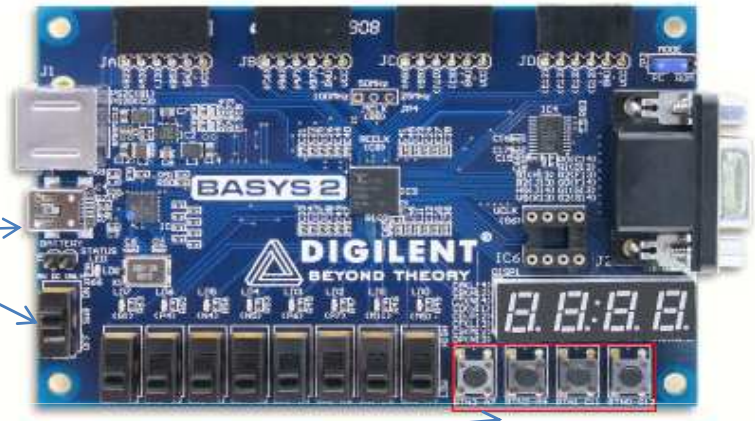
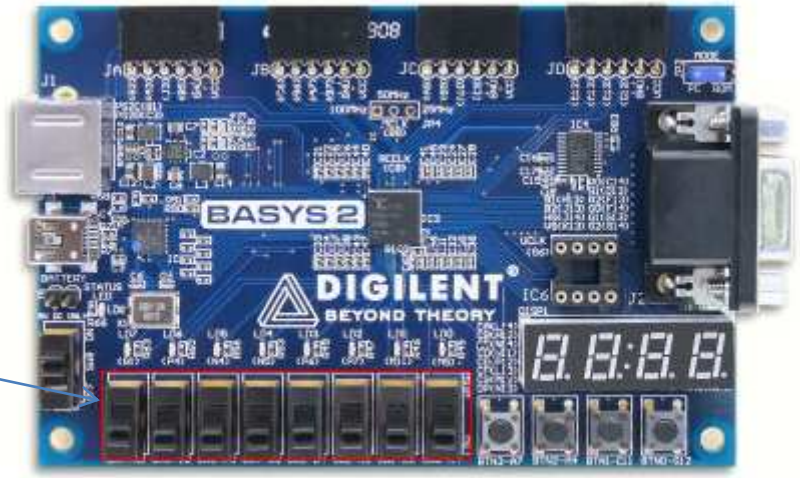
Week 5

I/O on the BASYS 2 Board

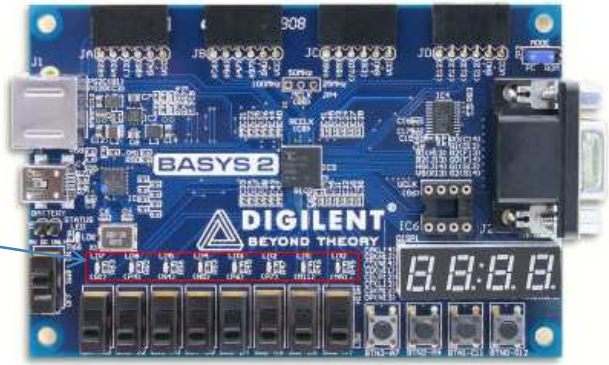
- **Power Supply**
  - Switch Selectable (SW8)
  - Options
    - USB
    - External (3.5VDC - 5.5VDC Power Supply)
- **Pushbuttons**
  - Four Pushbuttons
  - Normally low
  - Driven high when button is pressed
  - Short Circuit Protection Provided
    - Connecting FPGA pin as output could result in short circuit
    - Remedied with resistor
  - Refer to page 4 of *BASYS 2 Reference Manual*



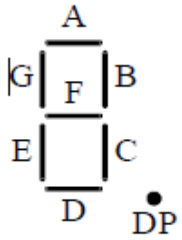
- **Slider Switches**
  - Eight Slider Switches
    - Constant high or low signal
    - Short Circuit Protection Provided
      - Connecting FPGA pin as output could result in short circuit
      - Remedied with resistor
    - Refer to page 4 of *BASYS 2 Reference Manual*

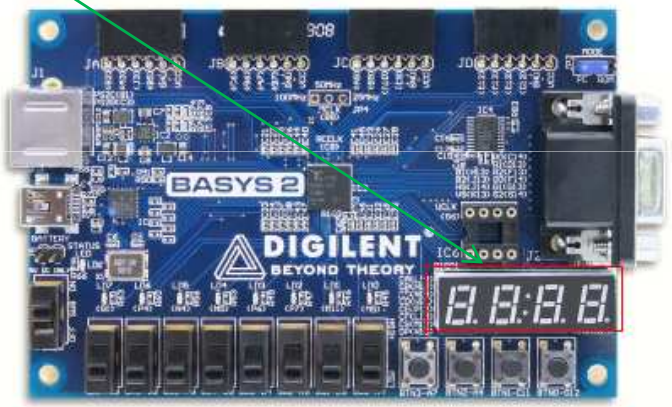


- **LEDs**
  - Eight LEDs
  - Anodes driven from FPGA
    - Logic 1 illuminates LED
  - Cathodes connected to ground via resistor
  - Refer to page 4 of *BASYS 2 Reference Manual*



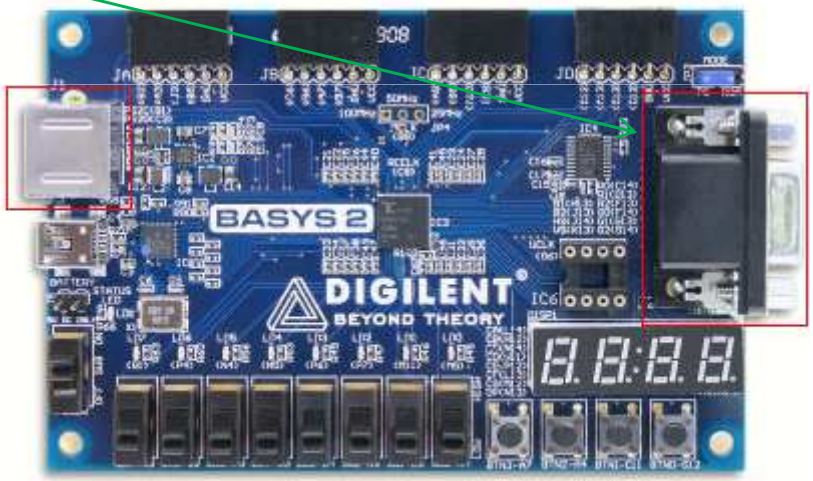
- **Seven-Segment Displays**
  - Four Seven-Segment Displays
  - Common Anode
  - Enabling Digits
    - AN0…AN3
    - Active Low
    - When enabled, cathodes (CA, CB, …, CF, CG, DP) can be driven
    - Low signal illuminates segment
  - Illuminating multiple digits with different values
    - All digits can APPEAR continuously illuminated if driven once every 1 to 16 ms (1 KHz to 60 Hz)
  - Refer to pages 4-5 of *BASYS 2 Reference Manual*

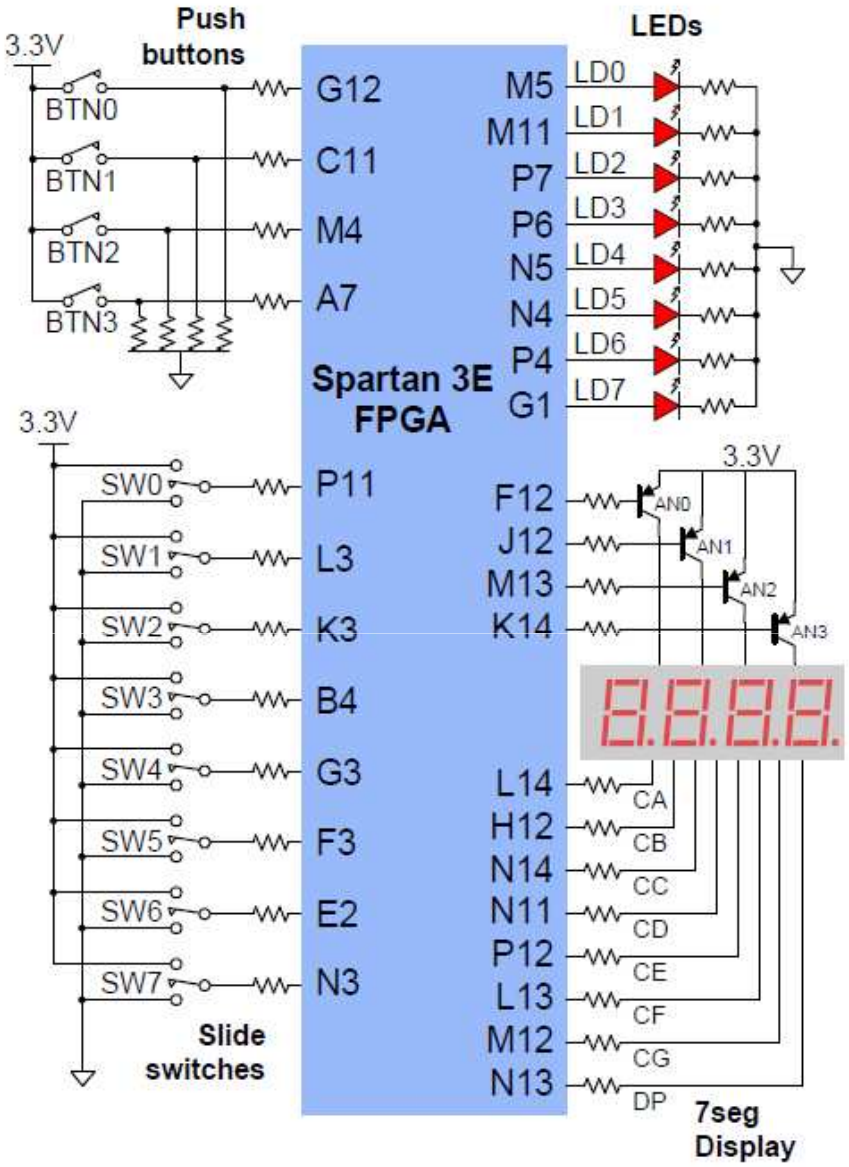




- **Oscillators**
  - Primary Oscillator
    - User selectable (soldering required)
      - 25, 50, or 100 MHz
      - Refer to page 3 of *BASYS Reference Manual*
    - CLK1
    - Pin B8 of FPGA
  - Secondary Oscillator
    - Socket provided
    - CLK2

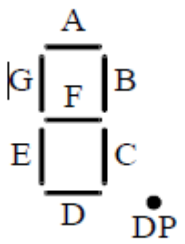
- **PS/2 &VGA Port**
  - PS/2 & VGA Ports Available





Basys2 I/O circuits

To display Number “1” on First Segment , then we need to have:
- AN1 to be activated which means “0” (active low)
- CB, CC are activated which means “0” and every other pins to be inactivated which means “1”



Managing a Large Project

- **Managing a Large Project**
  - Tasks
  - Functions
  - Modules

- Tasks
  - Can enable other tasks & functions
  - May execute in non-zero simulation time
  - May contain delay, event, or timing control statements
  - May have zero or more arguments
    - Type
      - Input
      - Output
      - Inout
  - Can pass values, but do NOT return a value
  - Syntax
    - Definition

```
task Task_Name;
input, Output, & Local Variable List
        begin
            body
        end
endtask
```

    - Call

```
Task_Name(Argument_List);
```

- Functions
  - Can enable another function
    - Not a task
  - Execute in zero simulation time
  - Can NOT contain
    - A delay event
    - Timing control statements
  - Must have at least one input argument
  - Always return a single value
    - Output & InOut arguments NOT allowed
  - Syntax
    - Definition

```
function Function_Name; // Note name may represent a vector
Input & Local Variable List
begin
    Body - Note the output must be assigned a value
end
endfunction
```

    - Call

```
Function_Name(Argument_List);
```

Tasks & Functions

- Local to the module
- They can contain
    - Local variables
    - Registers
    - Time Variables
    - Integers
    - Reals
    - Events
- Cannot contain
    - Wires
    - Always Blocks
    - Initial Blocks
- Behavioral only!

Modules

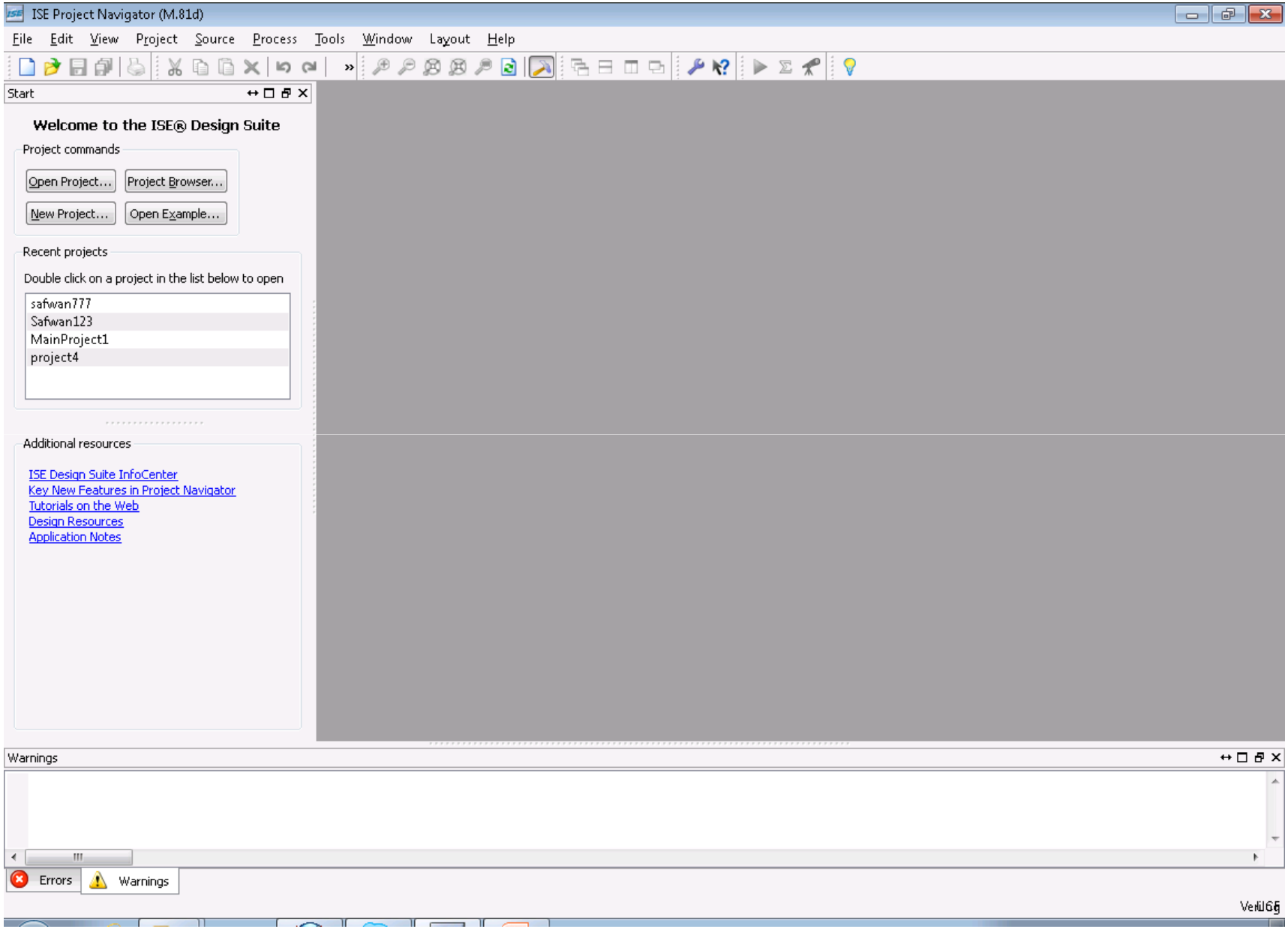
- Instantiating modules can help make code easier to write, modify, read, and debug

Strobe Example

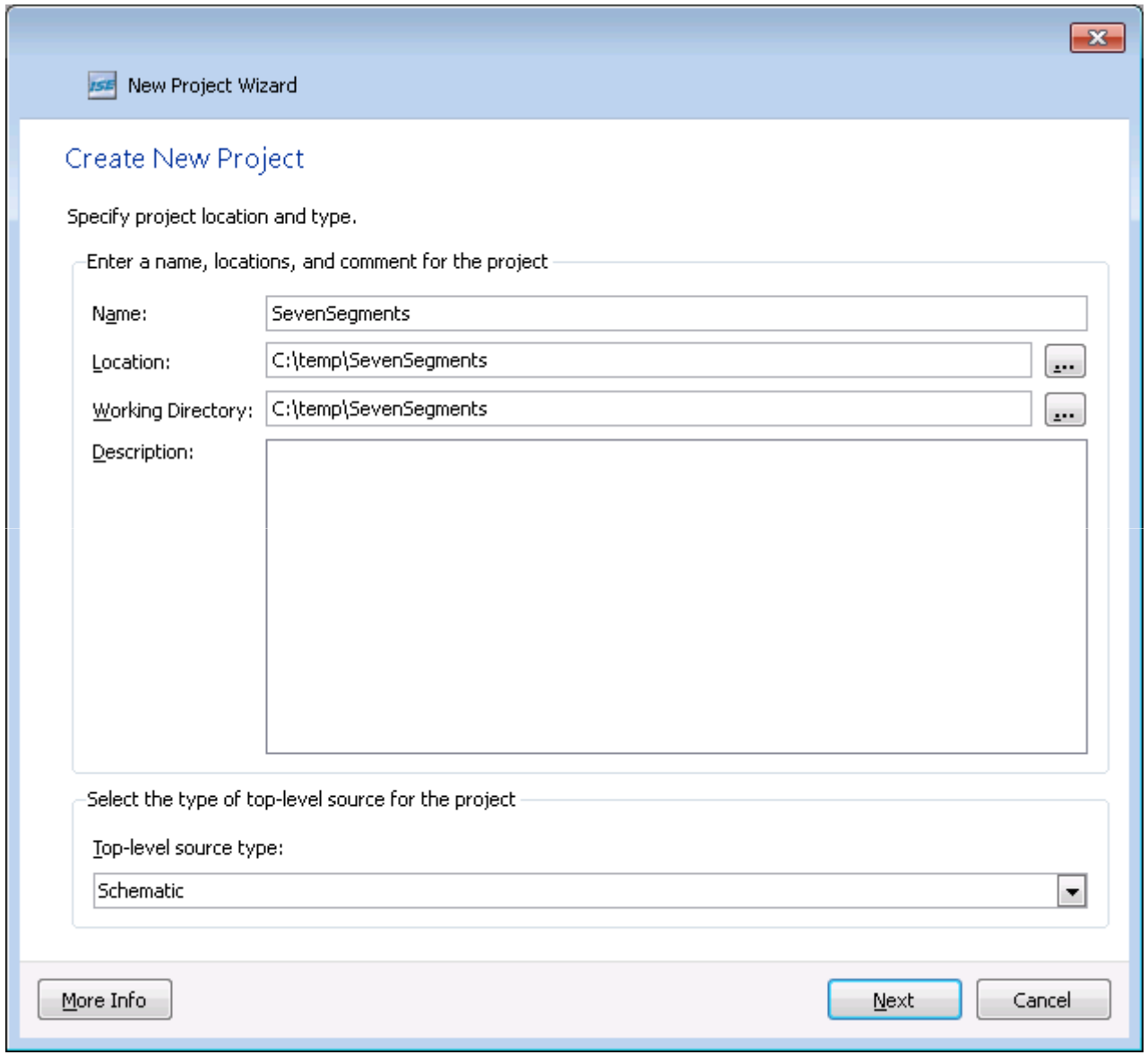
- Description
  - The number ‘12’ is displayed on the middle two digits of the 7-segment display
  - LED blinks at the frequency used to strobe the 7-segment display
  - Reset
    - Switch 0
    - When low, the 7-segment display is disabled
    - When high, ‘12’ is displayed on the 7-segment display
  - Pin Assignments
    - Reset (P11)
    - Clock (B8)
    - LD0 (M5)
    - Anodes 1…4 (F12, J12,M13, K14)
    - Seg A…G (L14, H12, N14, N11, P12, L13, M12)

- *In this example we are going to download Nonvolatile Designs to the FPGA PROM on the BASYS 2 Board*
- *If you are going to Print your design on PROM you **don't** need to change the clock to JTAG*

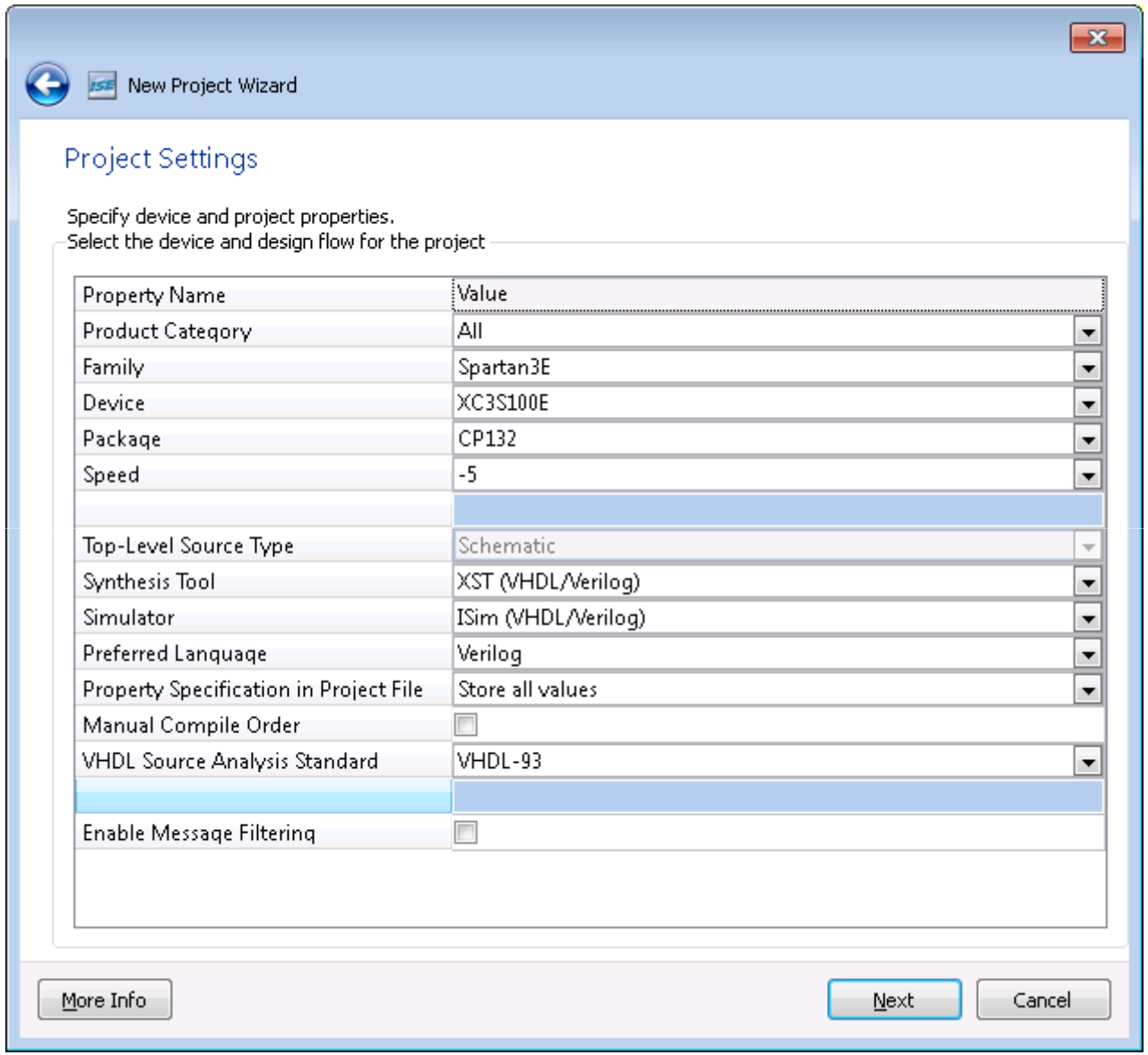
- Open the ISE project Navigator



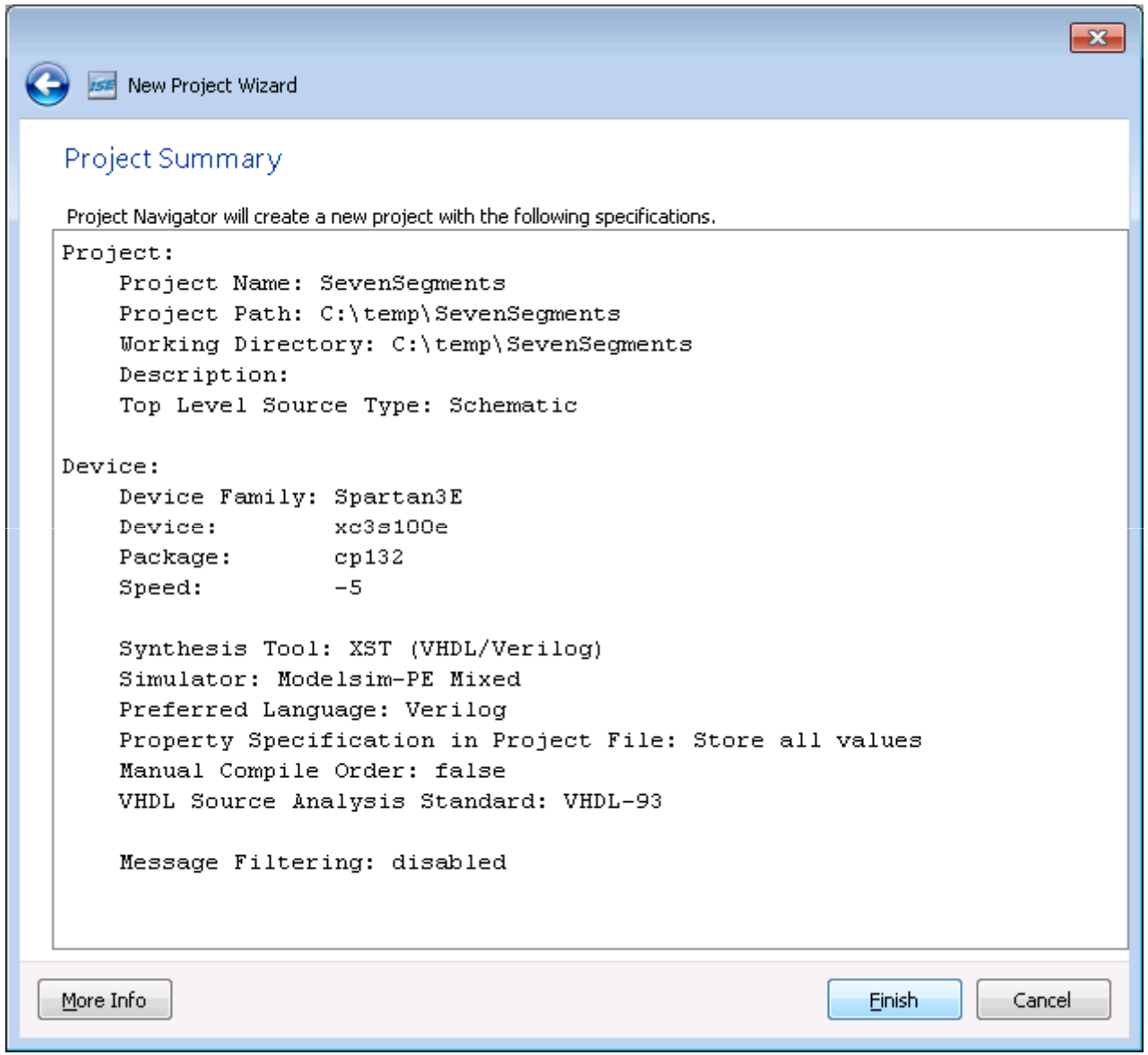
- Type the project name



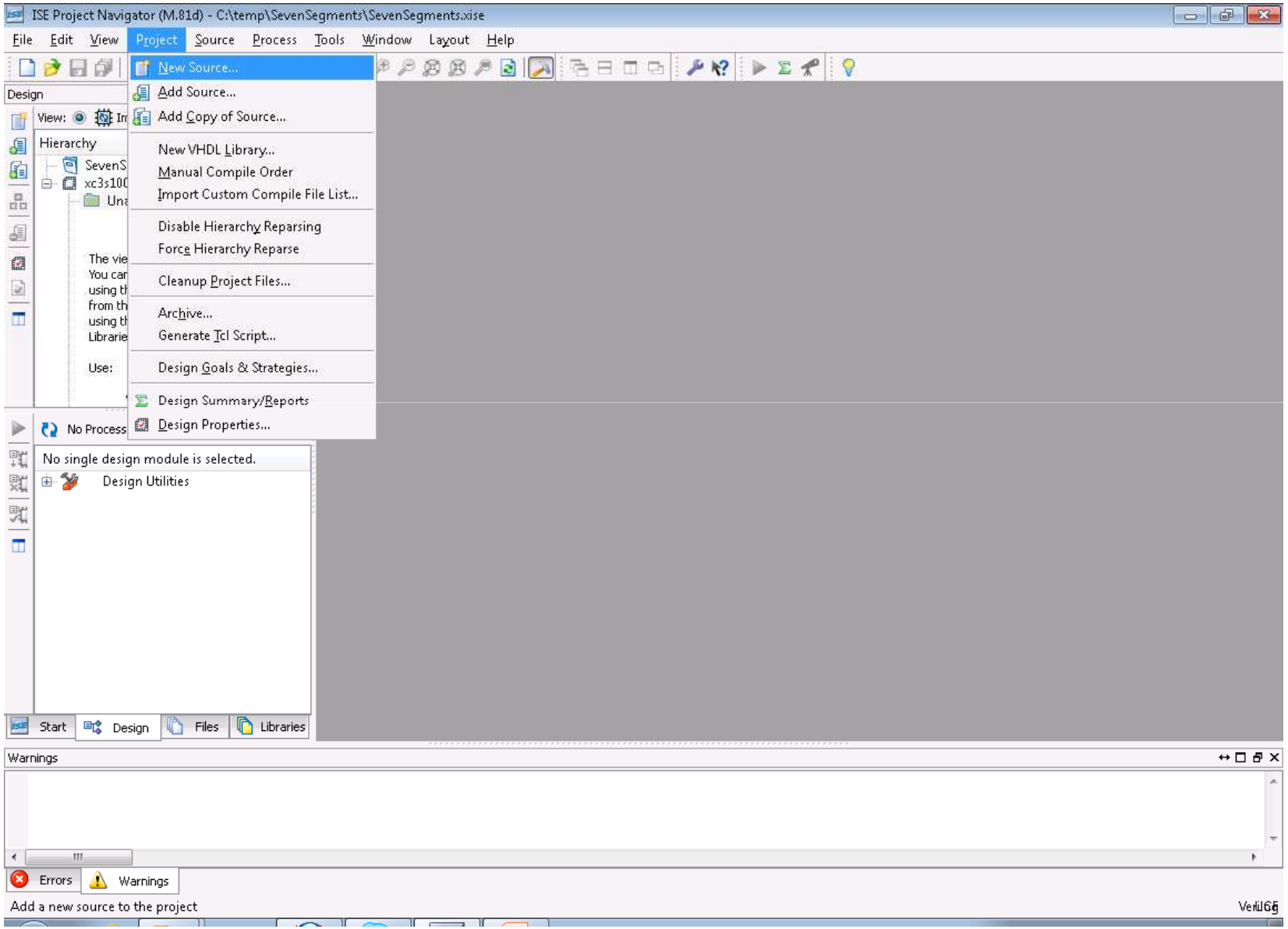
- Select these options then click next



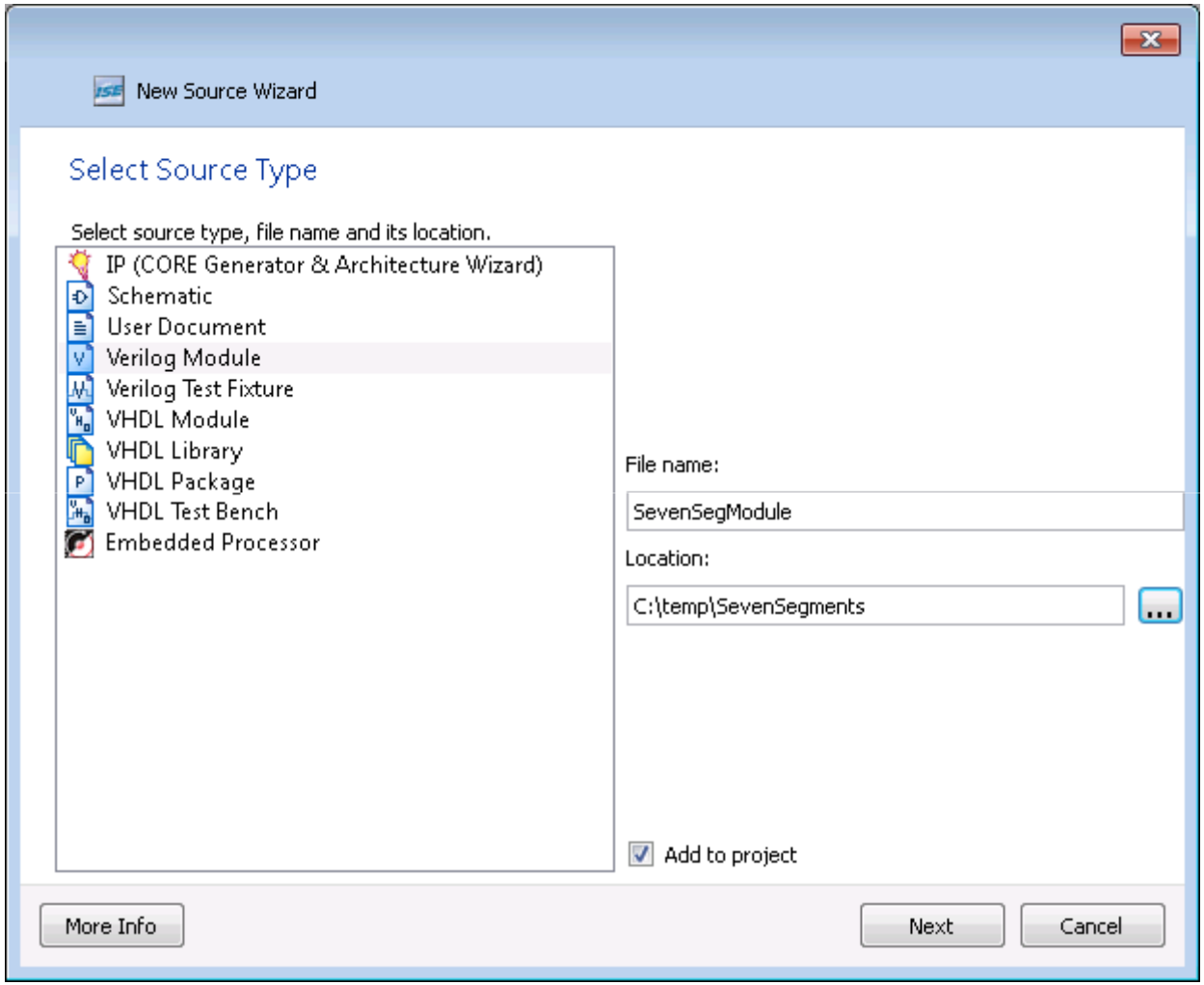
- click finish



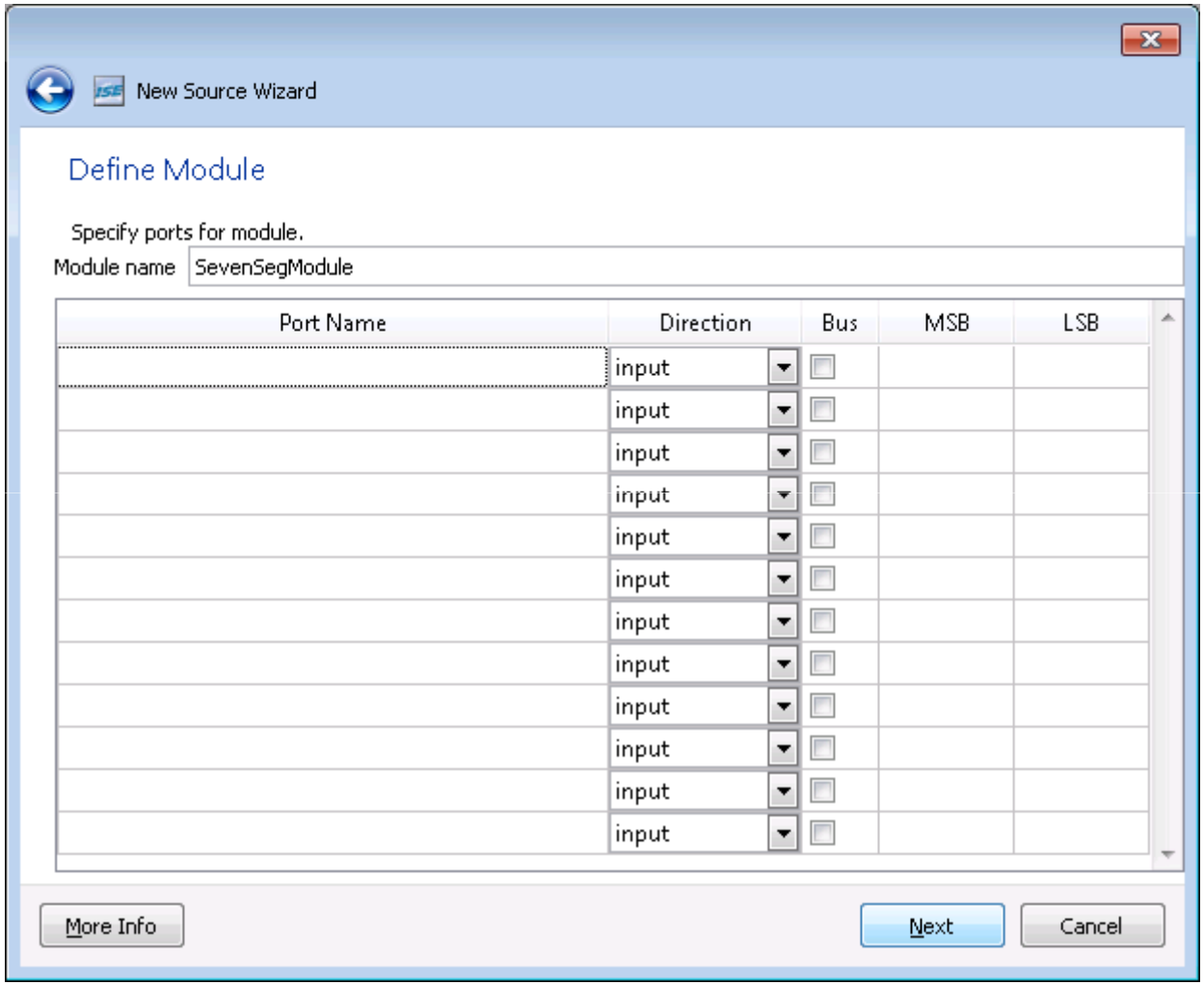
- Select Project →new source



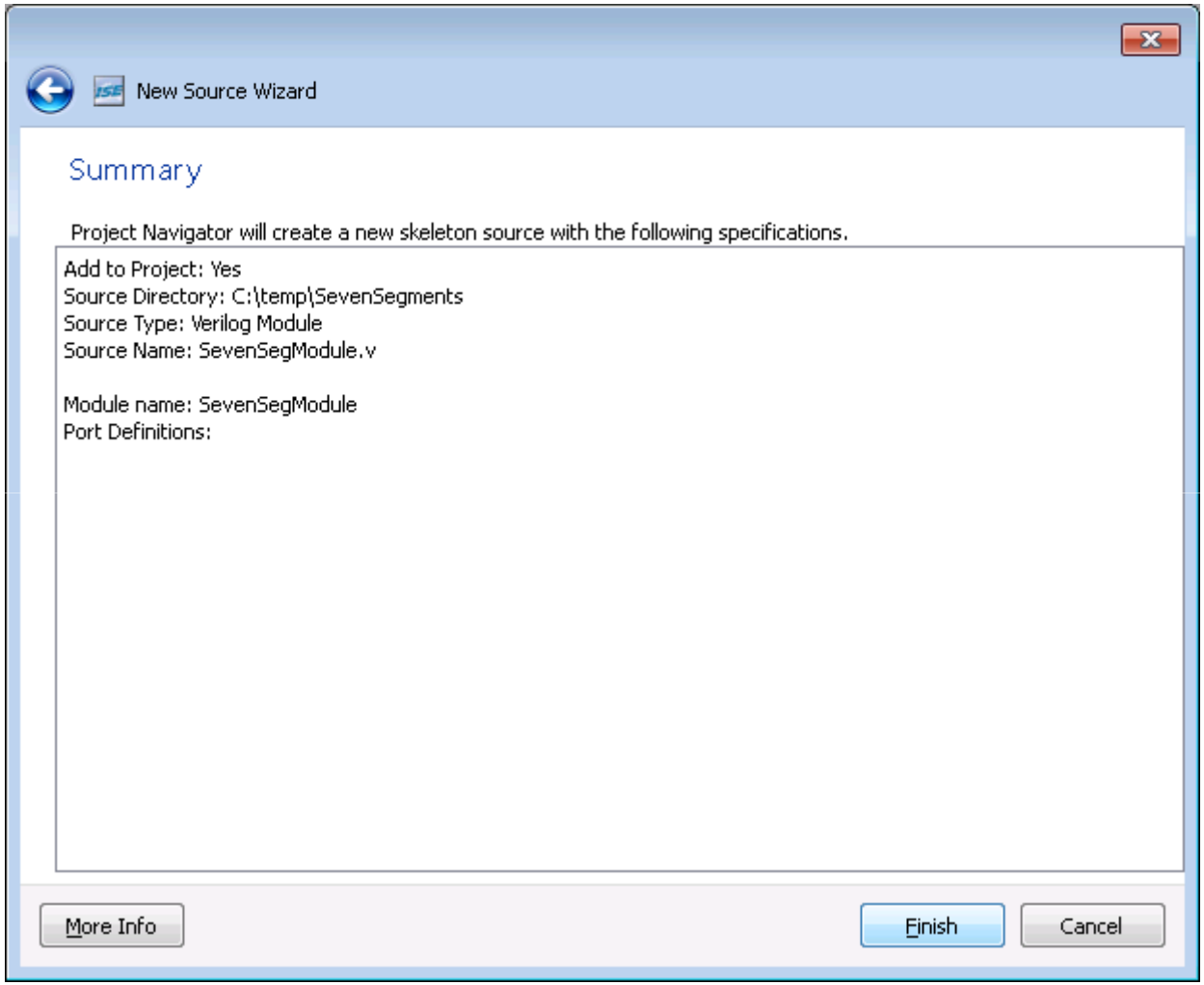
- Select Verilog module and write the file name, then click next



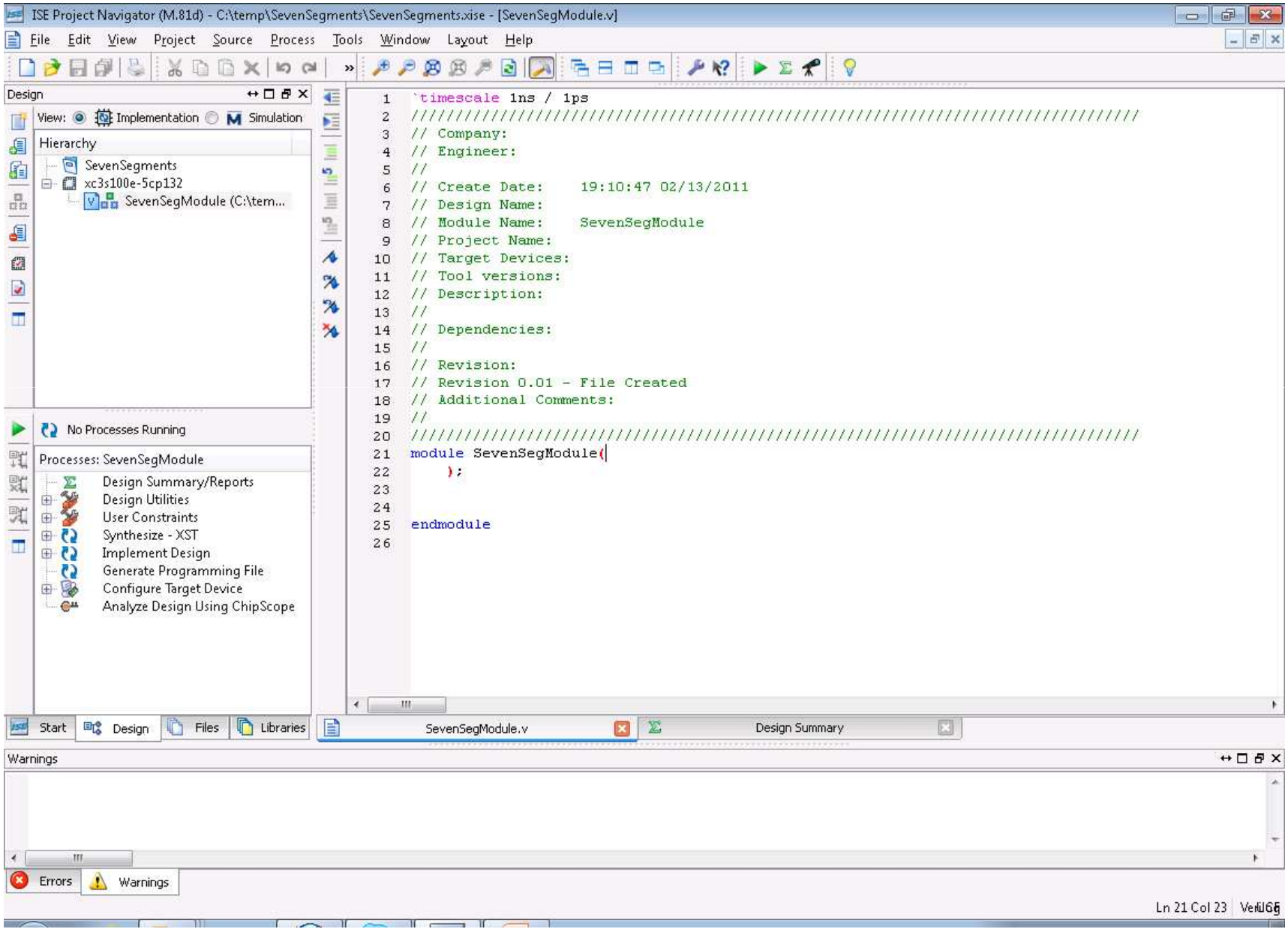
- click next


New Source Wizard
Define Module
Specify ports for module.
Module name
SevenSegModule
Port Name
Direction
Bus
MSB
LSB
input
More Info
Next
Cancel

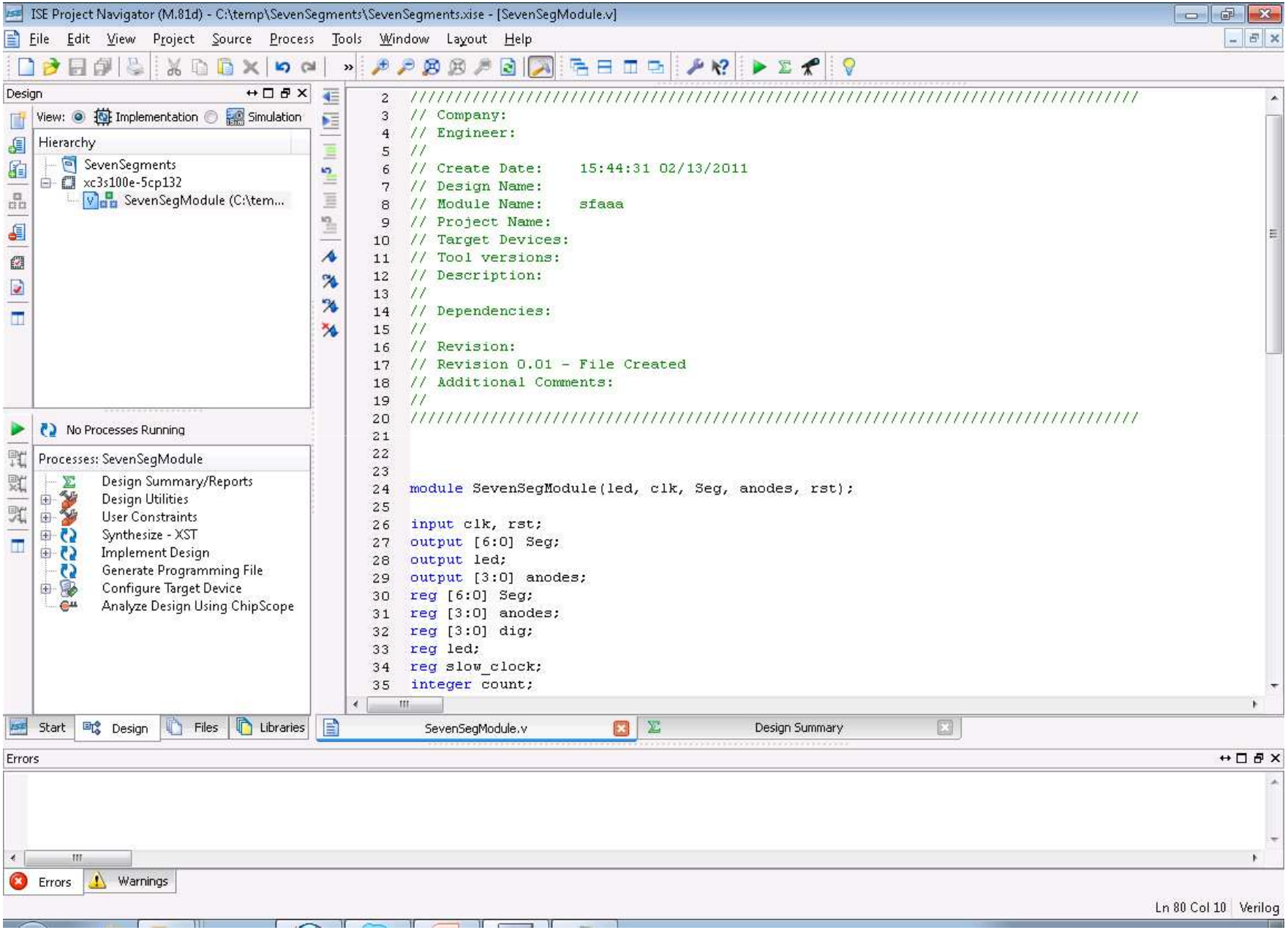
- Click Finish



- The file will be created and it will look like as below



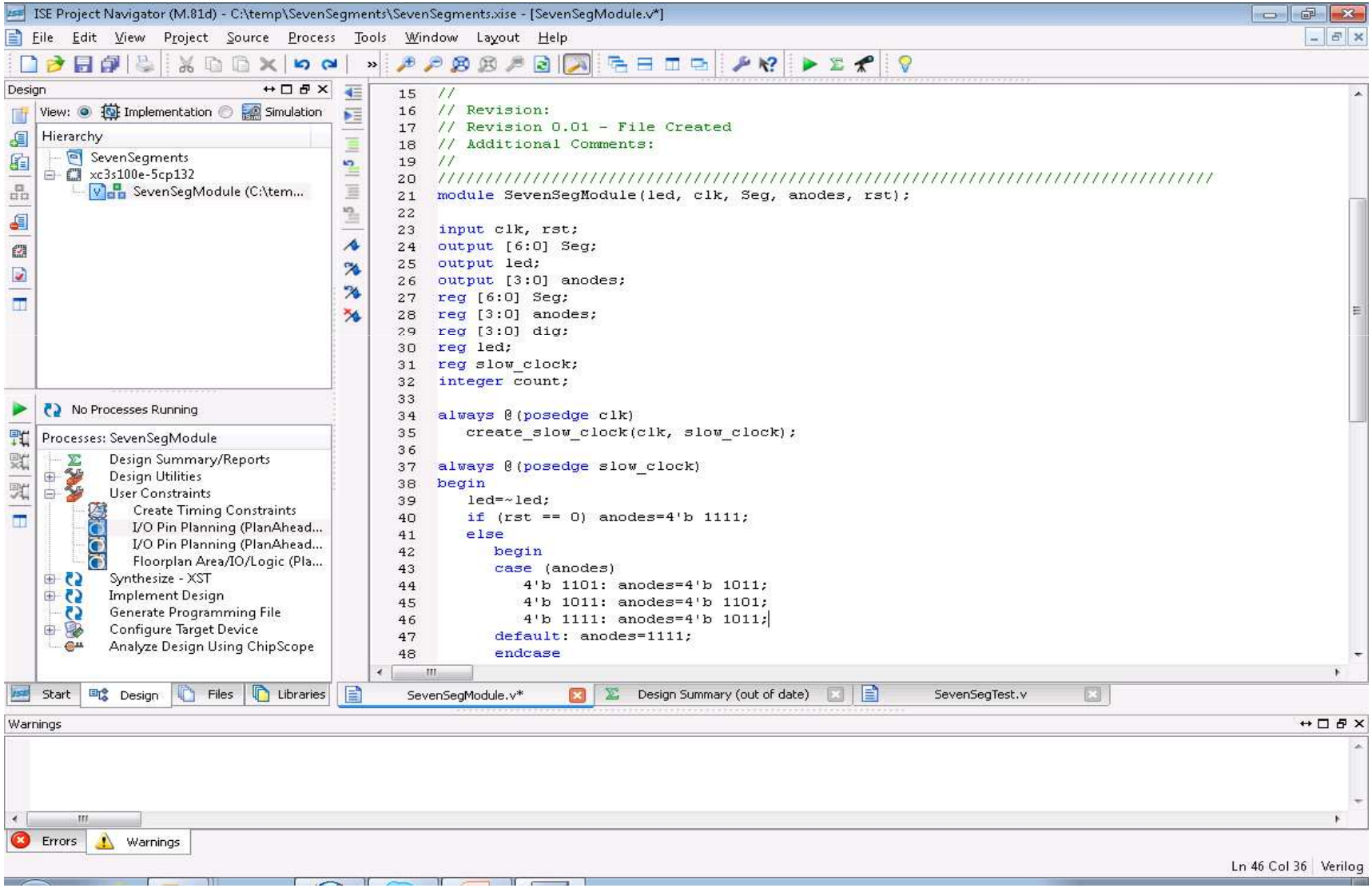
- Add your code



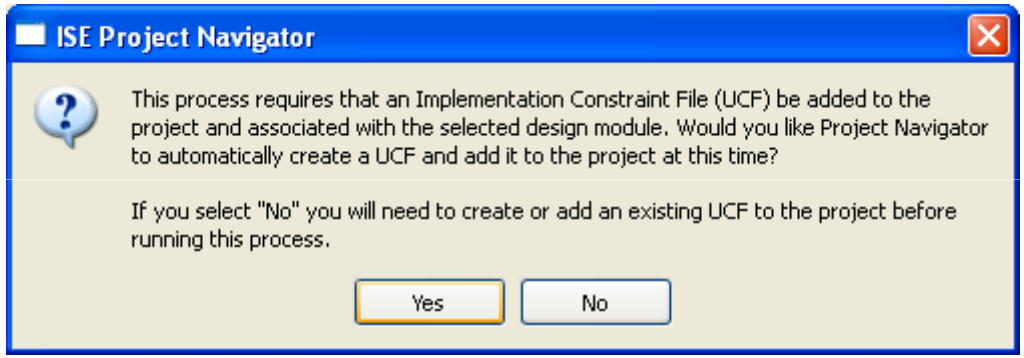
```
module SevenSegModule(led, clk, Seg, anodes, rst);

input clk, rst;
output [6:0] Seg;
output led;
output [3:0] anodes;
reg [6:0] Seg;
reg [3:0] anodes;
reg [3:0] dig;
reg led;
reg slow_clock;
integer count;

always @(posedge clk)
		create_slow_clock(clk, slow_clock);

always @(posedge slow_clock)
begin
led=~led;
if (rst == 0) anodes=4'b 1111;
else
		begin
		case (anodes)
				4'b 1101: anodes=4'b 1011;
				4'b 1011: anodes=4'b 1101;
```

```
4'b 1111: anodes=4'b 1011;
                default: anodes=1111;
                endcase
                case (anodes)
                                4'b 1011: dig=1;
                                4'b 1101: dig=2;
                endcase
                                case (dig)
                                                1: Seg = 7'b 1111001;
                                                2: Seg = 7'b 0100100;
                                endcase

end
end

task create_slow_clock;
                input clock;
                inout slow_clock;
                integer count;

                begin
                if (count > 250000)
                begin
                                count=0;
                                slow_clock = ~slow_clock;
                end
                                count = count+1;
                end
endtask
endmodule
```

-To implement the code on the fpga board go to implementation mode
-- Select circuit.v file and select I/O planning (Planhead-Presynthesis) to open Planahead application to assign the fpga I/O ports to the code I/O



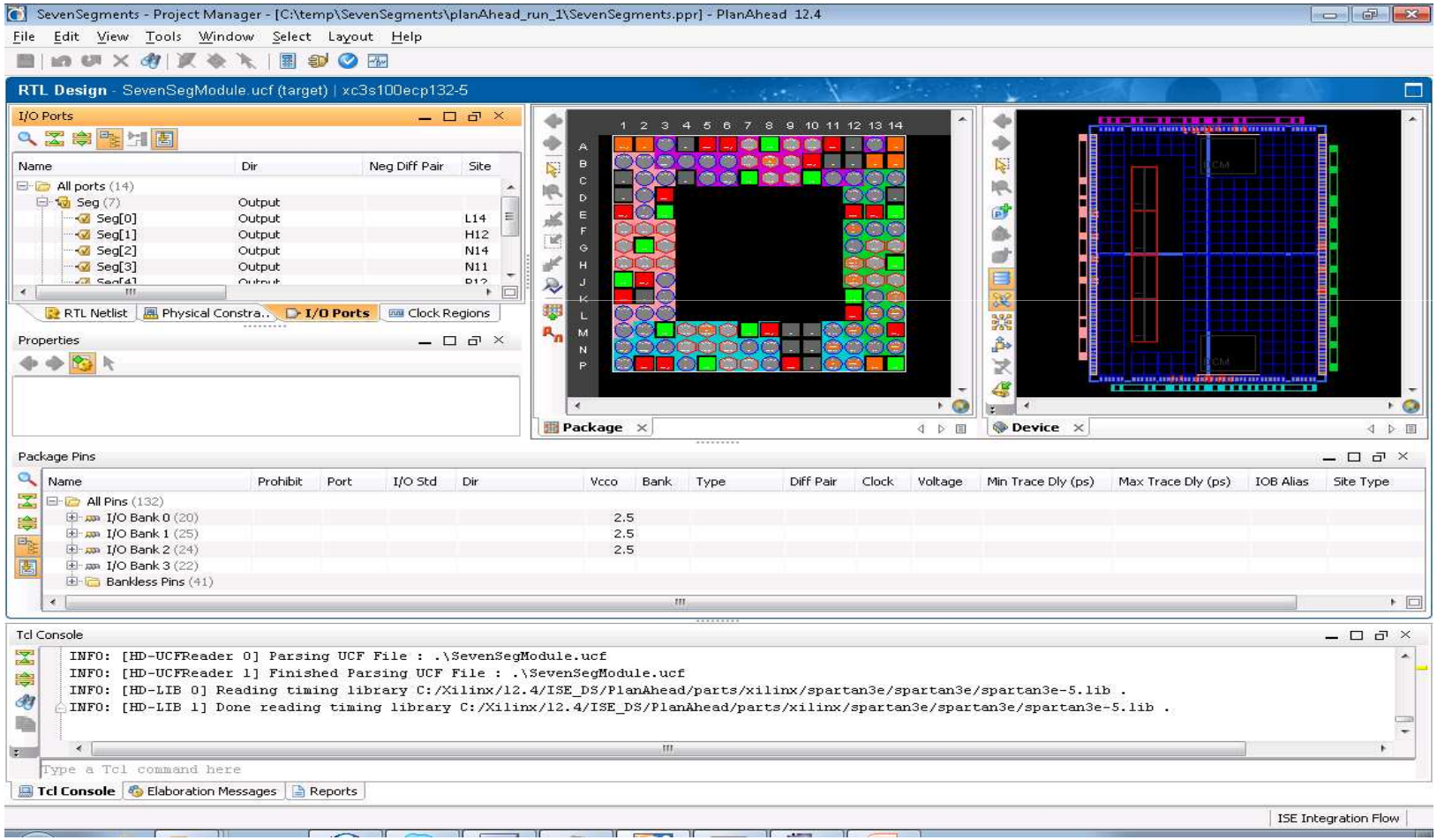
- Click yes to create UCF file and open Planahead application



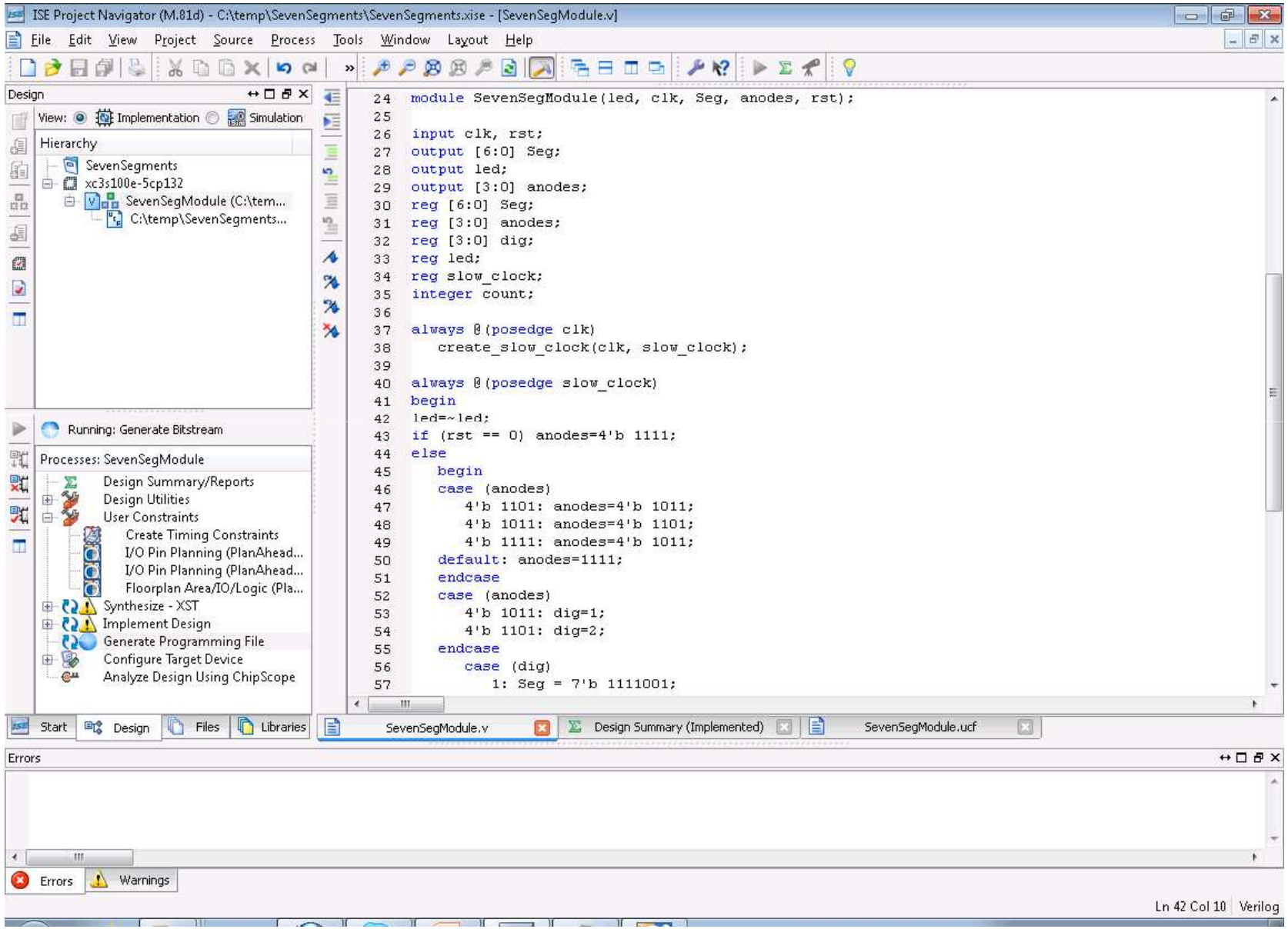
-Planahead Application after its opened

-Select I/O ports, drag and drop each of them to Pin in the Package figure, refer to http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf to find the pin definition

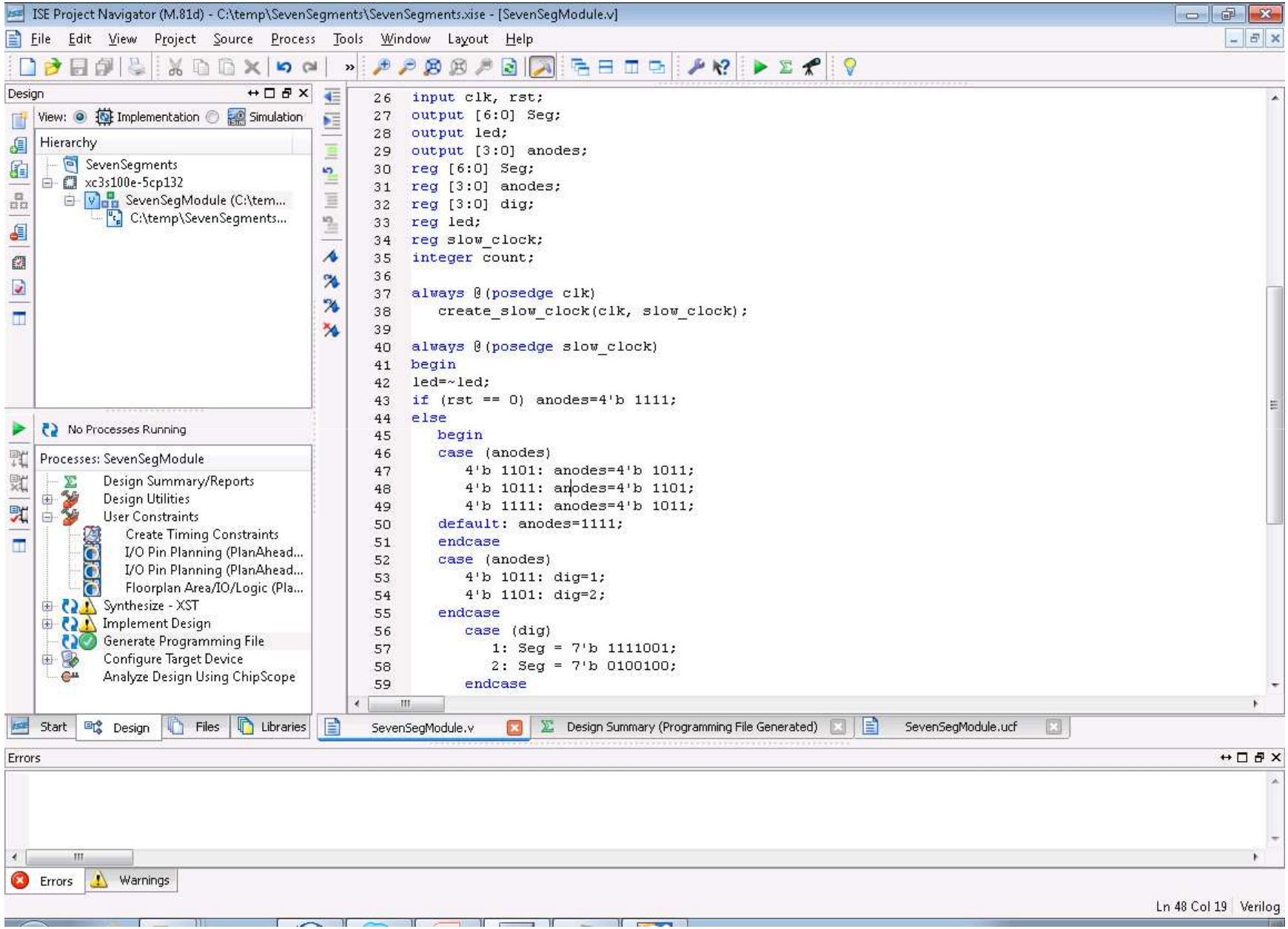
-- Click save design and exit



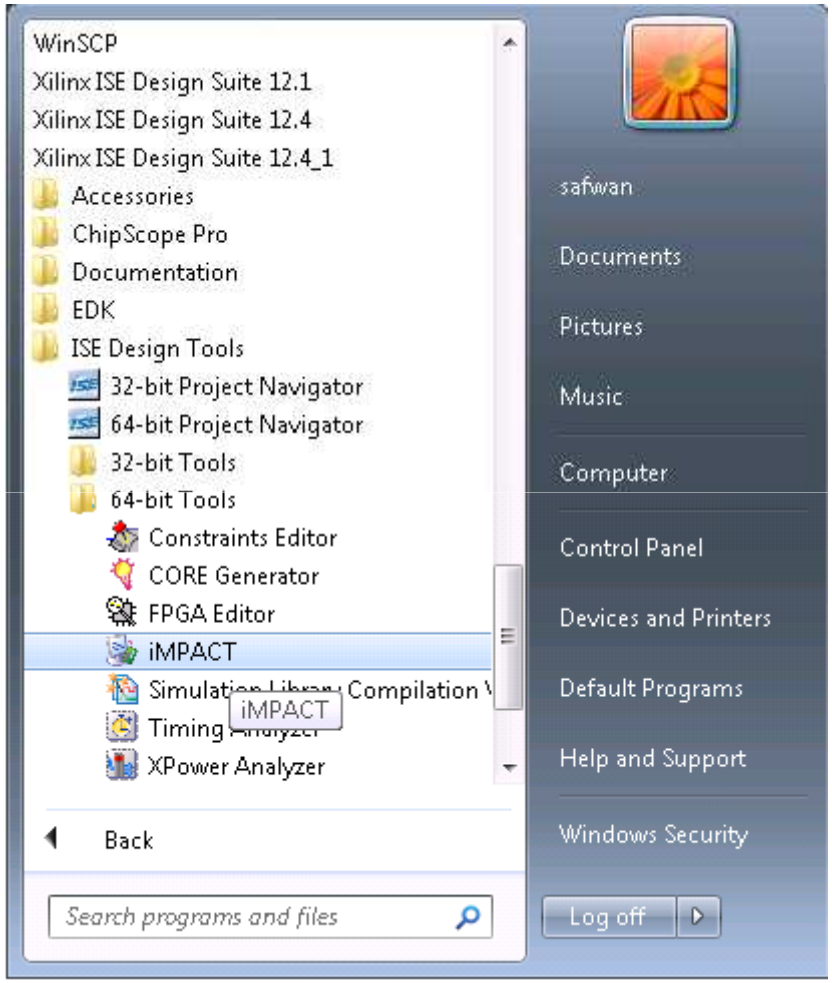
Create the *.bit file,* Double click on *Generate Programming File in the Processes Window*



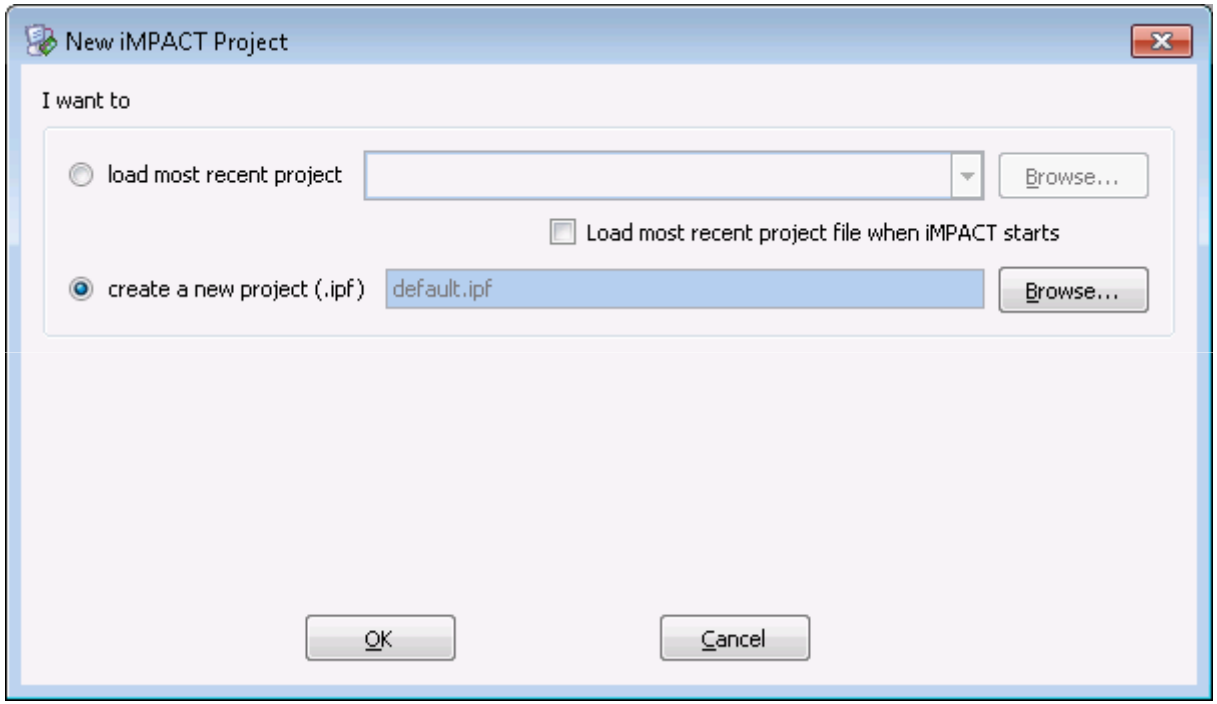
Make sure that every thing is compiling and building



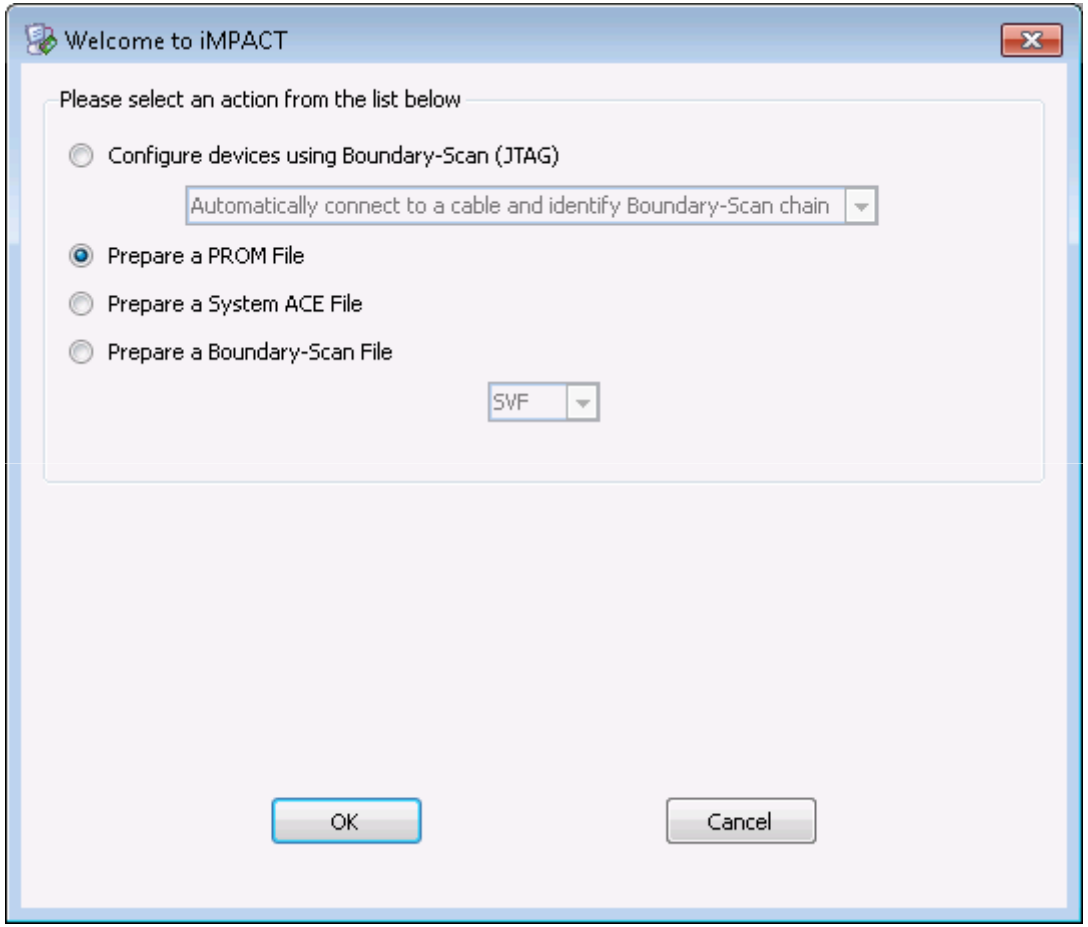
Open iMPACT application from Xillinx ISE Deisgn Suite 12.4 tools



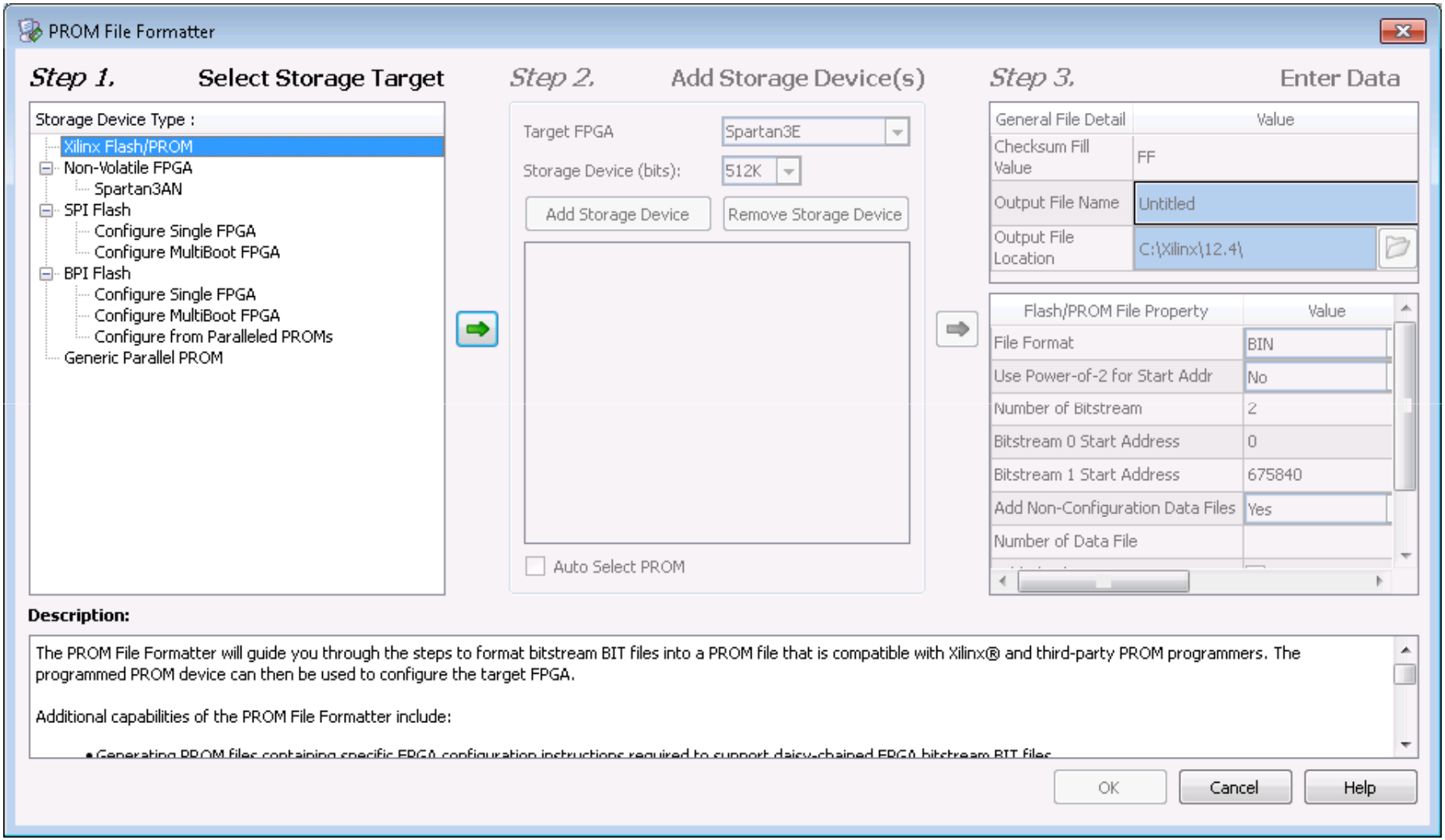
Create New Project



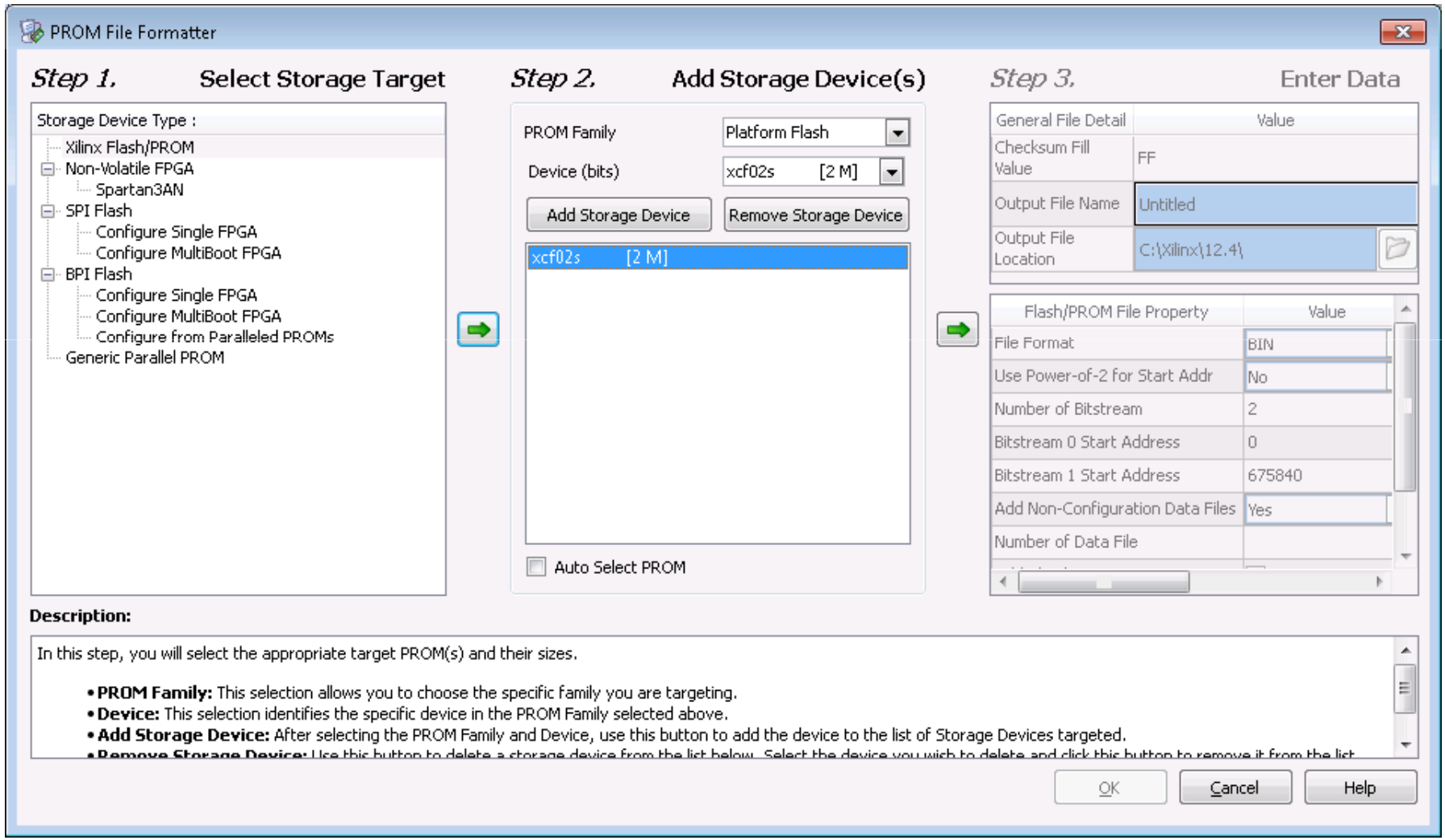
In the *Welcome to iMPACT popup window, select Prepare a PROM File and click on OK*



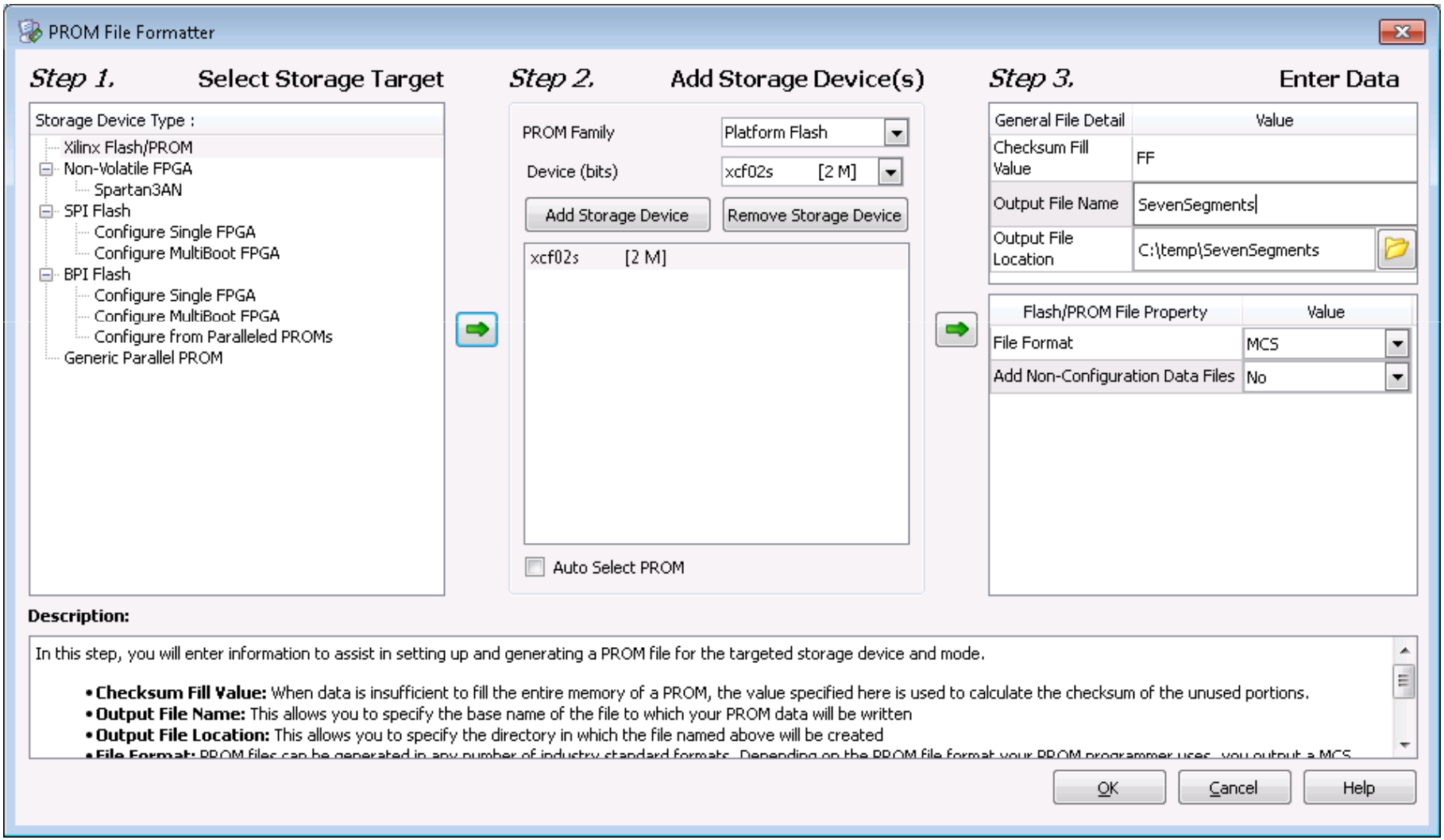
- Select *Xilinx Flash/PROM*
- Click on the green arrow



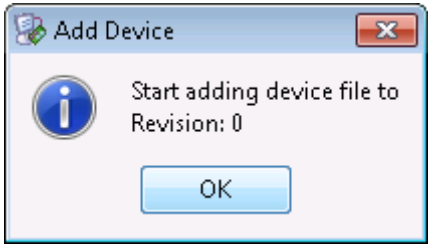
-Select *Platform Flash under PROM Family*
-Select *xcf02s under Device*
-Click on *Add Storage Device*
-Click on the green arrow



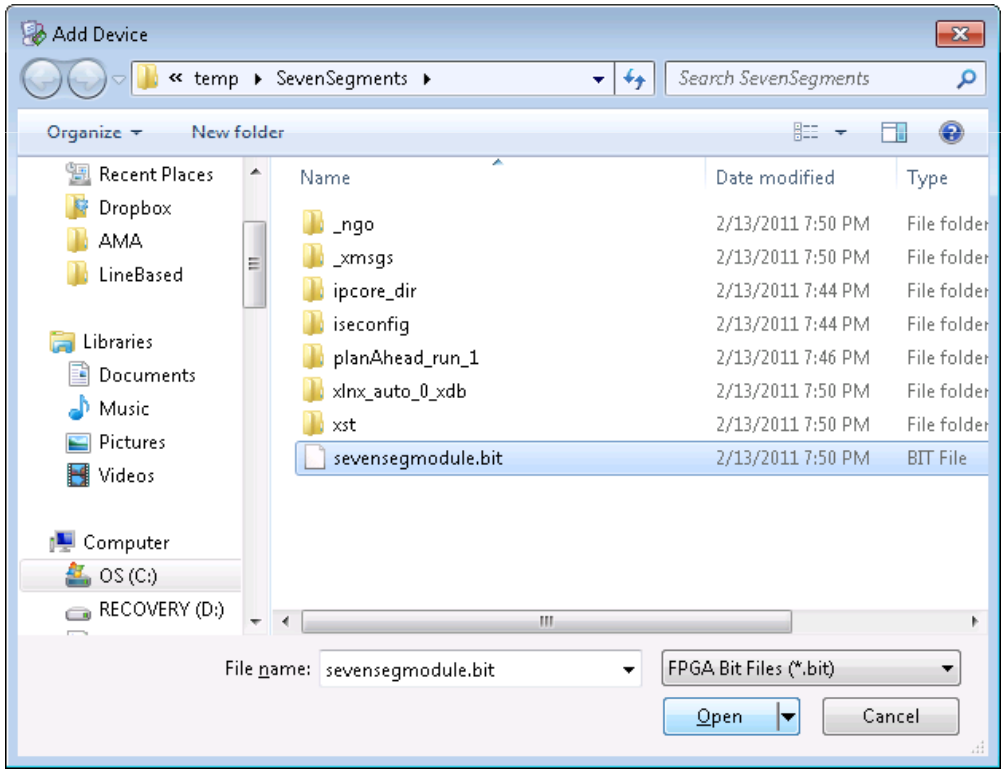
-*Checksum Fill Value should be FF*
-Enter a filename & location
-Select *MCS under File Format*
-Select *No under Add Data Files*



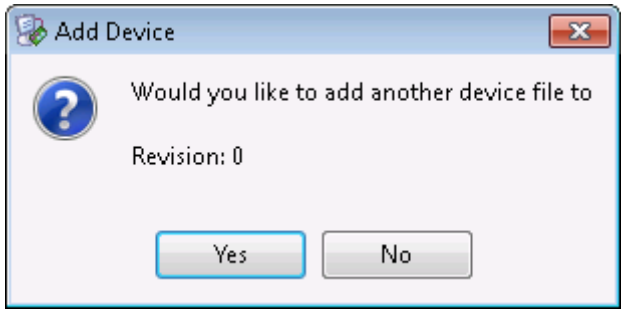
An *Add Device window will appear indicated that Xilinx will start adding the device file to data stream 0, click ok*



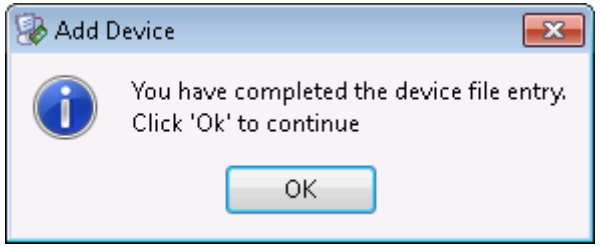
A popup window will appear Select your design (bit file) click *Open*



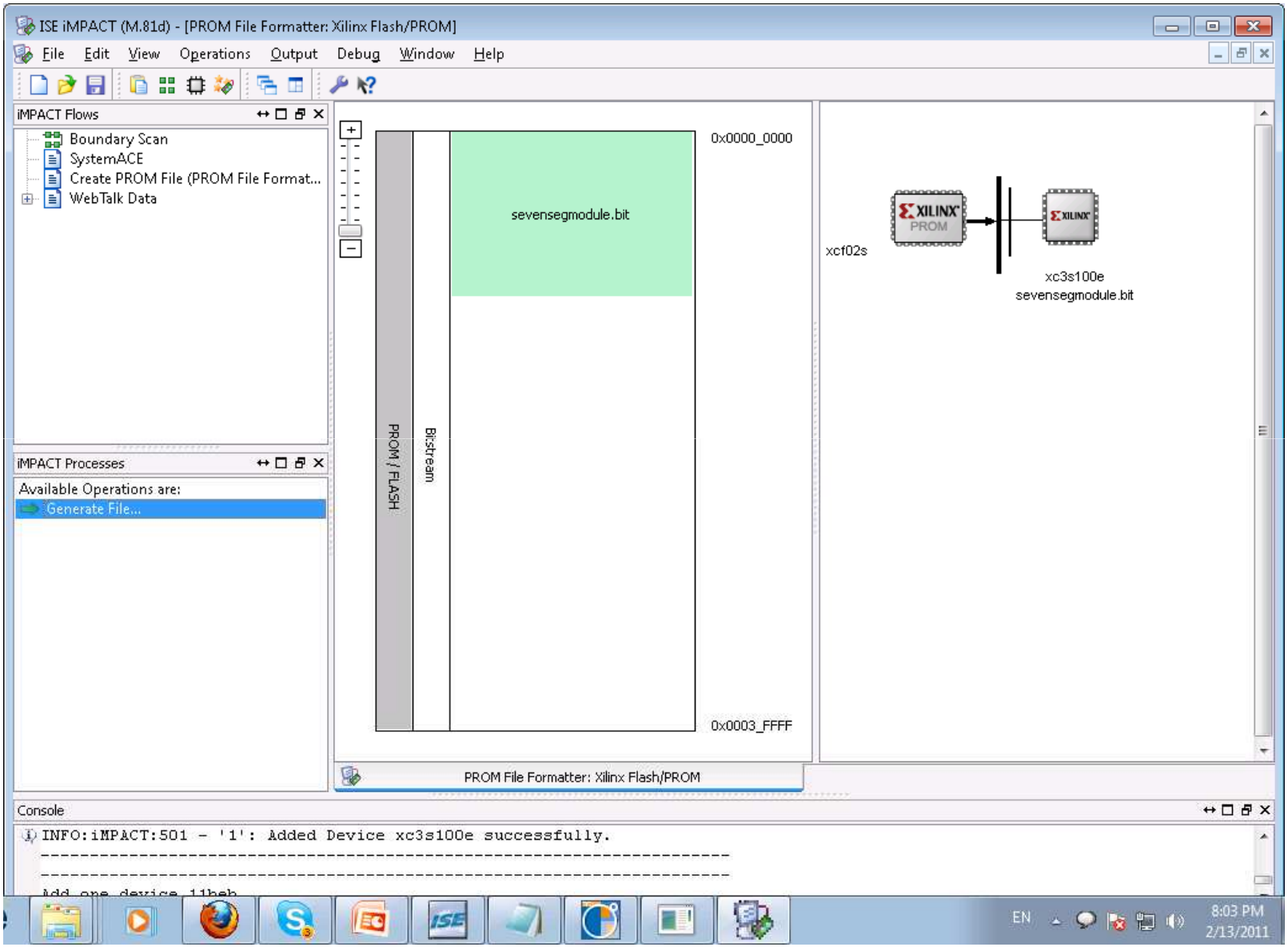
You'll be asked, "Would you like to add another design file to Data Stream:0?", click **NO**



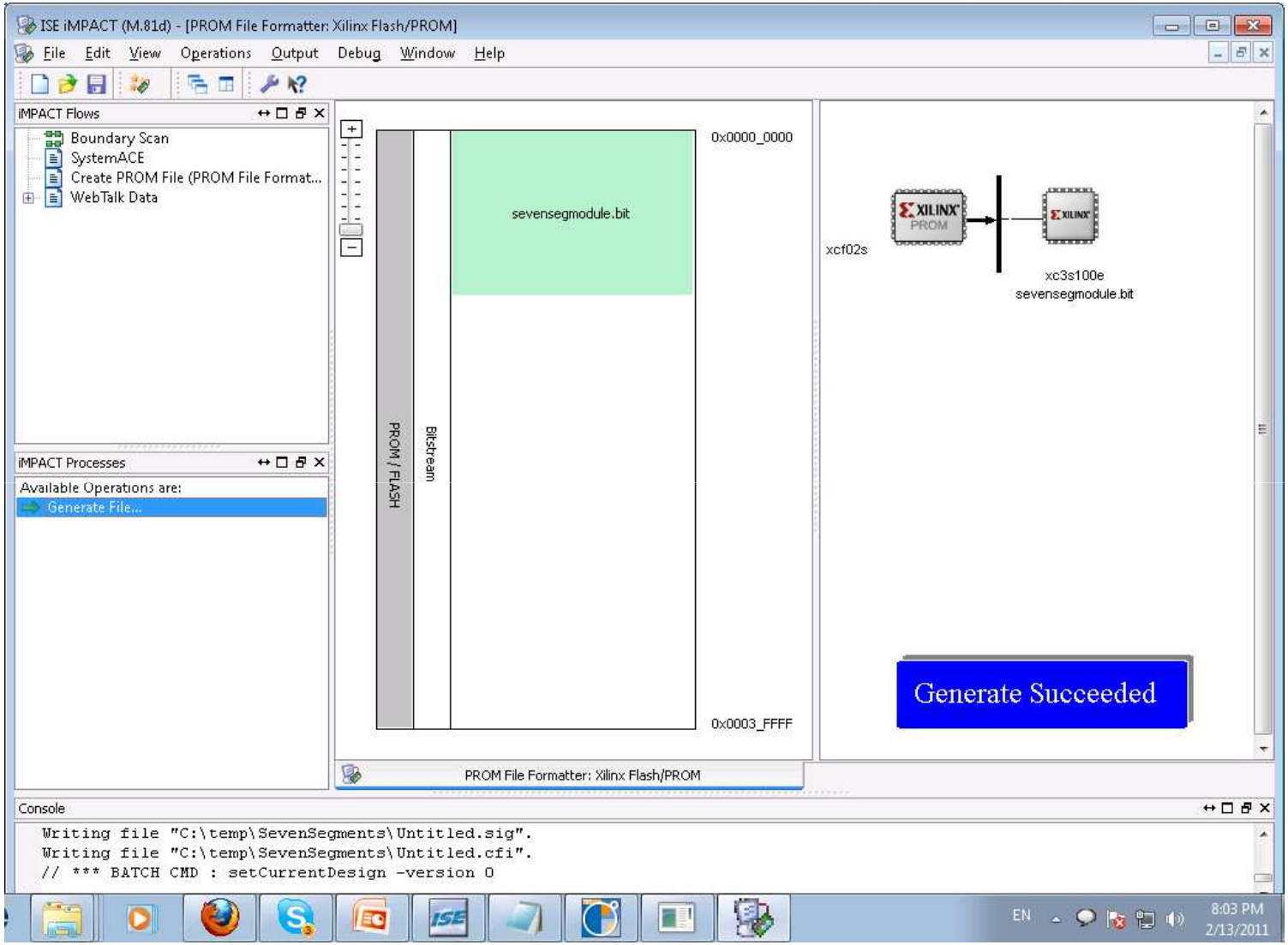
You'll be informed that you've completed the device file entry, click OK



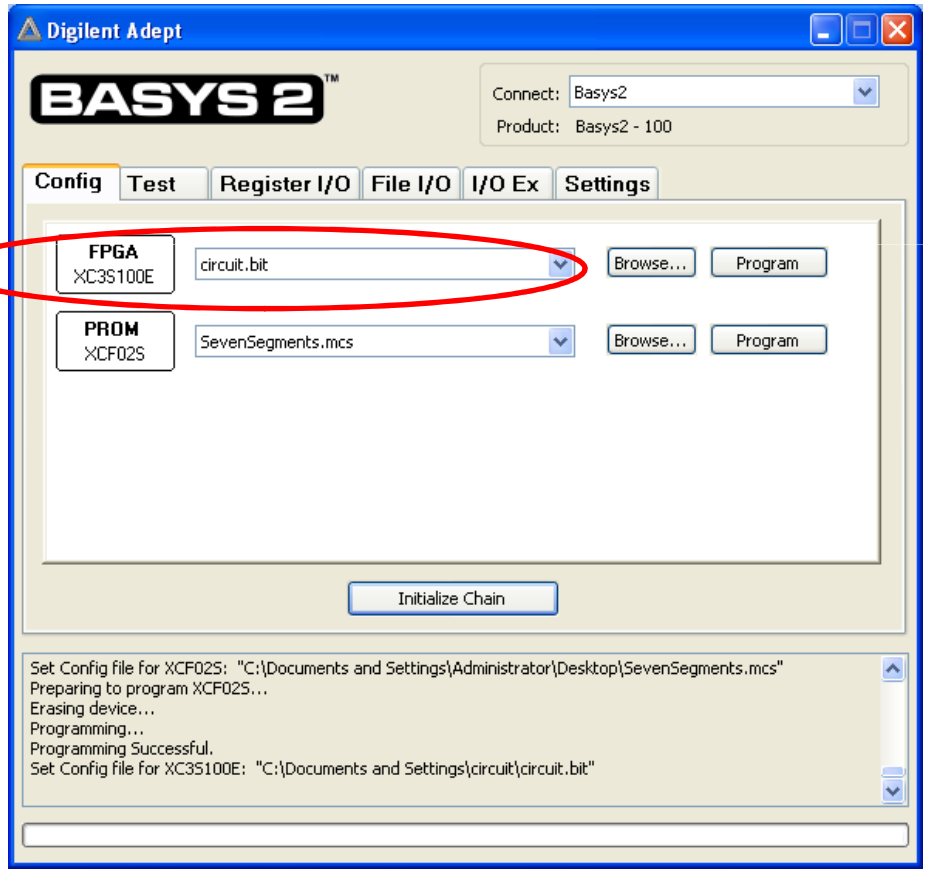
In the *Processes window, double click on Generate File*



If all goes well, a blue success message will be displayed



- Open *Adept2.1 (Downloaded from Digilent's Website)*
-Click on the *Config tab*
-Select the *.mcs file for the PROM (XCF02S) by clicking on the Browse button*
- Click *Program*
-*Wait until the Program successfully loaded*
-*Turn the board on and off , then run the program*



Example 2:

- A 0, 1, 2, or 3 is displayed on the seven-segment display, depending upon whether button #0, #1, #2, or #3 is pressed

- Enable
    - Switch #0

```
module ckt(btn, clk, a, b, c, d, e, f, g, an, rst);
input [3:0] btn;
input clk, rst;
output a, b, c, d, e, f, g;
output [3:0] an;
reg a, b, c, d, e, f, g;
reg [2:0] cstate, nstate;
reg [3:0] an;
always @(posedge clk or negedge rst)
begin
if (~rst) cstate<=7;
else cstate<=nstate;
an=14;
end
always @(btn or cstate)
case (btn)
4'b1000: nstate=3; // Button 3 pressed
4'b0100: nstate=2; // Button 2 pressed
4'b0010: nstate=1; // Button 1 pressed
4'b0001: nstate=0; // Button 0 pressed
4'b0000: nstate=cstate; // No button pressed
default: nstate=7; // No button pressed yet or
multiple
// buttons pressed
endcase
```

```
always @(posedge clk)
case (cstate)
3: begin // Button 3 pressed
a=0; b=0; c=0; d=0; e=1; f=1; g=0;
end
2: begin // Button 2 pressed
a=0; b=0; c=1; d=0; e=0; f=1; g=0;
end
1: begin // Button 1 pressed
a=1; b=0; c=0; d=1; e=1; f=1; g=1;
end
0: begin // Button 0 pressed
a=0; b=0; c=0; d=0; e=0; f=0; g=1;
end
7: begin // No button pressed yet or multiple
buttons pressed
a=1; b=1; c=1; d=1; e=1; f=1; g=1;
end
endcase
endmodule
```

References

- Michael D. Ciletti, *Advanced Digital Design with the Verilog HDL, Pearson Education, Inc.*
- (Prentice Hall), 2003
- Donald E. Thomas and Philip R. Moorby, *The Verilog Hardware Description Language,*
- Kluwer Academic Publishers, 1998
- Samir Palnitkar, *Verilog HDL A Guide to Digital Design and Synthesis, Prentice Hall, Inc., 4th*
- Edition, 1996
- David R. Smith and Paul D. Franzon, *Verilog Styles of Digital Systems, Prentice Hall, Inc.,*
- 2000
- *Digilent Basys 2 Board Reference Manual, Digilent, Inc., May 25, 2009*
- *Digilent BASYS 2 System Board Schematics, Digilent, Inc., December 12, 2008*