2 [get_ports {sw[8]]}
19     set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]]}
20
21 #7 segment display
22     set_property PACKAGE_PIN W7 [get_ports {seg[0]]}
23         set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]]}
24     set_property PACKAGE_PIN W6 [get_ports {seg[1]]}
25         set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]]}
26     set_property PACKAGE_PIN U8 [get_ports {seg[2]]}
27         set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]]}
28     set_property PACKAGE_PIN V8 [get_ports {seg[3]]}
29         set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]]}
30     set_property PACKAGE_PIN U5 [get_ports {seg[4]]}
31         set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]]}
32     set_property PACKAGE_PIN V5 [get_ports {seg[5]]}
33         set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]]}
34     set_property PACKAGE_PIN U7 [get_ports {seg[6]]}
35         set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]]}

```

50. Reemplazar en el nuevo archivo creado el nombre de las entradas y salidas por los nombres usados en el proyecto



```

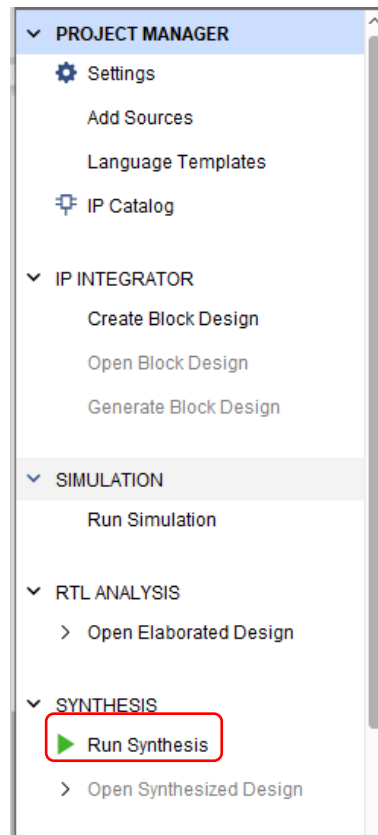
1  # Switches
2  set_property PACKAGE_PIN V17 [get_ports {T_I[0]}]
3      set_property IOSTANDARD LVCOS33 [get_ports {T_I[0]}]
4  set_property PACKAGE_PIN V16 [get_ports {T_I[1]}]
5      set_property IOSTANDARD LVCOS33 [get_ports {T_I[1]}]
6  set_property PACKAGE_PIN W16 [get_ports {T_I[2]}]
7      set_property IOSTANDARD LVCOS33 [get_ports {T_I[2]}]
8  set_property PACKAGE_PIN W17 [get_ports {T_I[3]}]
9      set_property IOSTANDARD LVCOS33 [get_ports {T_I[3]}]
10 set_property PACKAGE_PIN W15 [get_ports {T_I[4]}]
11     set_property IOSTANDARD LVCOS33 [get_ports {T_I[4]}]
12 set_property PACKAGE_PIN V15 [get_ports {T_I[5]}]
13     set_property IOSTANDARD LVCOS33 [get_ports {T_I[5]}]
14 set_property PACKAGE_PIN W14 [get_ports {T_I[6]}]
15     set_property IOSTANDARD LVCOS33 [get_ports {T_I[6]}]
16 set_property PACKAGE_PIN W13 [get_ports {T_I[7]}]
17     set_property IOSTANDARD LVCOS33 [get_ports {T_I[7]}]
18 set_property PACKAGE_PIN V2 [get_ports {T_I[8]}]
19     set_property IOSTANDARD LVCOS33 [get_ports {T_I[8]}]
20
21 #7 segment display
22 set_property PACKAGE_PIN W7 [get_ports {T_Y[0]}]
23     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[0]}]
24 set_property PACKAGE_PIN W6 [get_ports {T_Y[1]}]
25     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[1]}]
26 set_property PACKAGE_PIN U8 [get_ports {T_Y[2]}]
27     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[2]}]
28 set_property PACKAGE_PIN V8 [get_ports {T_Y[3]}]
29     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[3]}]
30 set_property PACKAGE_PIN U5 [get_ports {T_Y[4]}]
31     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[4]}]
32 set_property PACKAGE_PIN V5 [get_ports {T_Y[5]}]
33     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[5]}]
34 set_property PACKAGE_PIN U7 [get_ports {T_Y[6]}]
35     set_property IOSTANDARD LVCOS33 [get_ports {T_Y[6]}]

```

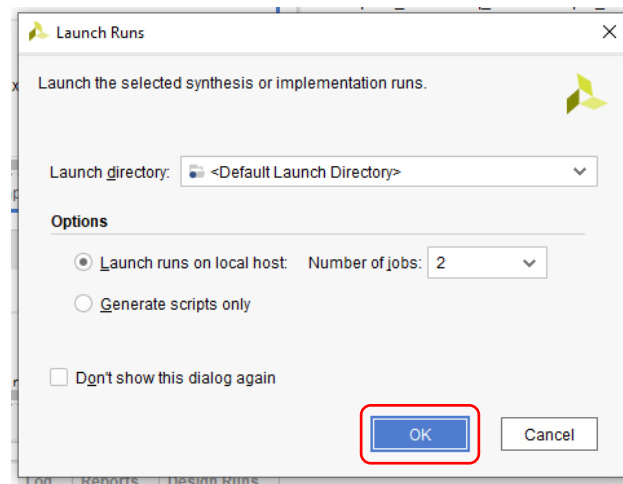
51. A continuación, clic en guardar



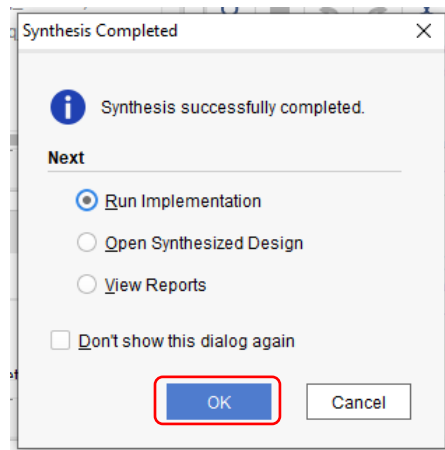
52. Y luego clic en “Run Synthesis”



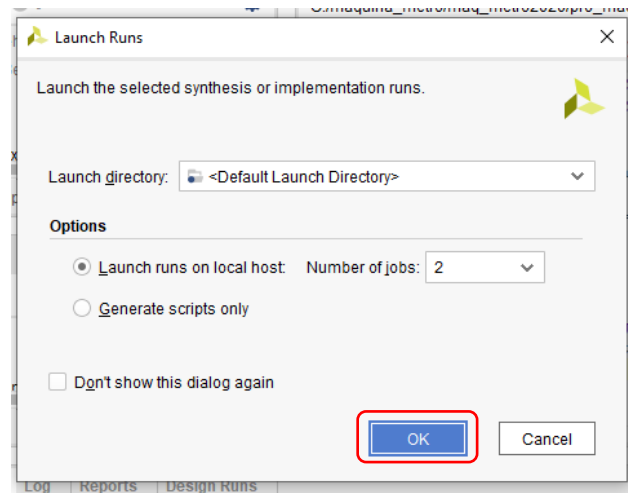
53. Clic en “OK”



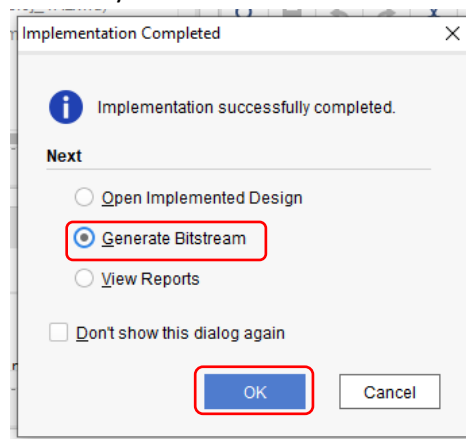
54. Esperar, una vez terminado clic en “OK”



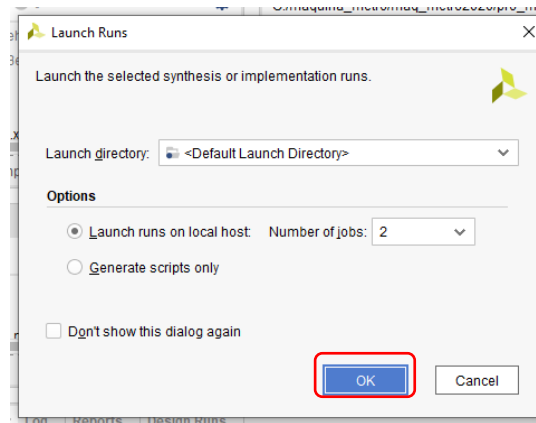
55. Clic en “OK”



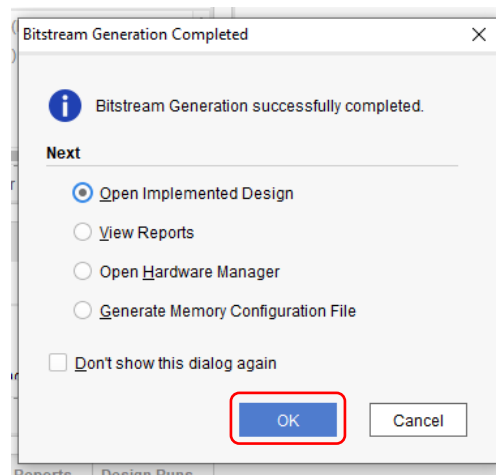
56. Seleccionar “Generate Bitstream” y clic en OK



57. Clic en “OK”



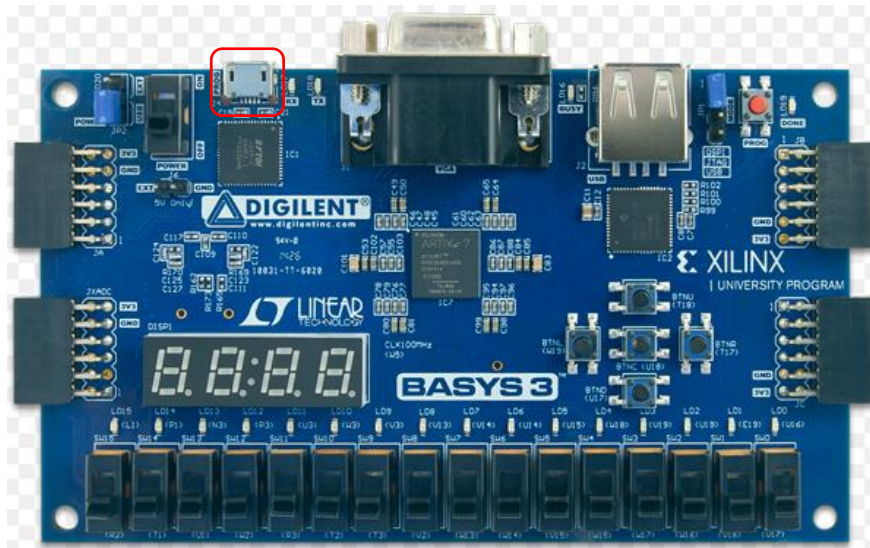
58. Clic en “OK”



PROGRAMACIÓN

A continuación, se continua con la programación de la FPGA y comprobación de funcionamiento

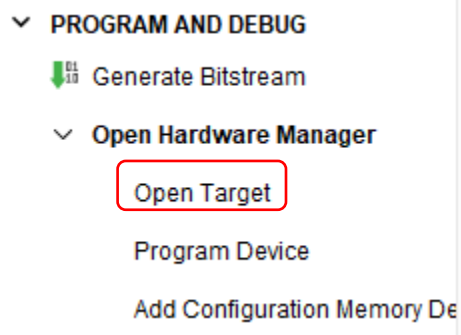
59. Conectar el sistema de desarrollo Basys 3 con el PC por medio de un cable USB. El cable USB se debe conectar a la Basys 3 por medio del conector señalado



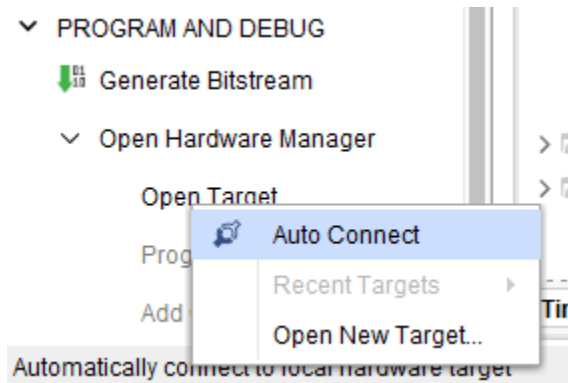
60. Encender el sistema de desarrollo Basys 3



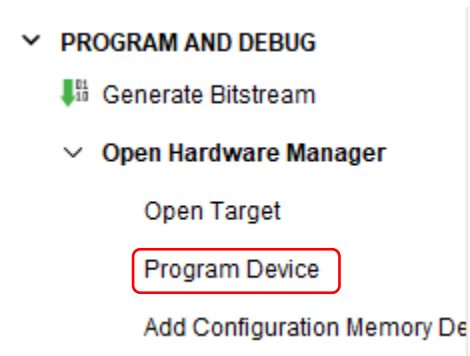
61. En “Open Hardware Manager” clic en “Open Target”



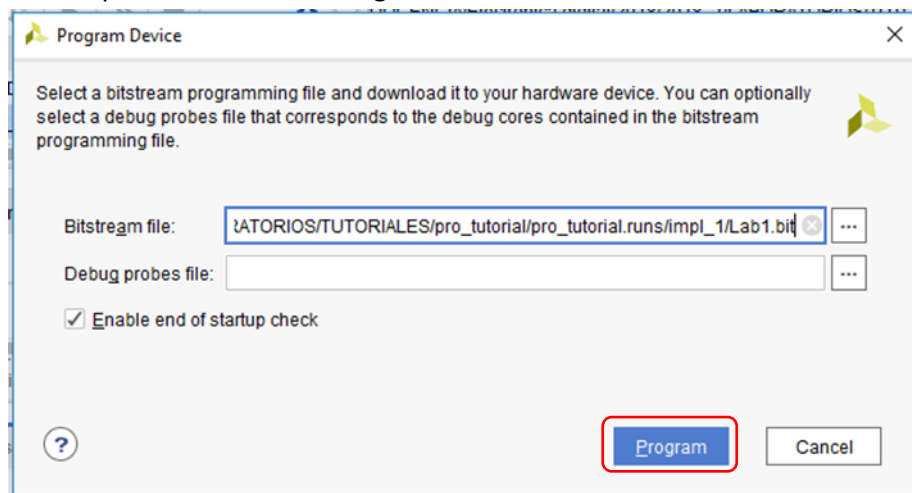
62. Clic en “Auto Connect”



63. Clic en “Program Device” y luego clic en la pestaña que se activa “xc7a35t_0”



64. En la ventana que se abre clic en “Program”



SISTEMAS SECUENCIAL EN FPGA

Este documento muestra cómo se implementa el siguiente ejemplo:

Diseñar un contador de 4 bits y mostrar el resultado en 4 leds.

PASOS PARA IMPLEMENTAR UN SISTEMA SECUENCIAL EN UNA FPGA

Para implementar el sistema secuencial del ejemplo en una FPGA se divide en dos subsistemas: reloj y contador.

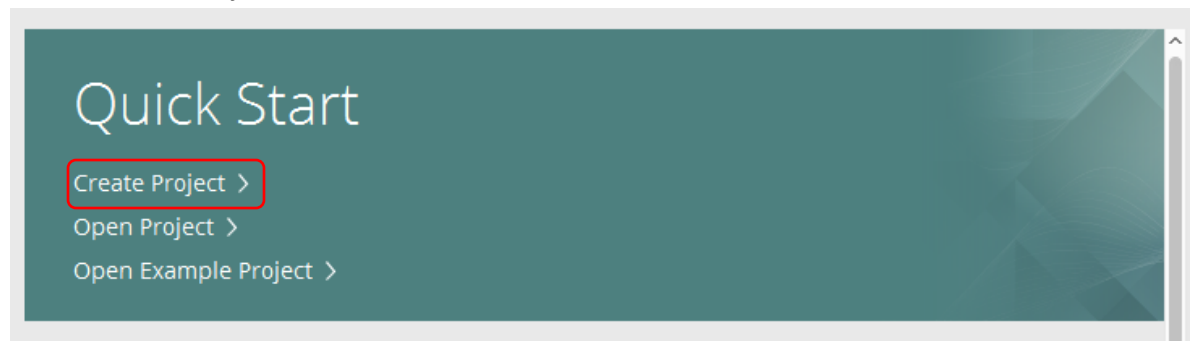
SUBSISTEMA RELOJ

El sistema de desarrollo Basys 3 cuenta con un reloj de 100MHz, sin embargo, para nosotros es imposible ver el funcionamiento de un contador con una señal de reloj de 100MHz, por esta razón, se hace necesario modificar la frecuencia de la señal de reloj a una frecuencia tal que permita ver en los cuatro leds el funcionamiento del contador; la frecuencia deseada será de 1 Hz, de esta manera, cada 1 segundo se incrementará el contador en uno. El circuito que convierte una frecuencia mayor en una menor es conocido como divisor de frecuencia, en este caso se implementará un circuito divisor de frecuencia de 100MHz a 1Hz. El principio de funcionamiento consiste en mantener una salida en alto durante 50 millones de flancos de subida de la señal de reloj (100 MHz) y posteriormente la salida se pone en bajo durante los otros 50 millones de flancos de subida de la señal de reloj, en total pasaran 100 millones de flancos de la señal de reloj (100MHz) por cada un flanco de subida de la nueva señal (1Hz). Los pasos a continuación describen la implementación del divisor de frecuencia.

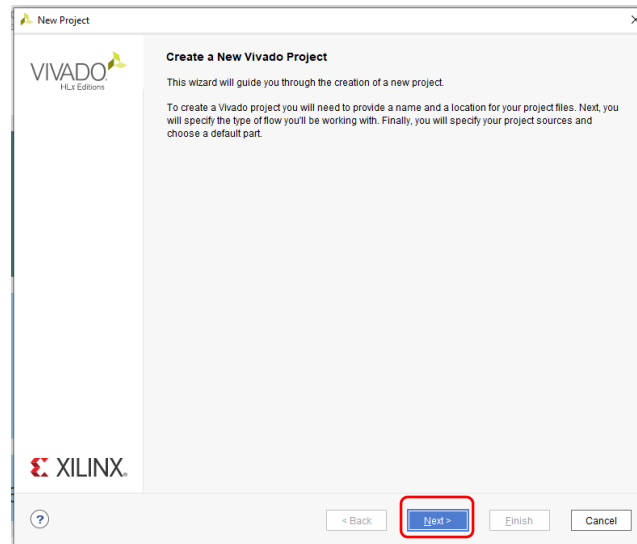
1. Iniciar el programa Vivado, dando doble clic en el icono



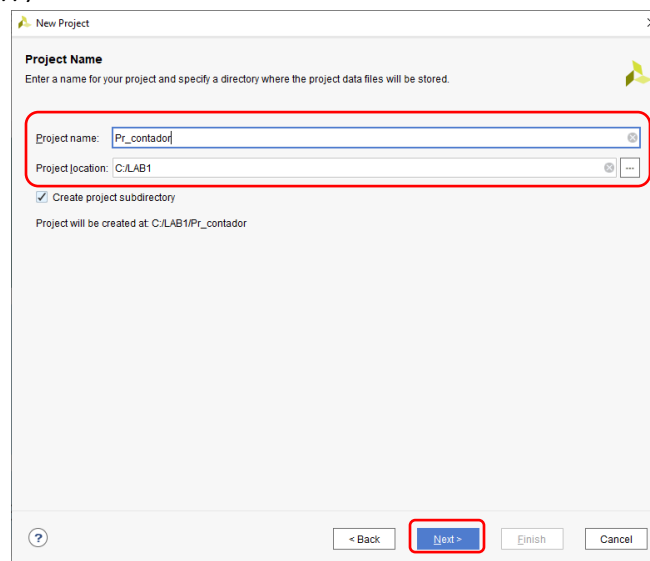
2. Clic en "Create Project"



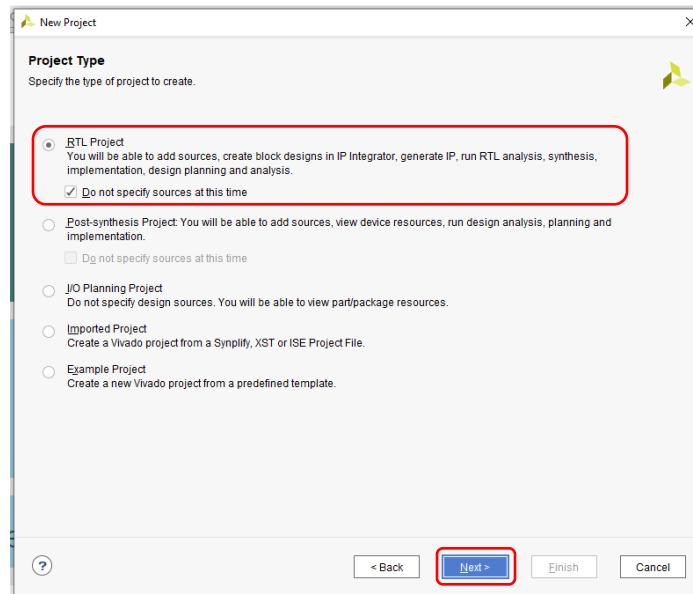
3. Se abre la siguiente ventana y clic en "Next"



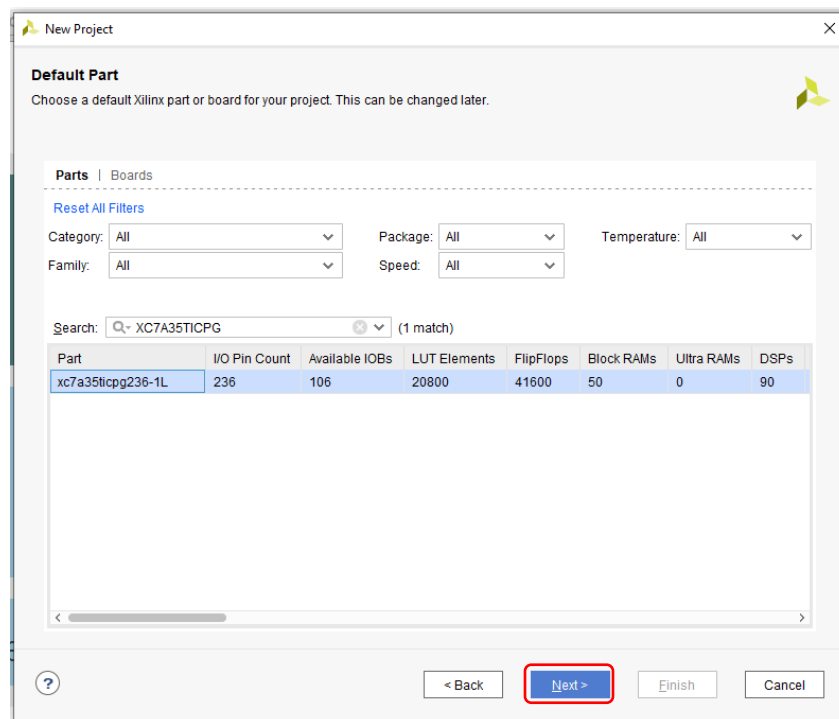
4. Dar un nombre y una ubicación al proyecto, evitar poner punto (.) en el nombre del proyecto. La ubicación debe ser fácil de encontrar preferiblemente en una carpeta del directorio raíz (C://). Clic en “Next”



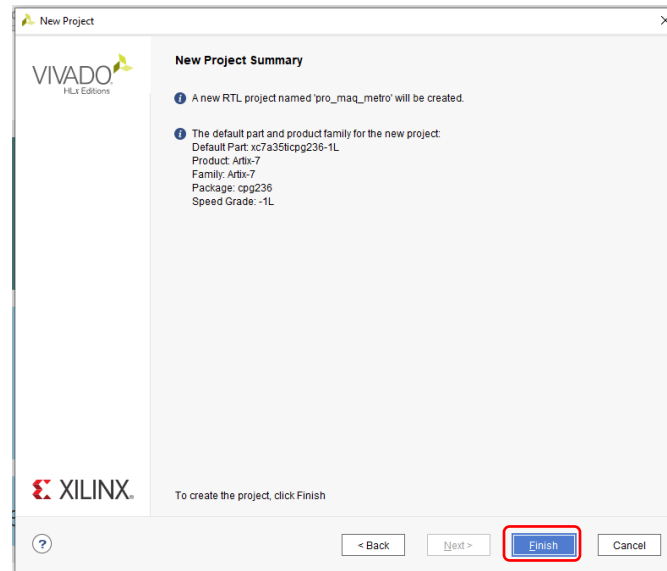
5. Seleccionar “RTL Project”



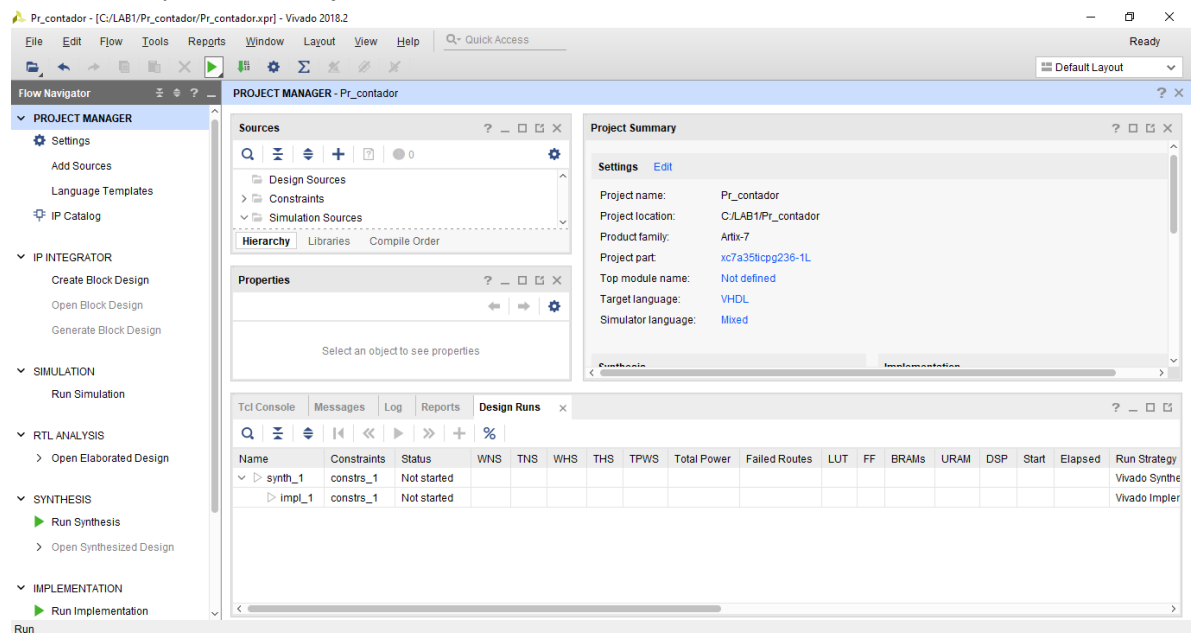
6. Seleccionar la FPGA con la que se va a trabajar, en este caso: xc7a35ticipg236-1L, clic en “Next”



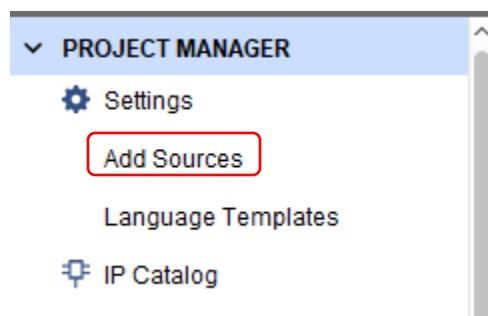
7. Aparece la siguiente ventana que indica que el proyecto se ha creado. Clic en “Finish”



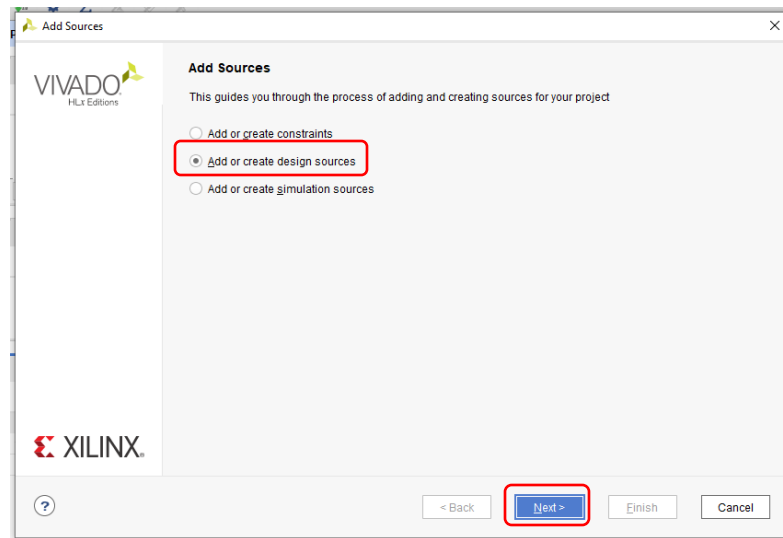
8. Se abre el espacio de trabajo de Vivado



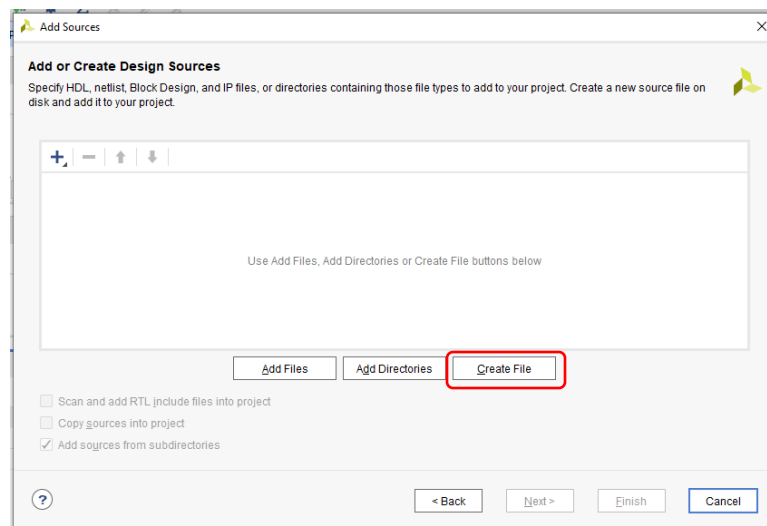
9. A continuación, se agrega el archivo vhd que permitirá generar el divisor de frecuencia, para ello clic en "Add Sources"



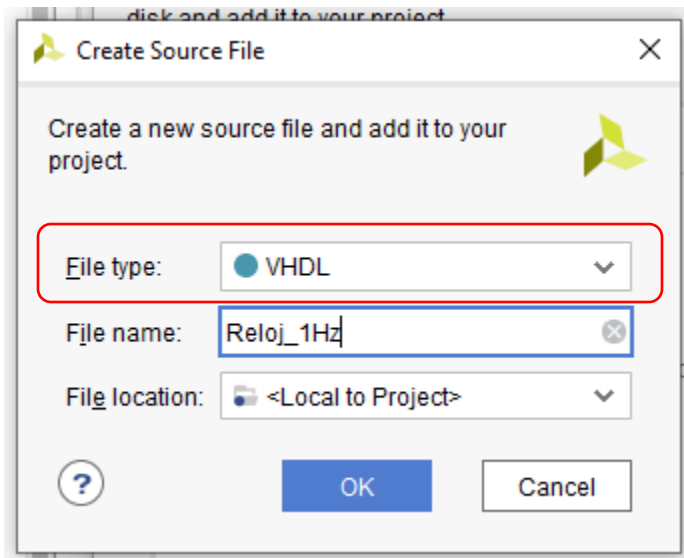
10. Aparece la siguiente ventana, en ella, seleccionar: "Add or create design sources" y clic en "Next"



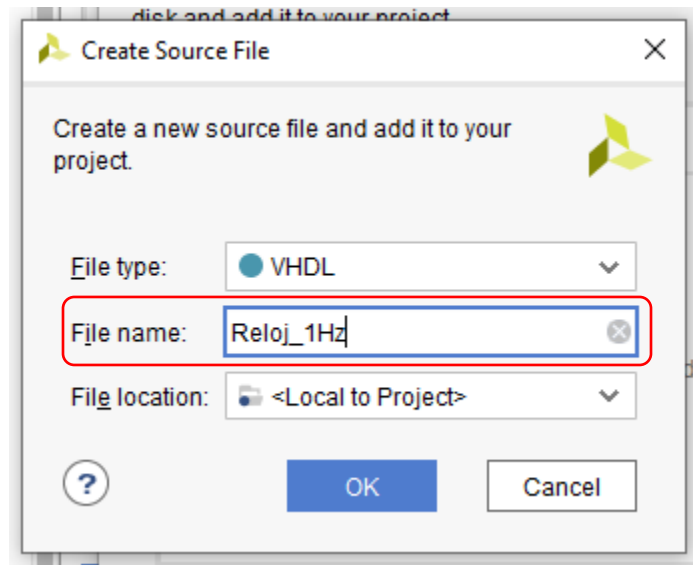
11. En la siguiente ventana clic en: “Create Files”



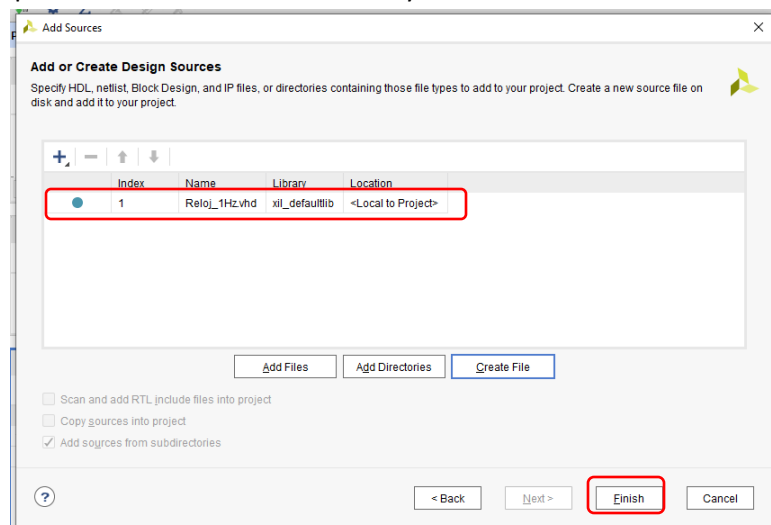
12. En la nueva ventana se selección “VHDL” en File Type



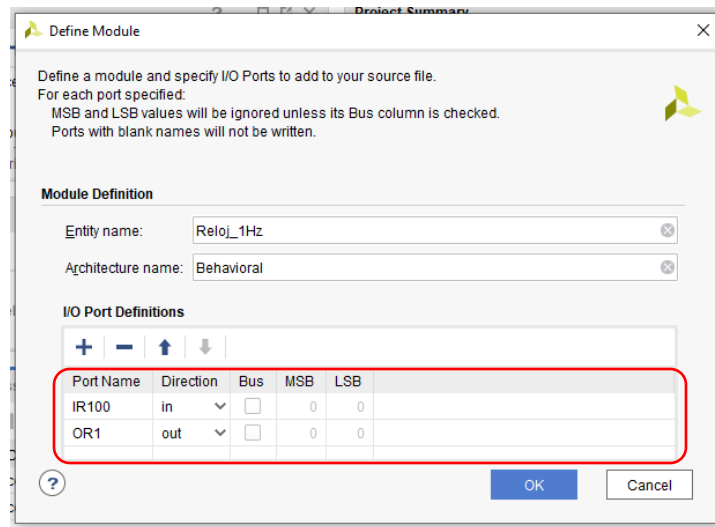
13. Se le asigna un nombre al archivo en *File name* y clic en “Ok”



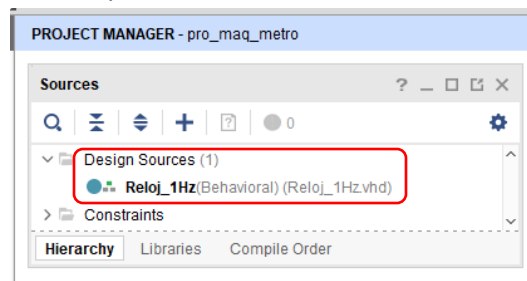
14. Revisar que aparezca el nuevo archivo creado y clic en “Finish”



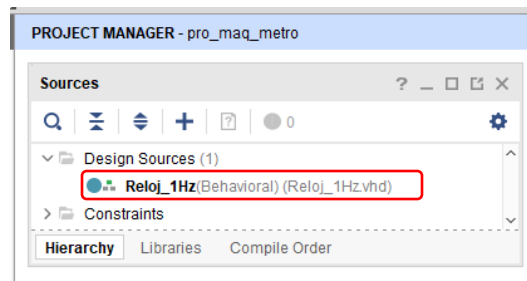
15. A continuación, crear las entradas y salidas que va a tener el divisor de frecuencia, en este caso una entrada tipo bit “IR100” y una salida tipo bit “OR1” y clic en “Ok”



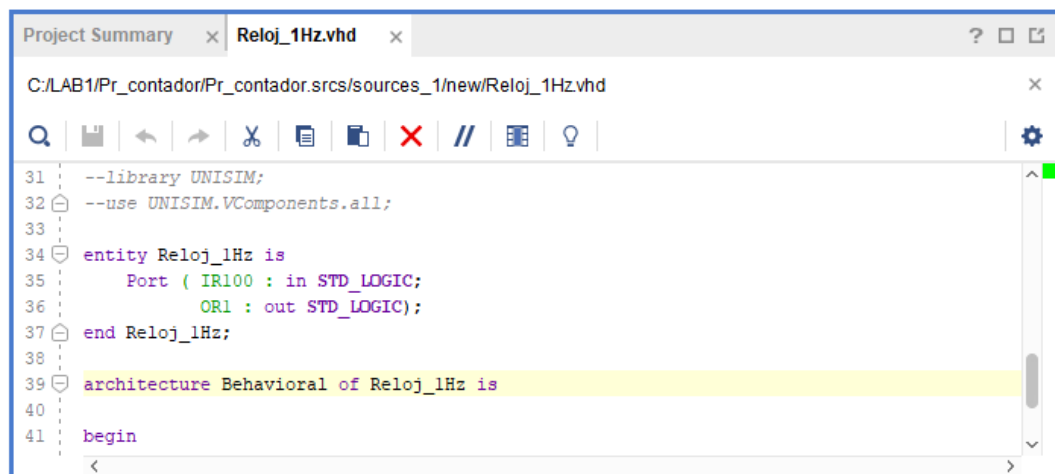
16. El nuevo archivo creado debe aparecer en:



17. Doble clic en el archivo creado



18. Se abre el archivo creado



19. En el archivo creado escribir el siguiente código:

```

22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use IEEE.NUMERIC_STD.ALL;
25
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 --use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity Reloj_1Hz is
36     Port ( IR100 : in STD_LOGIC;
37           OR1 : out STD_LOGIC);
38 end Reloj_1Hz;
39
40 architecture Behavioral of Reloj_1Hz is
41     signal aux : integer range 0 to 100000000 := 0;
42     signal x : STD_LOGIC;
43 begin
44     process (IR100)
45     begin
46         if rising_edge (IR100) then
47             aux <= aux + 1;
48
49             if (aux = 49999999) then
50                 x <= NOT x;
51                 aux <= 0;
52             end if;
53         end if;
54     end process;
55     OR1 <= x;
56 end Behavioral;
```

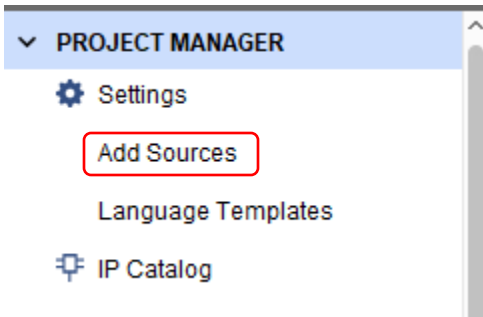
20. Clic en guardar



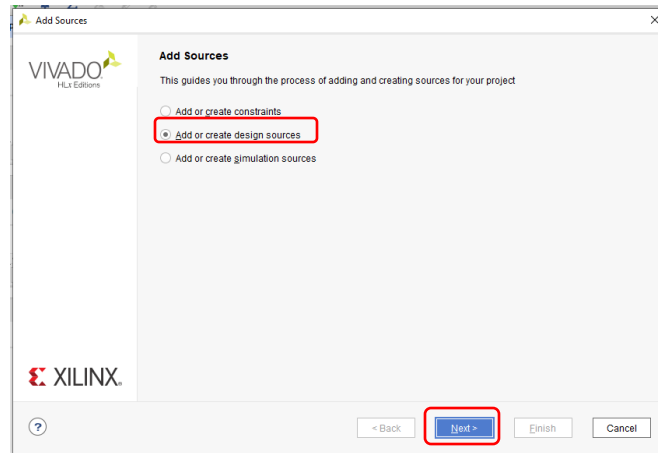
SUBSISTEMA CONTADOR DE 4 BITS

A continuación, se debe crear el archivo VHDL que deberá contener el contador de 4 bits.

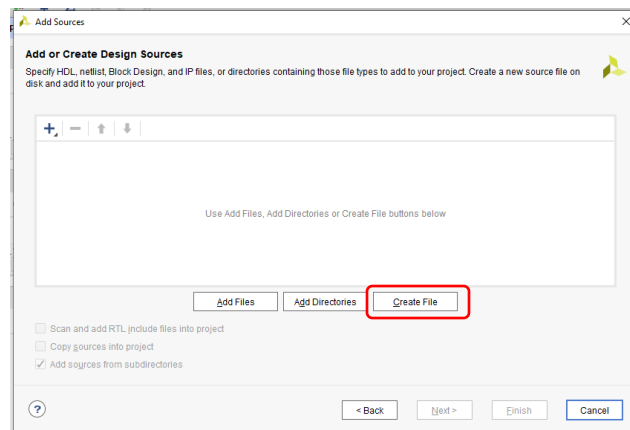
21. Clic en “Add Sources”



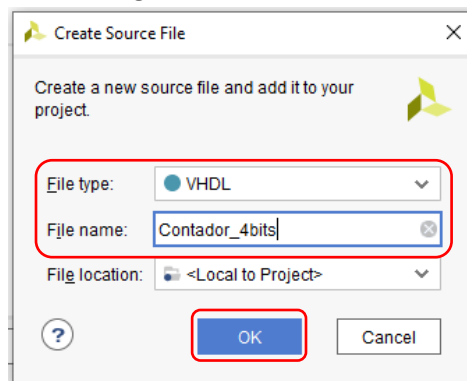
22. Seleccionar “Add or create design sources” y clic en “Next”



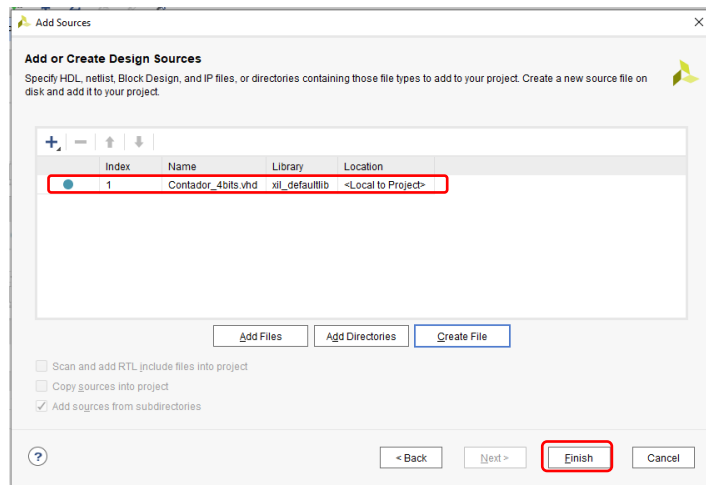
23. Clic en “Create File”



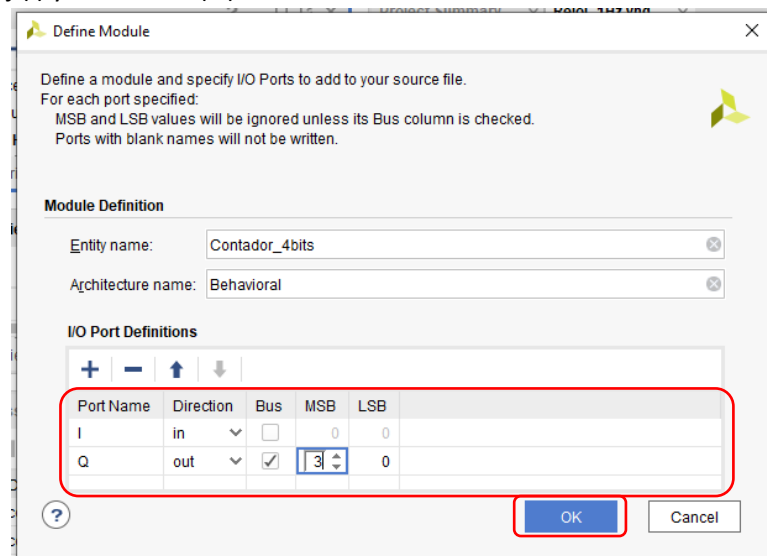
24. Seleccionar archivo tipo “VHDL”, asignar un nombre al nuevo archivo y clic en “OK”



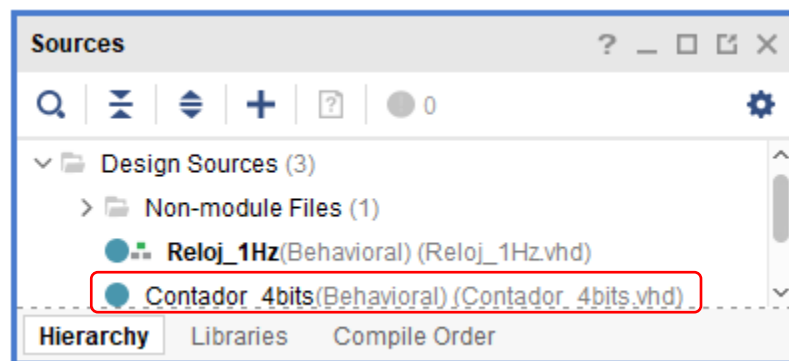
25. El archivo creado deberá aparecer en la siguiente ventana y clic en “Finish”



26. Agregar las entradas y salidas del contador; el contador tiene una entrada de 1 bit que es la señal de reloj (I) y una salida (Q) de cuatro bits. Clic en “OK”



27. El nuevo archivo creado aparece en



28. Doble clic en el nuevo archivo para crear el código en VHDL del contador

```

22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24  Use IEEE.NUMERIC_STD.ALL;
25  use IEEE.STD_LOGIC_UNSIGNED.ALL;
26  -- Uncomment the following library declaration if using...
34
35  entity Contador_4bits is
36      Port ( I : in STD_LOGIC;
37            Q : out STD_LOGIC_VECTOR (3 downto 0));
38  end Contador_4bits;
39  architecture Behavioral of Contador_4bits is
40      signal cont_1: STD_LOGIC_VECTOR (3 downto 0) := "0000";
41  begin
42      process (I)
43      begin
44          if rising_edge (I) then
45              cont_1 <= cont_1 + 1;
46          end if;
47
48      end process;
49
50      Q <= cont_1;
51
52  end Behavioral;
--

```

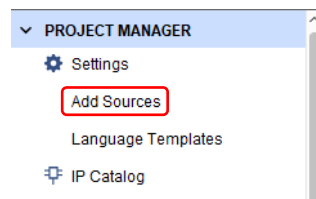
29. Clic en guardar



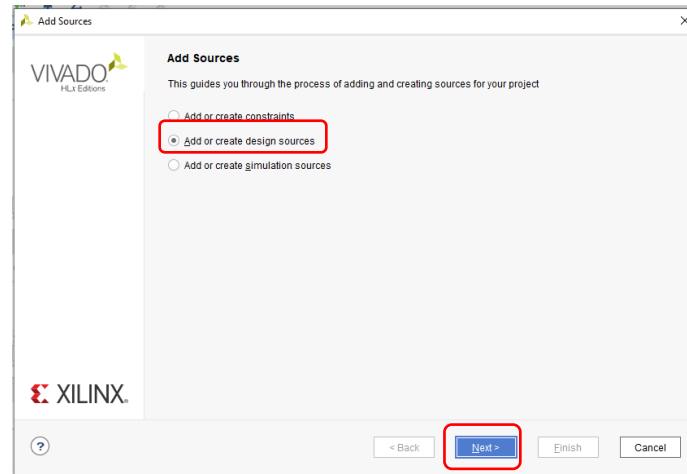
SISTEMA GENERAL

A continuación, se debe crear un archivo que contenga a todos los dos subsistemas creados, ese archivo será nuestro archivo TOP.

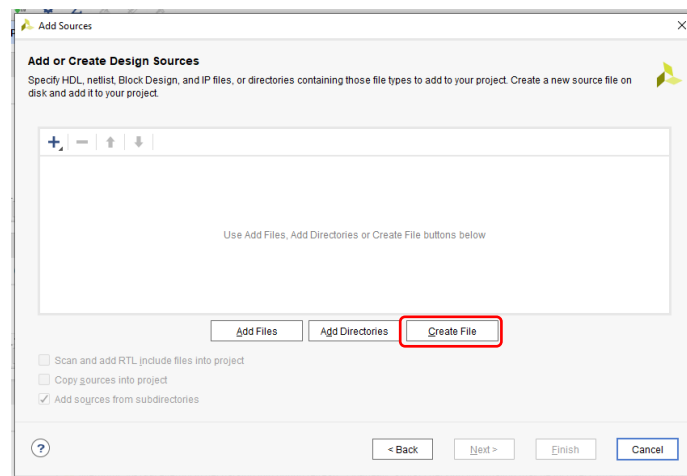
30. Clic en “Add Sources”



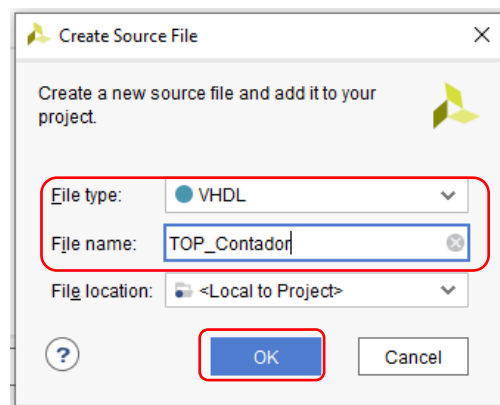
31. Seleccionar y clic en “Next”



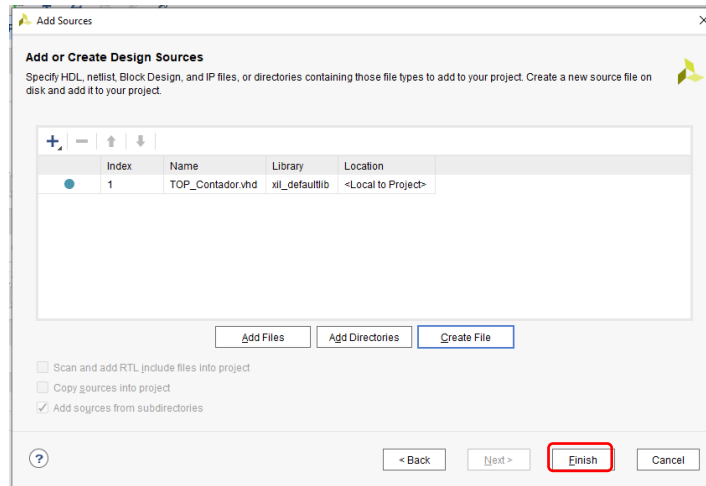
32. Clic en



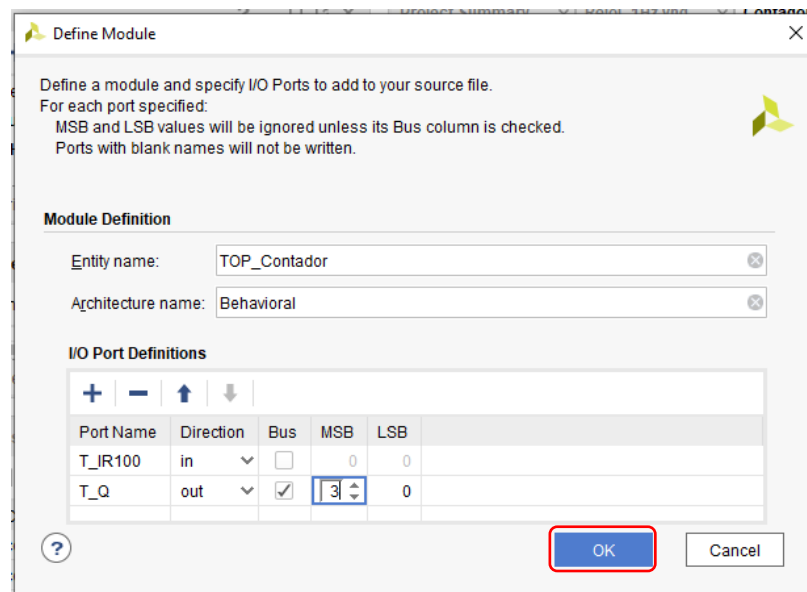
33. Nombre y tipo de archivo



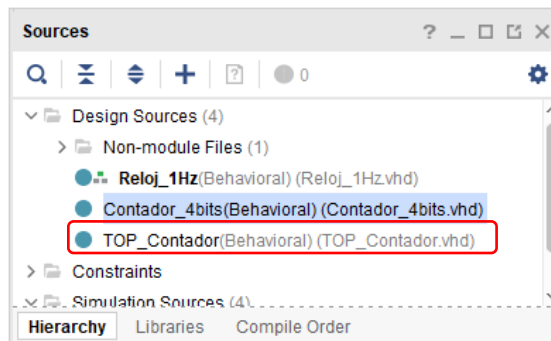
34. Clic en



35. Se crean las salidas y entradas del sistema que va a contener los otros dos subsistemas. Clic en “OK”



36. Aparece un nuevo archivo en



37. Doble clic en ese archivo

38. Se comienzan a agregar la entidad de los otros subsistemas como si fueran componentes del nuevo sistema y se incluye una señal que se llamara: aux.

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 Use IEEE.NUMERIC_STD.ALL;
25 use IEEE.STD_LOGIC_UNSIGNED.ALL;
26
27 -- Uncomment the following library declaration if using...
35
36 entity TOP_Contador is
37     Port ( T_IR100 : in STD_LOGIC;
38           T_Q : out STD_LOGIC_VECTOR (3 downto 0));
39 end TOP_Contador;
40
41 architecture Behavioral of TOP_Contador is
42     component Reloj_1Hz is
43     Port ( IR100 : in STD_LOGIC;
44           OR1 : out STD_LOGIC);
45     end component;
46
47     component Contador_4bits is
48     Port ( I : in STD_LOGIC;
49           Q : out STD_LOGIC_VECTOR (3 downto 0));
50     end component;
51
52     signal aux : STD_LOGIC;
53
54     begin
```

39. Posteriormente se realiza la conexión de los subsistemas y el TOP. El código final quedará de la siguiente manera:

```

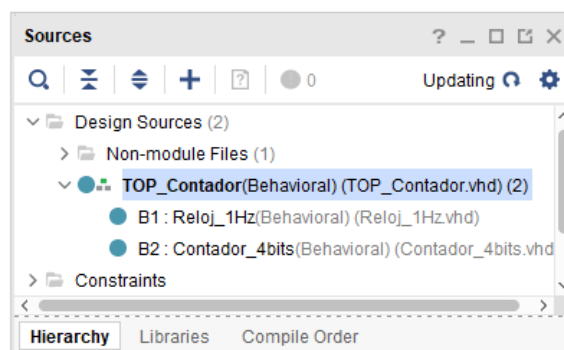
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24  Use IEEE.NUMERIC_STD.ALL;
25  use IEEE.STD_LOGIC_UNSIGNED.ALL;
26
27  -- Uncomment the following library declaration if using...
35
36  entity TOP_Contador is
37      Port ( T_IR100 : in STD_LOGIC;
38            T_Q : out STD_LOGIC_VECTOR (3 downto 0));
39  end TOP_Contador;
40
41  architecture Behavioral of TOP_Contador is
42      component Reloj_1Hz is
43      Port ( IR100 : in STD_LOGIC;
44            OR1 : out STD_LOGIC);
45      end component;
46
47      component Contador_4bits is
48      Port ( I : in STD_LOGIC;
49            Q : out STD_LOGIC_VECTOR (3 downto 0));
50      end component;
51
52      signal aux : STD_LOGIC;
53
54      begin
55          B1: Reloj_1Hz PORT MAP(T_IR100,aux);
56          B2: Contador_4bits PORT MAP(aux,T_Q);
57
58  end Behavioral;
59

```

40. Clic en guardar



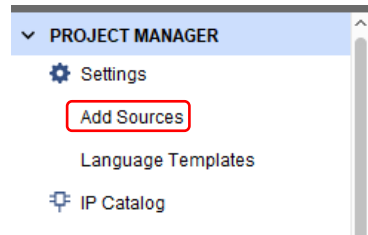
41. Observar el nuevo aspecto de



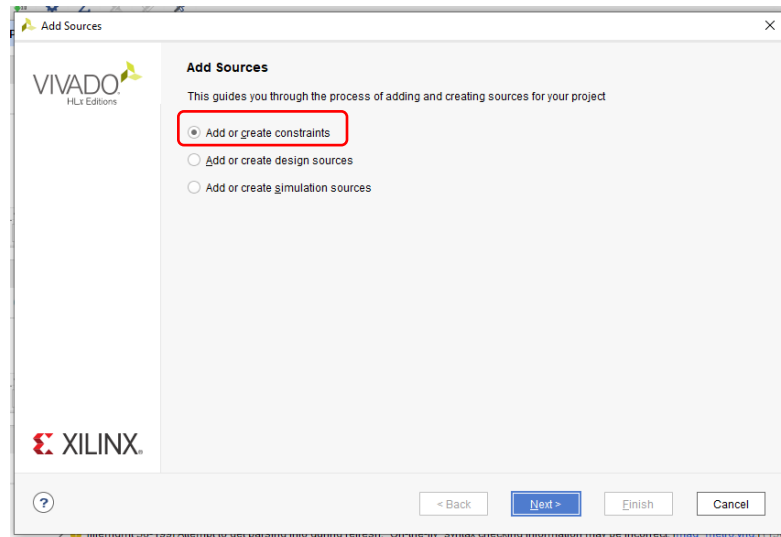
CONEXIÓN DE LAS ENTRADAS Y SALIDAS DEL SISTEMA DESARROLLADO CON LAS ENTRADAS Y SALIDAS DE LA BASYS 3

Una vez creado el archivo de programación lo que viene a continuación es la conexión de la entrada y las salidas del circuito con la señal de reloj y los leds del sistema de desarrollo Basys3

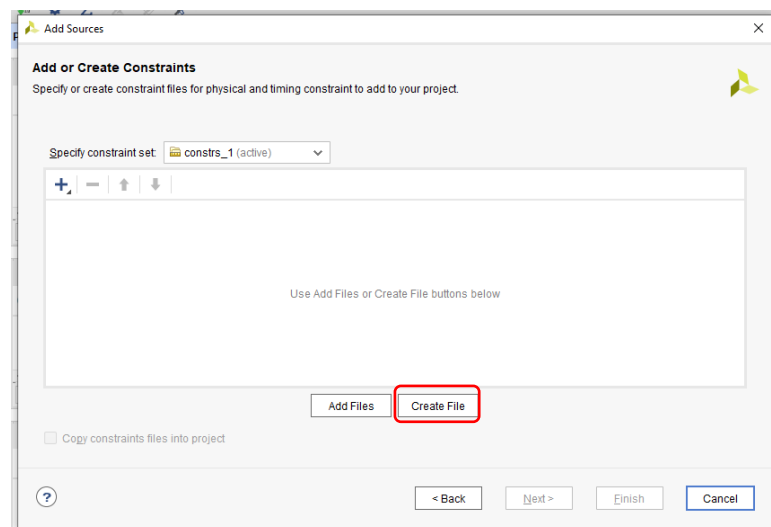
42. Clic en



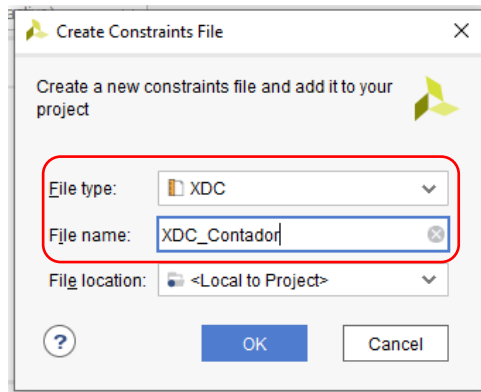
43. Seleccionar “Add or create constraints” y clic en “Next”



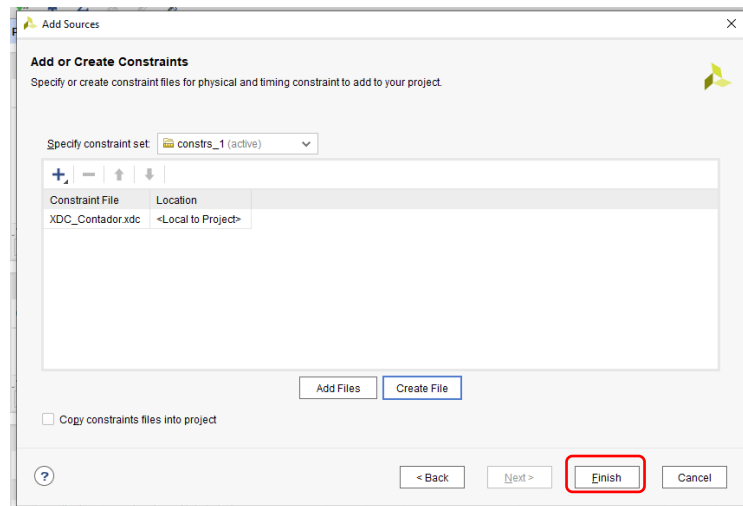
44. Clic en “Create File”



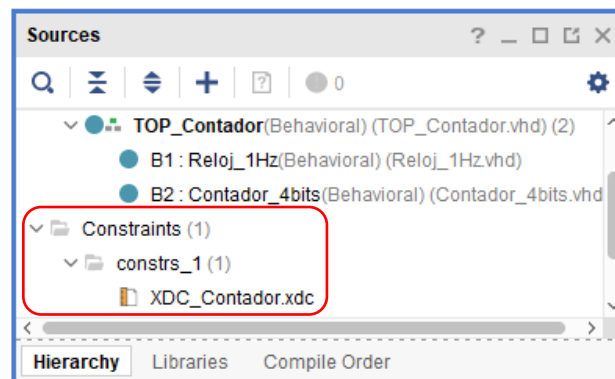
45. Se le asigna un nombre y clic en “OK”



46. Clic en “Finish”



47. El nuevo archivo creado aparece en



48. Doble clic sobre el nuevo archivo creado y es posible editarlo

49. Para editar el archivo creado se debe ingresar a la siguiente dirección:

https://github.com/Digilent/Basys3/blob/master/Projects/XADC_Demo/src/constraints/Basys3_Master.xdc

Copiar las entradas, salidas y señal de reloj que se necesitan y pegarlos en el nuevo archivo creado.

50. El archivo creado deberá tener el siguiente aspecto:

```

1 # Clock signal
2 set_property PACKAGE_PIN W5 [get_ports CLK100MHZ]
3 set_property IOSTANDARD LVCMOS33 [get_ports CLK100MHZ]
4 create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLK100MHZ]
5
6 # LEDs
7 set_property PACKAGE_PIN U16 [get_ports {LED[0]}]
8 set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
9 set_property PACKAGE_PIN E19 [get_ports {LED[1]}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
11 set_property PACKAGE_PIN U19 [get_ports {LED[2]}]
12 set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
13 set_property PACKAGE_PIN V19 [get_ports {LED[3]}]
14 set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]

```

51. Reemplazar en el nuevo archivo creado el nombre de las entradas y salidas por los nombres usados en el proyecto

```

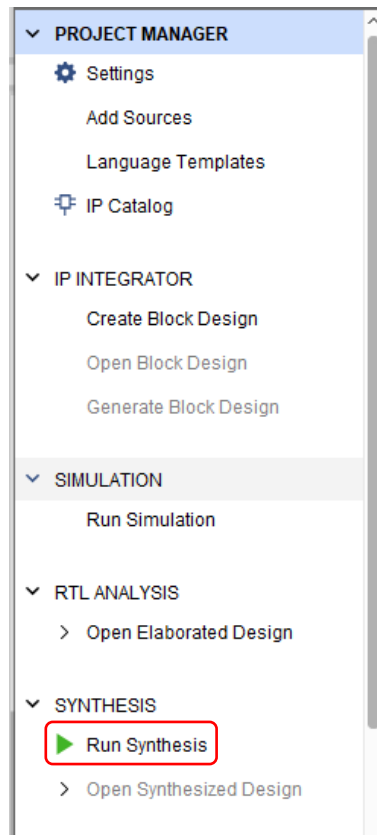
1 Clock signal
2 t_property PACKAGE_PIN W5 [get_ports T_IR100]
3 set_property IOSTANDARD LVCMOS33 [get_ports T_IR100]
4 create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports T_IR100]
5
6 LEDs
7 set_property PACKAGE_PIN U16 [get_ports {T_Q[0]}]
8 set_property IOSTANDARD LVCMOS33 [get_ports {T_Q[0]}]
9 set_property PACKAGE_PIN E19 [get_ports {T_Q[1]}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {T_Q[1]}]
11 set_property PACKAGE_PIN U19 [get_ports {T_Q[2]}]
12 set_property IOSTANDARD LVCMOS33 [get_ports {T_Q[2]}]
13 set_property PACKAGE_PIN V19 [get_ports {T_Q[3]}]
14 set_property IOSTANDARD LVCMOS33 [get_ports {T_Q[3]}]

```

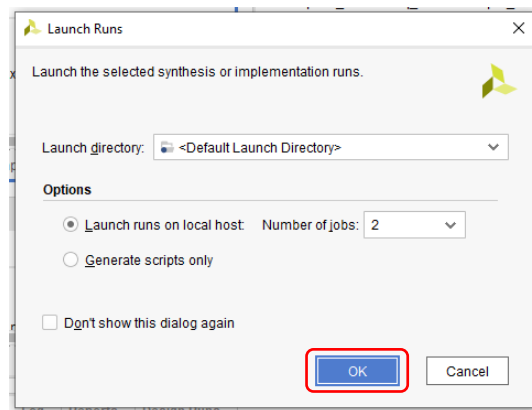
52. A continuación, clic en guardar



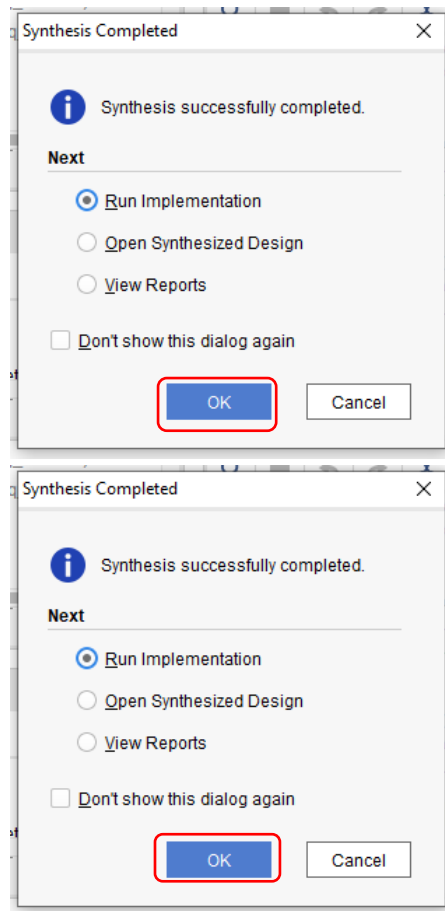
53. Y luego clic en “Run Synthesis”



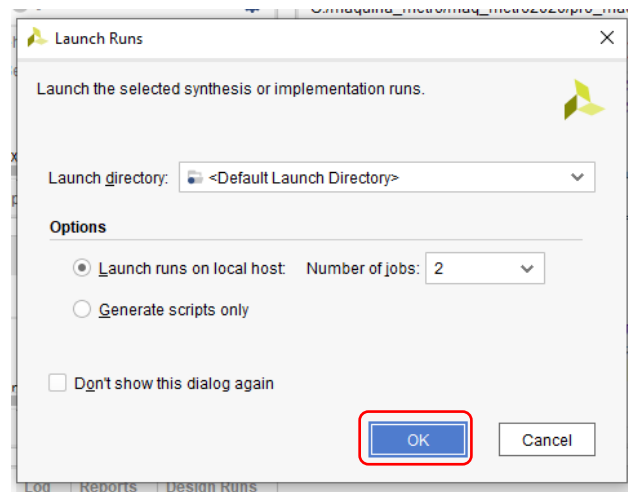
54. Clic en “OK”



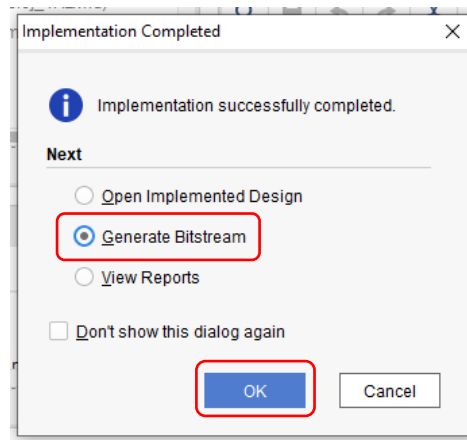
55. Esperar, una vez terminado clic en “OK”



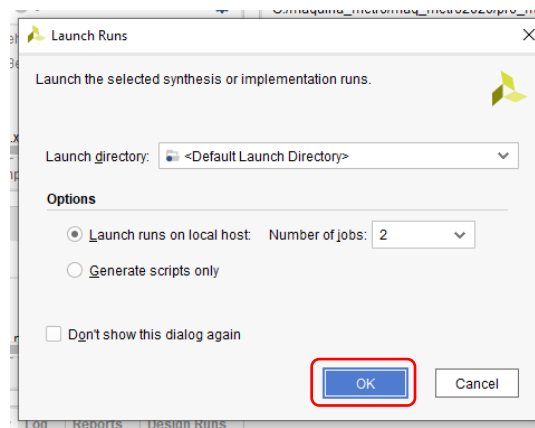
56. Clic en "OK"



57. Seleccionar "Generate Bitstream" y clic en OK



58. Clic en “OK”



PROGRAMACIÓN

59. Proceder con la programación igual como se hizo en el circuito combinacional

COMPROBAR FUNCIONAMIENTO

EJERCICIO

Diseñar un contador de 0000 a 9999, mostrar el resultado en los cuatro displays 7 segmentos que tiene la Basys 3