

FPGA Development for the DSPtronics boards

DSPtronics LLC

April 8, 2011

Contents

1	Introduction	3
2	Creating a new ISE Project	3
2.1	Start ISE Project Navigator	3
2.2	Create a New Project	3
2.3	Project Configuration	4
2.4	Project Design Files	6
3	HDL Creation	8
3.1	Create a Verilog Design	10
3.2	Create a VHDL Design	13
3.3	Create a MyHDL Design	14
4	Pin Constraints and the Xilinx ISE UCF File	15
5	Synthesis and PAR	18
5.1	Reports	19
6	Programming the .bit File	20
7	Archived Xilinx ISE Projects	20

1 Introduction

If you are unfamiliar with FPGA development the following is a step-by-step guide to running an HDL design through the Xilinx synthesis and place and route (PAR) and how to configure one of the DSPtronics development boards. This tutorial is intended for those new to FPGA development. This tutorial will walk through the design flow and the tools required to create a bit file. A bit file is the binary configuration for the FPGA. The bit will program the FPGA with the logic described in the design. This tutorial will toggle the LEDs on the board.

Download the following designs files for this tutorial from www.dsptroncis.com/start.html.

The Xilinx FPGAs on the DSPtronics boards are supported by the Xilinx ISE Webpack software which is FREE. The software can be downloaded from www.xilinx.com. You will need to create an account at Xilinx to be able to download the Webpack software. The download is fairly large which may take some time. After the software is downloaded run the installer and run the ISE application. You should see a window like Figure 15.

2 Creating a new ISE Project

2.1 Start ISE Project Navigator

Find “Project Navigator” in the start menu or click on the ISE icon on the desktop. This will bring up the ISE Project Navigator as shown in Figure 15.

2.2 Create a New Project

Start a new project “File → New Project”. This will start a new project wizard and it will ask questions such as device type etc. The first screen will simply prompt for a “Project Name” and “Project Directory”. The “Top-Level Source Type” should be set as **HDL**. For simplicity this tutorial will assume that the project will be located in C : / DSPtronics / ise / led_example on Windows systems.

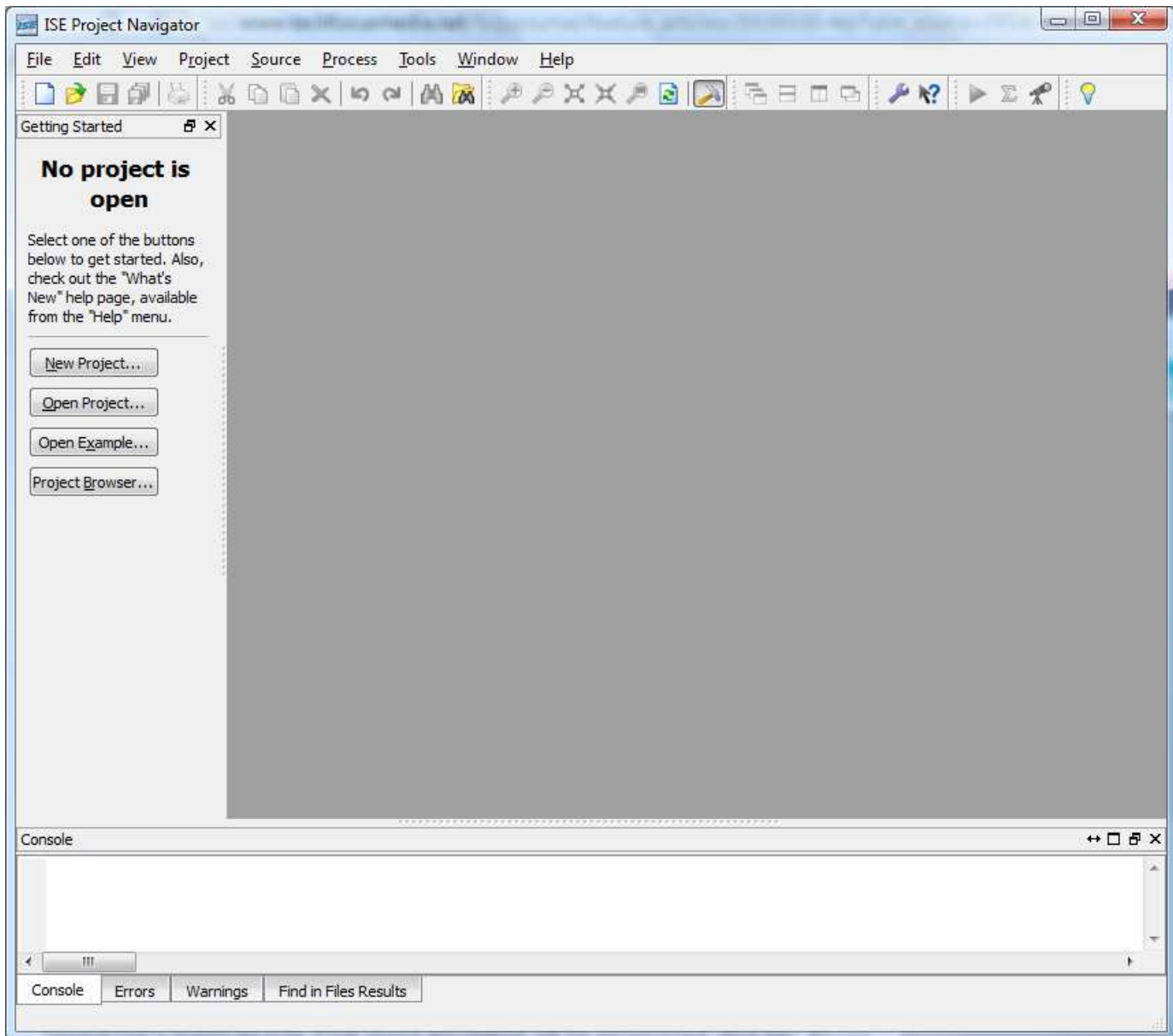


Figure 1: ISE Project Navigator

2.3 Project Configuration

On the next screen fill in the information as follows

Note set the “preferred language” to either VHDL or Verilog based on your preference. This tutorial has examples for VHDL, Verilog, and MyHDL (MyHDL is converted to Verilog or VHDL).

Check the first two check marks at the end of the form and leave the last unchecked. Feel

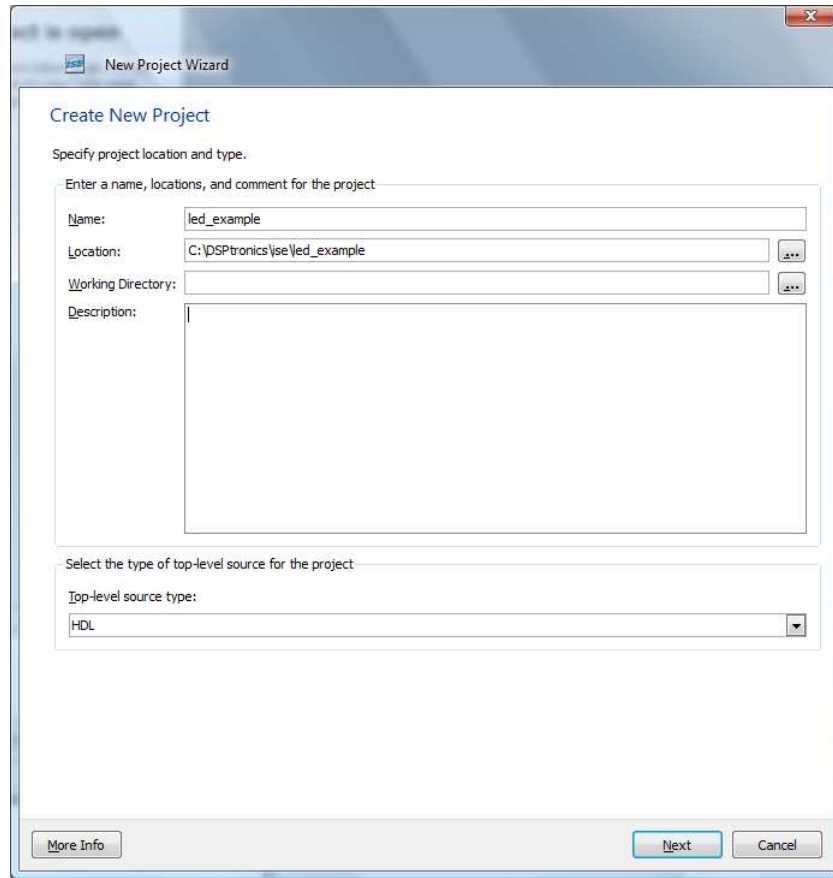


Figure 2: New Project Wizard

Product Category:	ALL
Family:	Spartan3E
Device:	XC3S500 (or XC3S250)
Package:	VG100
Speed:	-4
Top-Level Source Type	HDL
Synthesis Tool:	XST (VHDL/Verilog)
Simulator:	Pick Your Sim
Preferred Language:	VHDL or Verilog

Table 1: ISE Settings

free to set the check marks as desired if you are familiar with the features.

Figure 3 is a screen shot of the correct configuration for the Signa-X500 development board.

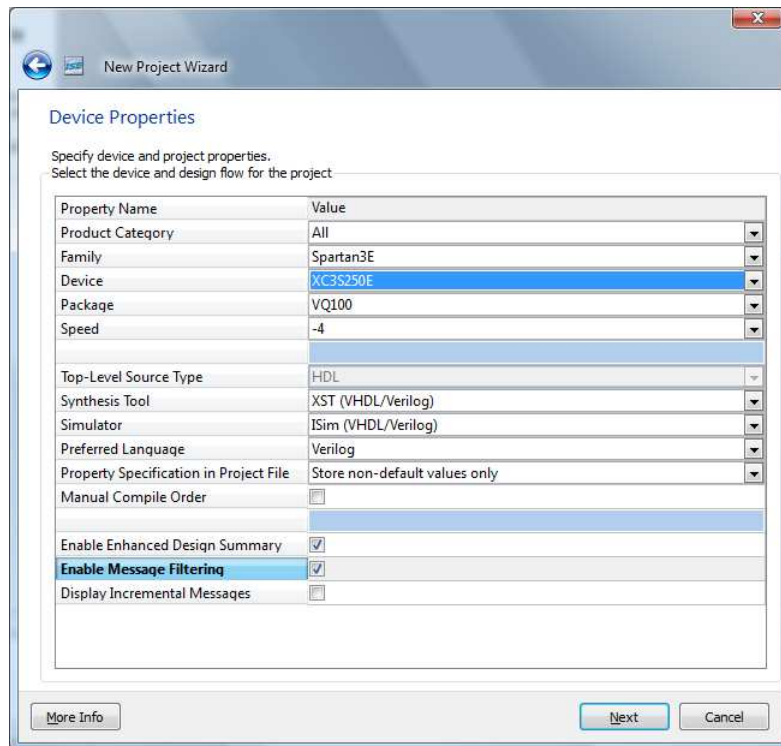


Figure 3: Device Properties

2.4 Project Design Files

The next screen prompts for new files. Create one new VHDL/Verilog file simply called example.vhd (or example.v). If using MyHDL flow don't create a new file here. The generated VHDL/Verilog will be added later. Click "New Source" (pops up a new screen) and type in example.vhd (example.v) in the "File Name" and click "VHDL Module" or "Verilog Module".

The "New Source Wizard" will prompt for the ports of the new module. The ports will be added later, at this point leave the ports blank (don't add anything to the "Define Module" screen).

Click "next" on this screen and "next" on the next couple screens until the project summary appears. The project summary should look something like the following (depending on the installed version of ISE).

Project:

```
Project Name: example1
Project Path: <path>\example1
Top Level Source Type: HDL
```

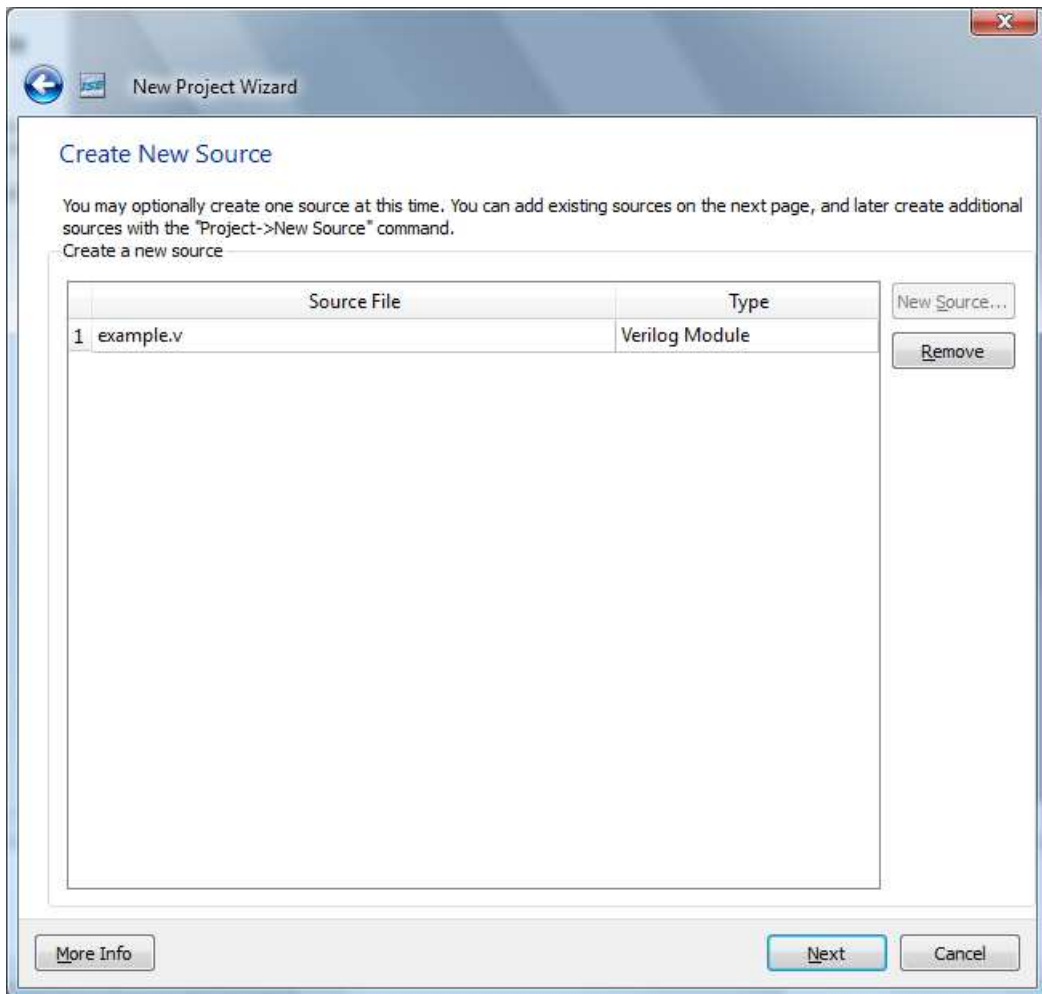


Figure 4: ISE Create New Source

Device:

Device Family: Spartan3E
 Device: xc3s500E
 Package: vq100
 Speed: -4

Synthesis Tool: XST (VHDL/Verilog)
 Simulator: ISE Simulator (VHDL/Verilog)
 Preferred Language: Verilog

Enhanced Design Summary: enabled
 Message Filtering: enabled
 Display Incremental Messages: disabled

New Source:

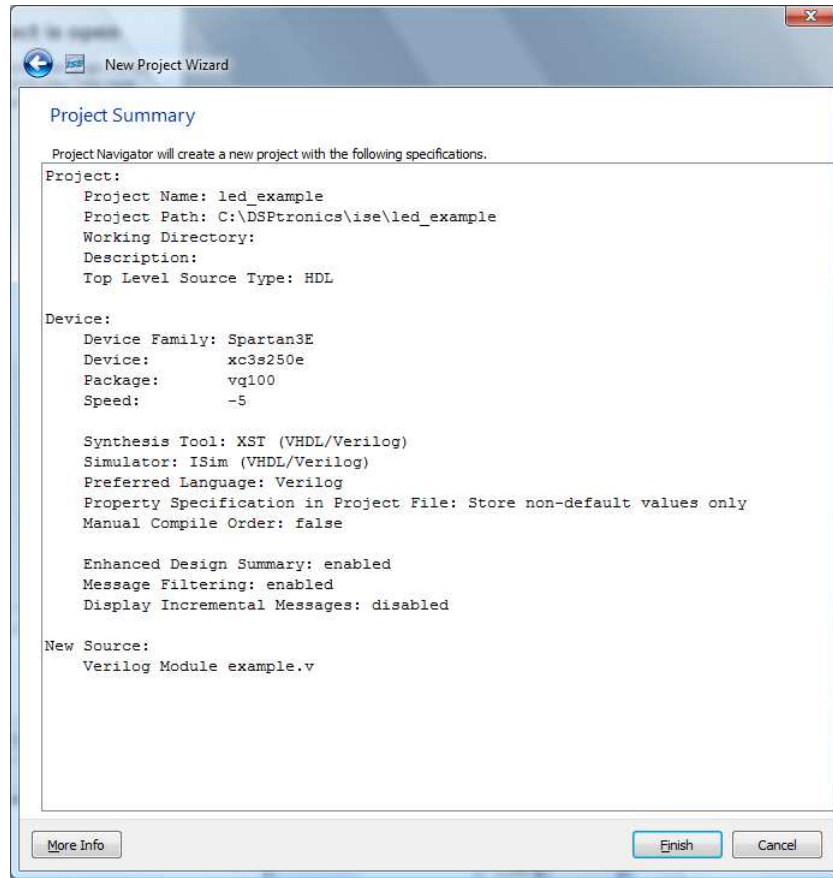


Figure 5: ISE Project Summary

Verilog Module example.v

Click the finish button and you should see a window similar to Figure ??.

3 HDL Creation

Next the HDL (Verilog/VHDL/MyHDL) will be edited to create the design. This section will walk through creating a basic design for the DSPtronics development boards. Each development board will have a set of LEDs (the number of LEDs may vary). The following design examples will create a simple counter that is displayed on the LEDs. To create a binary counter that counts at a slow rate an additional counter will be used to control the rate that the LED counter increments.

All the DSPtronics boards will have some clock that is fed to the FPGA. See the datasheet

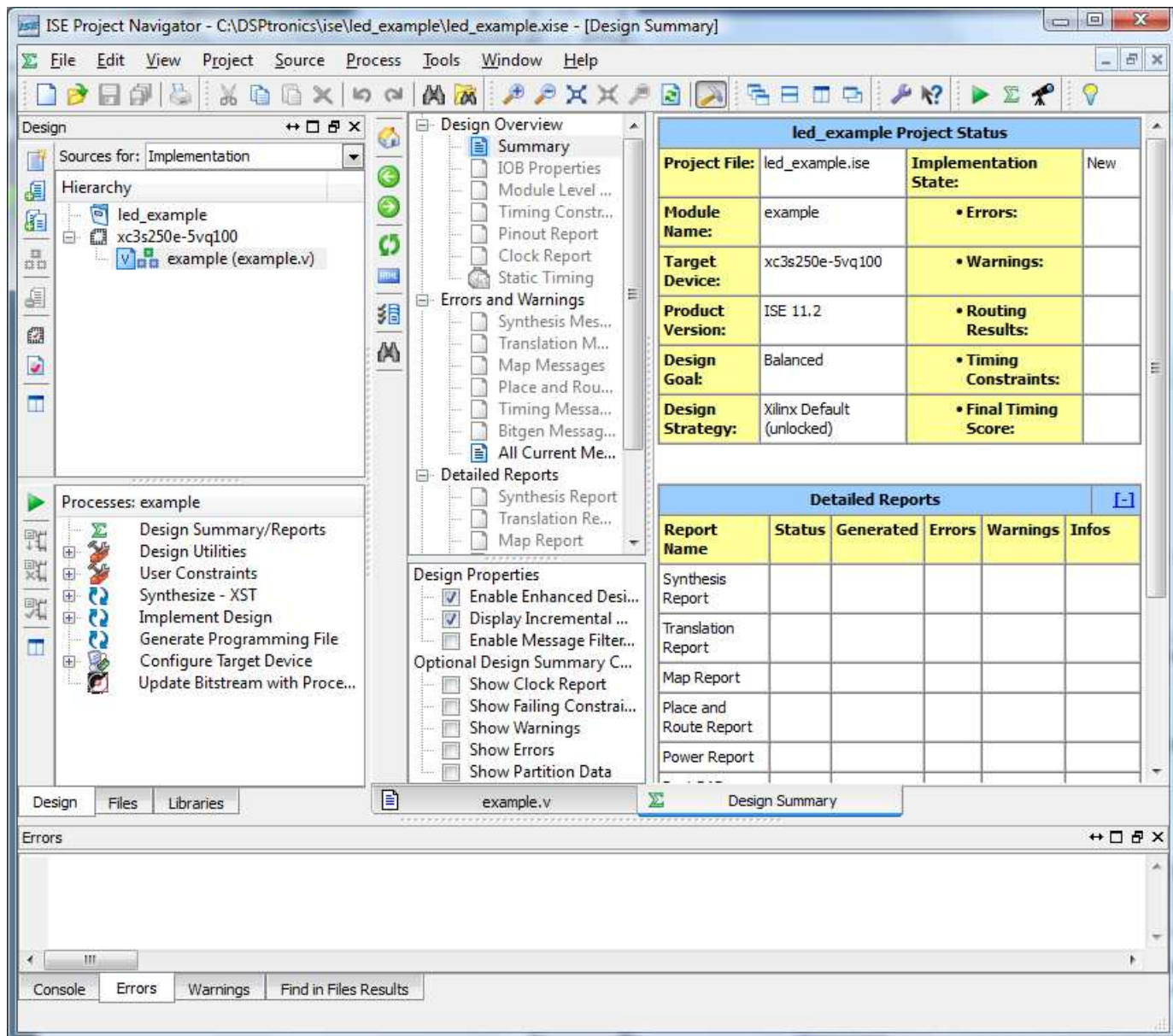


Figure 6: ISE Project Navigator

or user guide for information on available clocks. Some of the boards with the high speed USB controller have a 48MHz clock that is provided by the USB controller. The following examples use this 48MHz clock but the designs can easily be adjusted for any clock source.

If the source was created with the “New Project Wizard” click on the source in the upper left hand corner of the ISE workspace. If the file was not created right click on the device in the upper right hand corner and create a new HDL file (VHDL or Verilog). If using the MyHDL flow goto the MyHDL section and it will describe how to add the generated VHDL/Verilog.

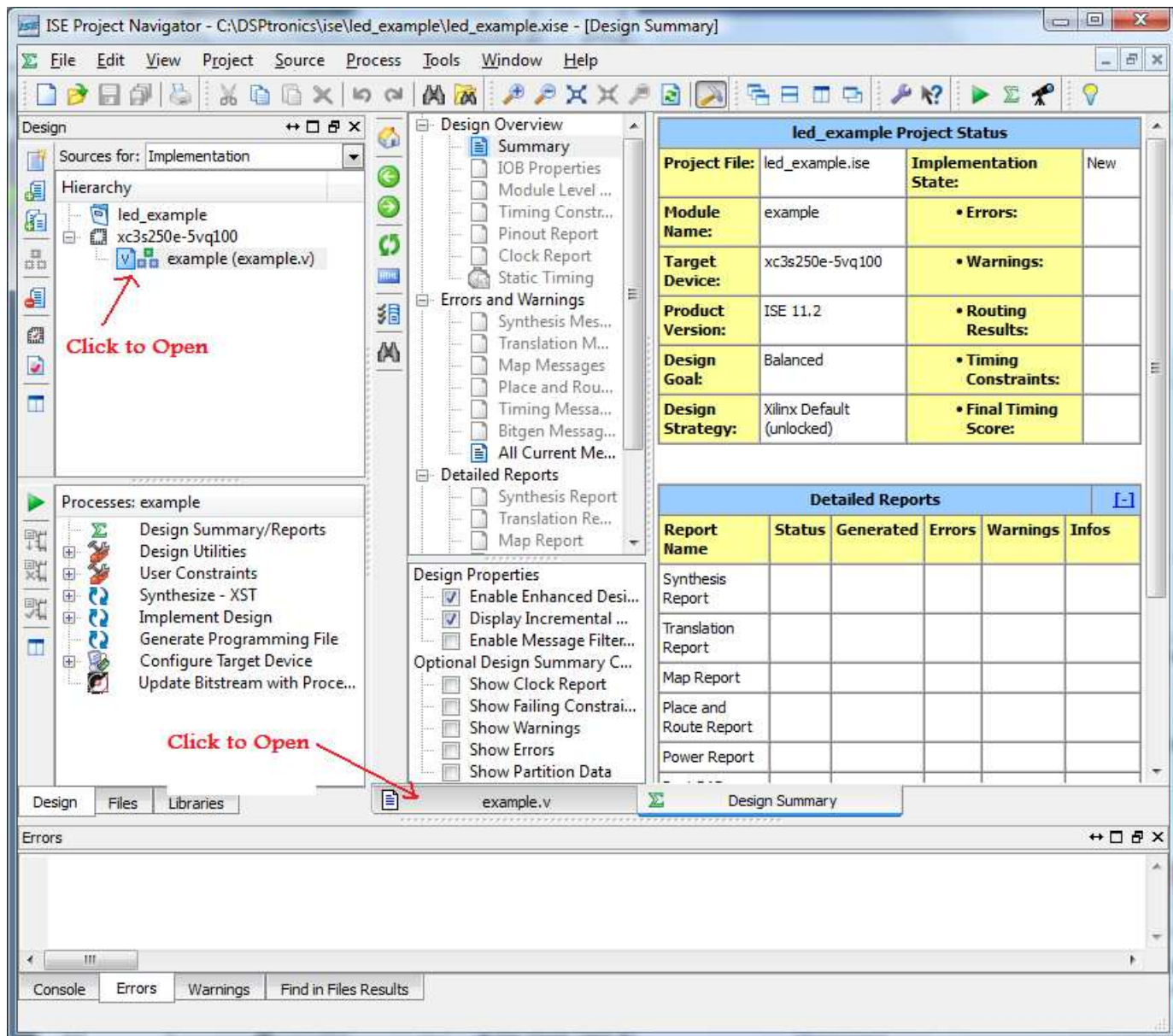


Figure 7: Open the HDL file

To enter the HDL jump to the appropriate HDL is being used, 3.2, 3.1, or 3.3.

3.1 Create a Verilog Design

The Xilinx ISE project wizard automatically entered in the following information when the new source example.v was added in the first part of this tutorial.

```

'timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    19:05:25 03/17/2007
// Design Name:
// Module Name:    example
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module example();

endmodule

```

The above is a basic stub for a Verilog module. The rest of the design will be filled in. The built in editor in ISE will be used for editing the source.

The first item to be entered will be the port definitions. The top-level ports which will map to the FPGA pins on the DSPtronics FPGA development boards. For this example only the clock and LED pins will be used. Again see the datasheet and user guide for more information on the FPGA pins available on the boards.

Enter the following into the Verilog module.

```

module example
(
    input  wire  clk,
    input  wire  reset_n,
    output wire  [6:0] LEDs
);

endmodule

```

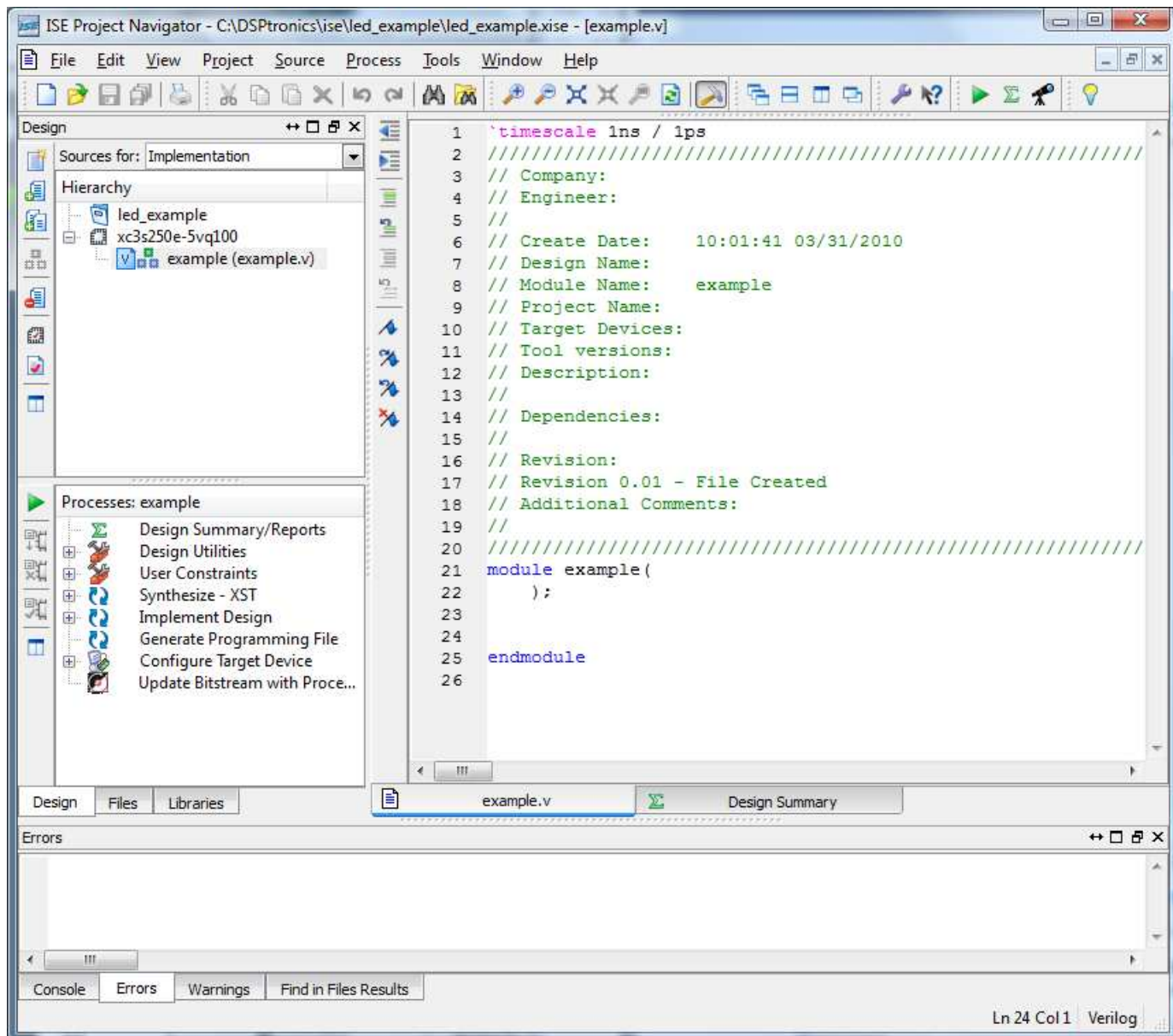


Figure 8: ISE Editor Verilog

This example is a simple binary counter. Since the clock is 48MHz we will create a clock divider (another counter) that will create a clock enable (CE) signal for the LED counter. The following is the complete Verilog code for the simple logic.

```

`timescale 1ns / 1ns

module example
(
    input wire clk,

```

```

output wire [6:0] LEDs
);

localparam CLK_RATE = 48000000 / 10; // 100ms = 1sec/10

reg [6:0] led_reg;
reg [31:0] clk_cnt;
reg clk_en;

assign LEDs = led_reg;

always @(posedge clk) begin
    if(clk_cnt >= CLK_RATE) begin
        clk_cnt <= 32'b0;
        clk_en <= 1'b1;
    end
    else begin
        clk_cnt <= clk_cnt + 1;
        clk_en <= 1'b0;
    end
end

always @(posedge clk) begin
if(clk_en) led_reg <= led_reg + 1;
end

endmodule

```

After the HDL has been entered goto 4 section to define the pin to signal connections.

3.2 Create a VHDL Design

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity example is
port (
    clk : in std_logic;
    LEDs : out unsigned(6 downto 0)
);

```

```

end entity example;

architecture RTL of example is
    signal led_reg : unsigned(6 downto 0) := 0;
    signal clk_en  : std_logic := '0';
    signal clk_cnt : unsigned(31 downto 0) := 0;
begin -- RTL

    CLK_CNT: process(clk)
    begin
        if clk_cnt >= CNT_RATE then
            clk_cnt <= 0;
            clk_en  <= '1'
        else
            clk_cnt <= clk_cnt + 1;
            clk_en  <= '0';
        end if;
    end process;

    LED_CNT: process (clk)
    begin -- process LED_CNT
        if clk_en = '1' then
            led_reg <= led_reg + 1
        end if;
    end process LED_CNT;

    LEDs <= led_reg
end RTL;

```

After the HDL has been entered goto 4 section to define the pin to signal connections.

3.3 Create a MyHDL Design

```

def example(clk, LEDs):
    CLK_RATE = 48000000 / 10    # 100ms = 1sec/10

    led_reg = Signal(intbv(0) [7:])
    clk_cnt = Signal(intbv(0, min=0, max=CLK_RATE+1))
    clk_en  = Signal(False)

```

```

@always(clk.posedge)
def rtl_cnt():
    if clk_cnt >= CLK_RATE:
        clk_cnt.next = 0
        clk_en.next = True
    else:
        clk_cnt.next = clk_cnt + 1
        clk_en.next = False

@always(clk.posedge)
def rtl_led():
    if clk_en:
        led_reg.next = led_reg + 1

@always_comb
def rtl_assign():
    LEDs.next = led_reg

return instances()

```

To convert the MyHDL to Verilog or VHDL

```

def convert():
    clk = Signal(False)
    LEDs = Signal(intbv(0)[7:])
    toVerilog(example, clk, LEDs)
    toVHDL(example, clk, LEDs)

```

After the Verilog/VHDL file is created add it to the ISE project.

After the HDL has been entered goto 4 section to define the pin to signal connections.

4 Pin Constraints and the Xilinx ISE UCF File

Now that the logic is defined in the HDL files we need to map the top-level ports to the FPGA pins. This is accomplished with an UCF file. Xilinx ISE has some other tools for assign pins and setting up timing constraints but here we will simply type in the UCF file.

In the Xilinx ISE Project navigator right click on the device in the upper left hand corner.

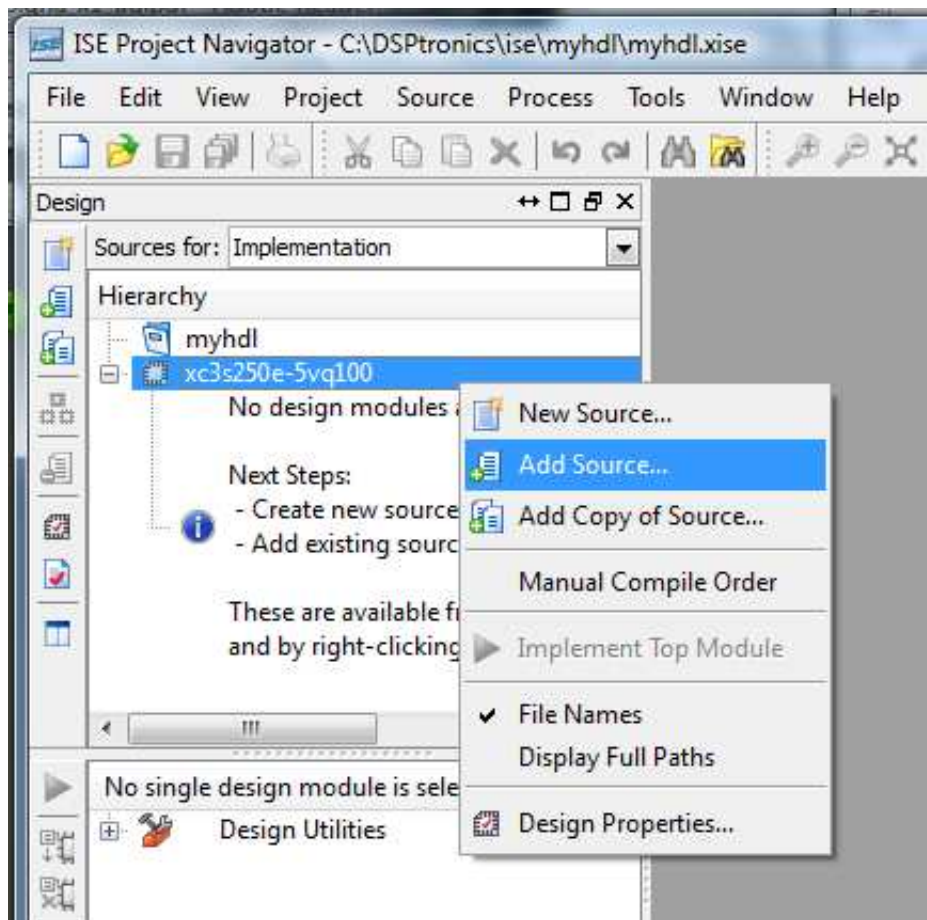


Figure 9:

Select “New File” select UCF type and type example.ucf for the file name. Then select the added UCF file and in the process pane select edit file. Enter the following text.

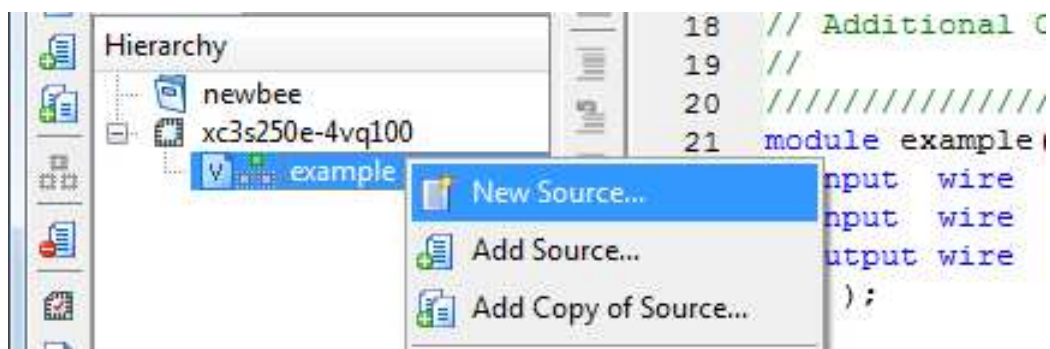


Figure 10: Add New File to Design

To edit the UCF file, click on the newly added file in the source pane and then double click “Edit Constraints” in the process pane as shown in 13. If you double click on the UCF file it will bring up a different editor for entering the constraints. This tutorial doesn’t cover how

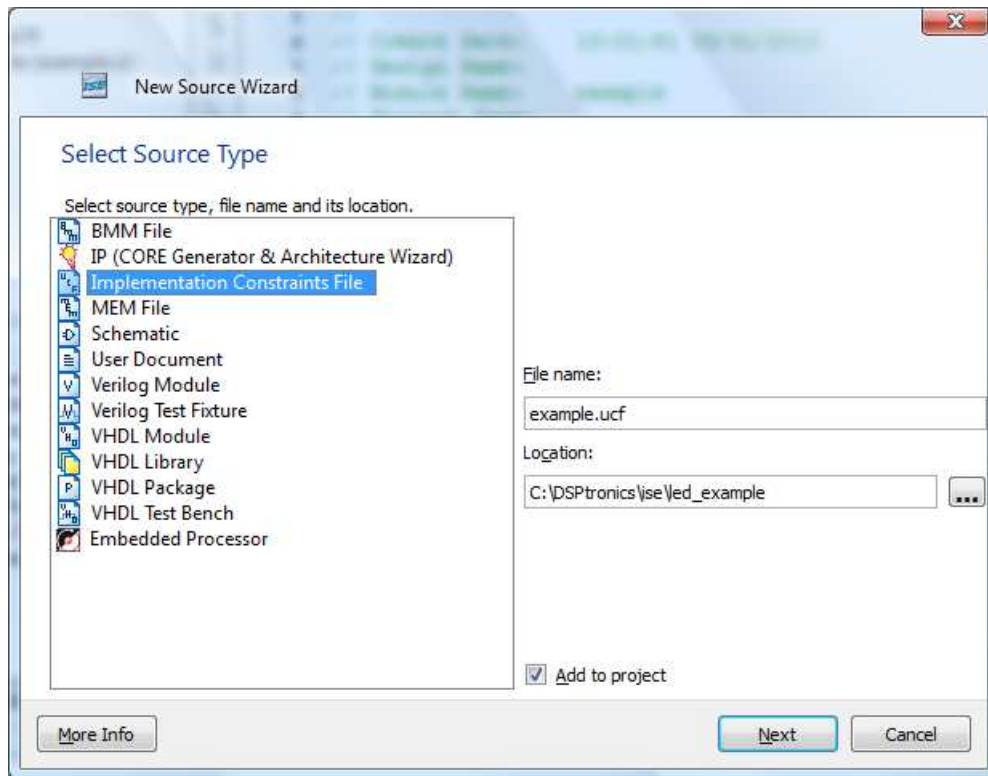


Figure 11: Define UCF File

to use the constraint editor. See the Xilinx website for more information on the constraint editor.

Add the following to the UCF file, this is the constraints for the Signa-X250 and the Signa-X500. See the user guide for other board's pinout.

```
## Signa-X250 and Signal-X500 Constraints
## Pin Constraints
NET "clk"      LOC = P35 | IOSTANDARD=LVCMOS33 ;
NET "LEDs<0>"  LOC = P90 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<1>"  LOC = P91 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<2>"  LOC = P92 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<3>"  LOC = P94 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<4>"  LOC = P95 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<5>"  LOC = P98 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;
NET "LEDs<6>"  LOC = P99 | IOSTANDARD=LVCMOS33 | DRIVE=2 ;

## Timing Constraints
NET "clk" TNM_NET = "clk" ;
TIMESPEC "TS_clk" = PERIOD "clk" 20 ns HIGH 50% ;
```

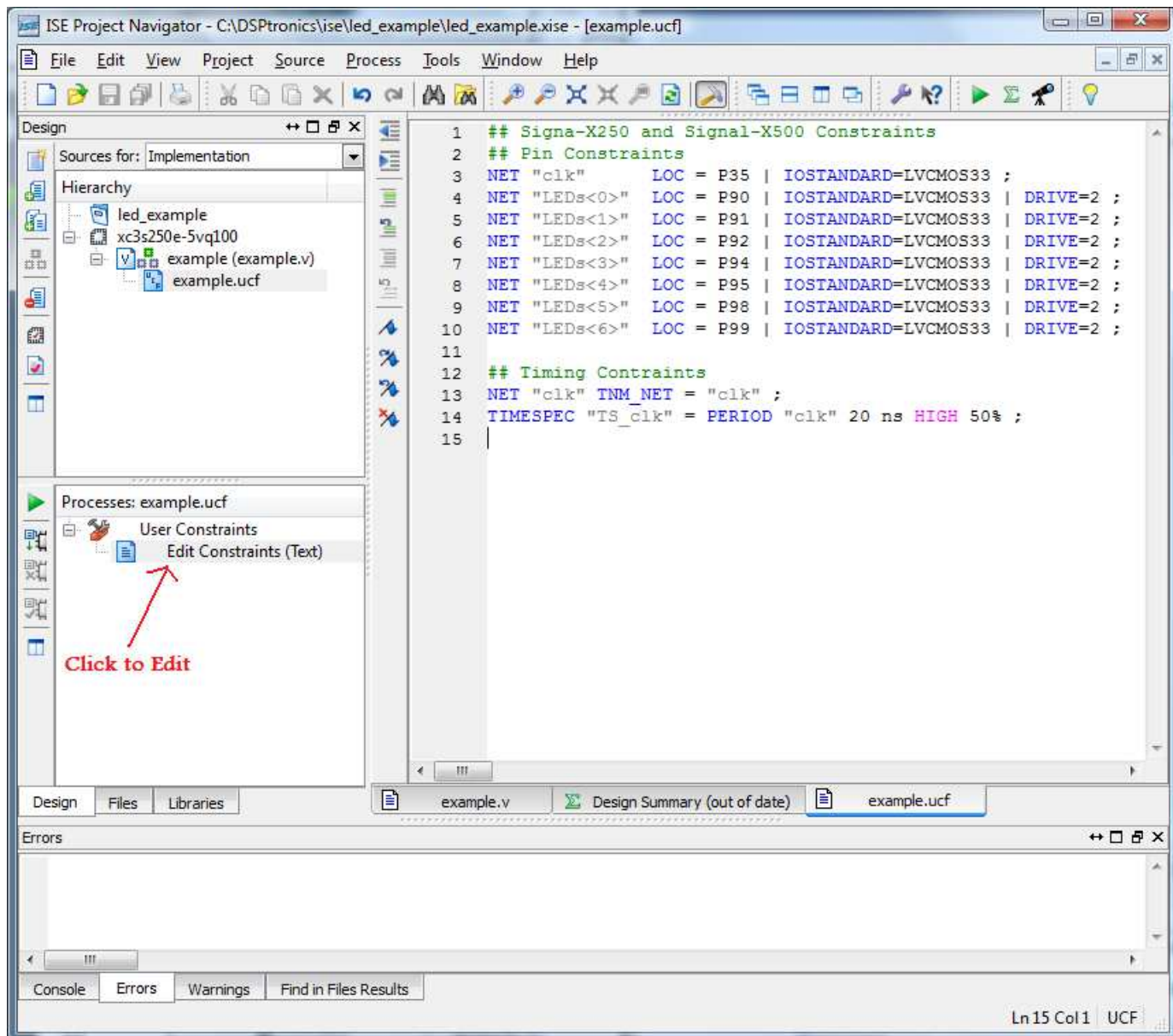


Figure 12:

5 Synthesis and PAR

Once the HDL has been entered and the constraints defined the design needs to be synthesized, placed, and routed. The ISE tool flow can be used to synthesize, place, and route the design. To execute the complete design flow simply double click the “Generate Programming File” in the lower left had corner (process pane). Click on the top-level module, “example”, to get the process pane as shown in ??.

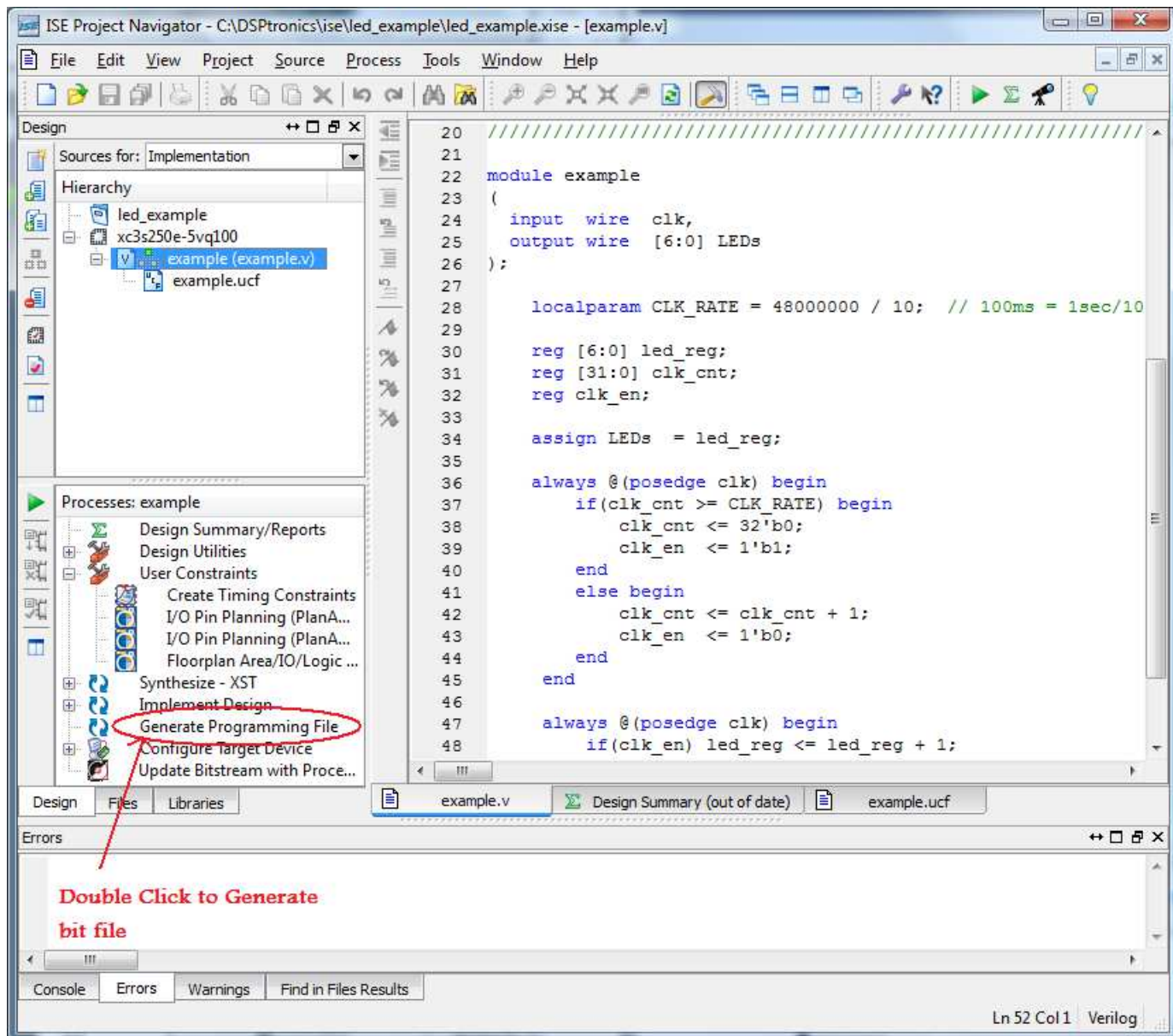


Figure 13: Generate Bit File

Running synthesis, place, and route can take some time. It should go fairly quick for this small design but medium to large designs will take some time.

5.1 Reports

Synthesis, Translation, Mapping, and Place and Route all create reports. The reports can be reviewed and provide information on the logic creation and mapping to hardware resources.

The PaR report will give timing information. 14 shows the summary of the flow. This design only used 40 flip-flops and 15 LUTs. The detailed reports can be accessed by clicking on the circled links in 14.

6 Programming the .bit File

Once the “Generate Programming File” phase is complete a *.bit file is created, in this case it is example.bit. Once the configuration file is created the “FPGA Programmer” application can be used to download the configuration file to the FPGA. Simply open the “FPGA Programmer” application browse to the example.bit file. Then click the config button. Programming the FPGA should take a couple seconds. After the FPGA is configured the binary counter will be displayed on the LEDs.

7 Archived Xilinx ISE Projects

The following are archived Xilinx ISE projects for different DSPtronics mixed-signal development boards. The following can be downloaded and imported into ISE instead of doing the above steps. Only the last step (create the bit file) will need to be executed if the archived project files are used.

Signal-X250 (X500) Boards, USB and Xilinx FPGA with audio CODEC. http://www.dsptronics.com/start/start_xilinx.zip

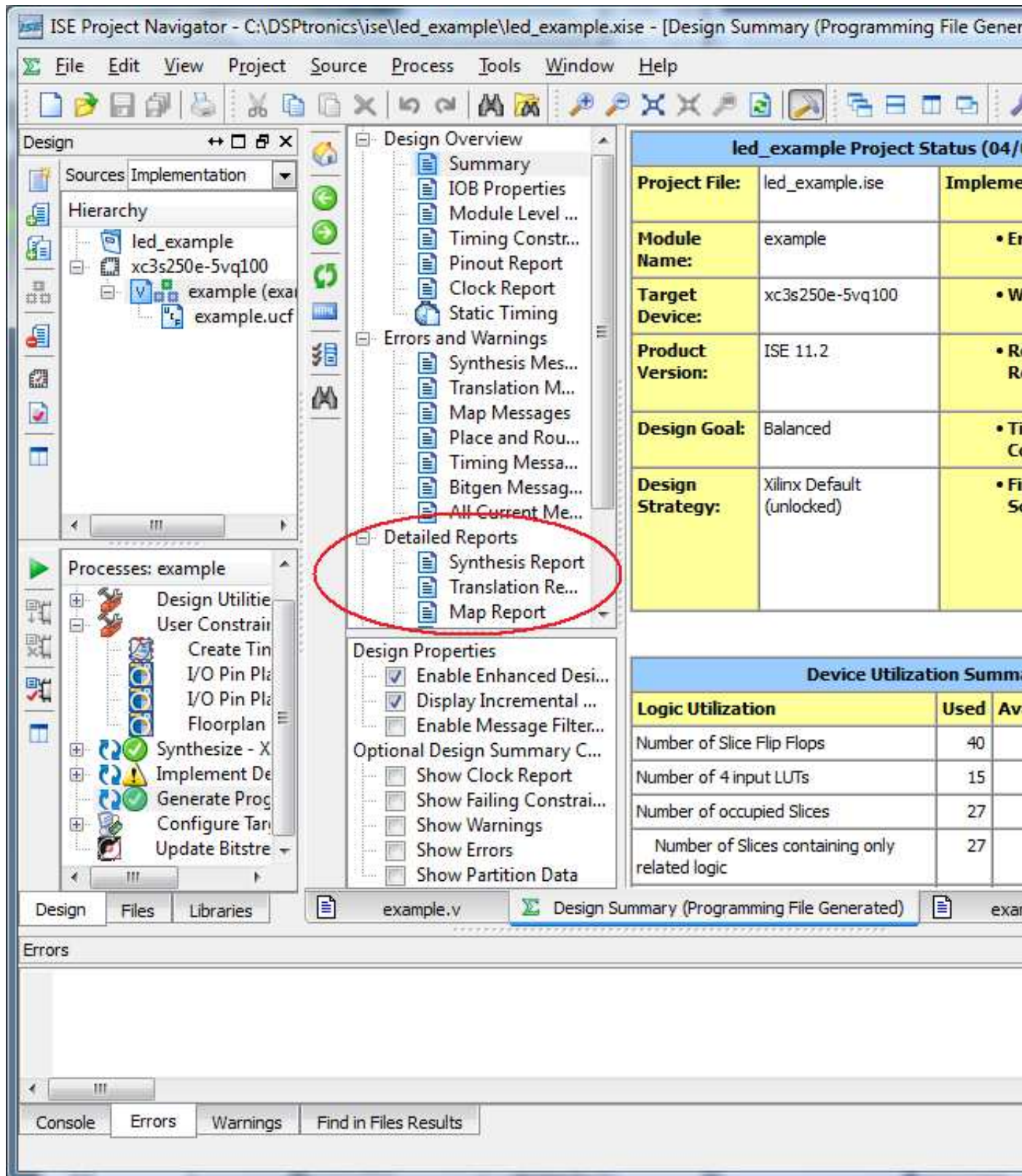


Figure 14: ISE Reports

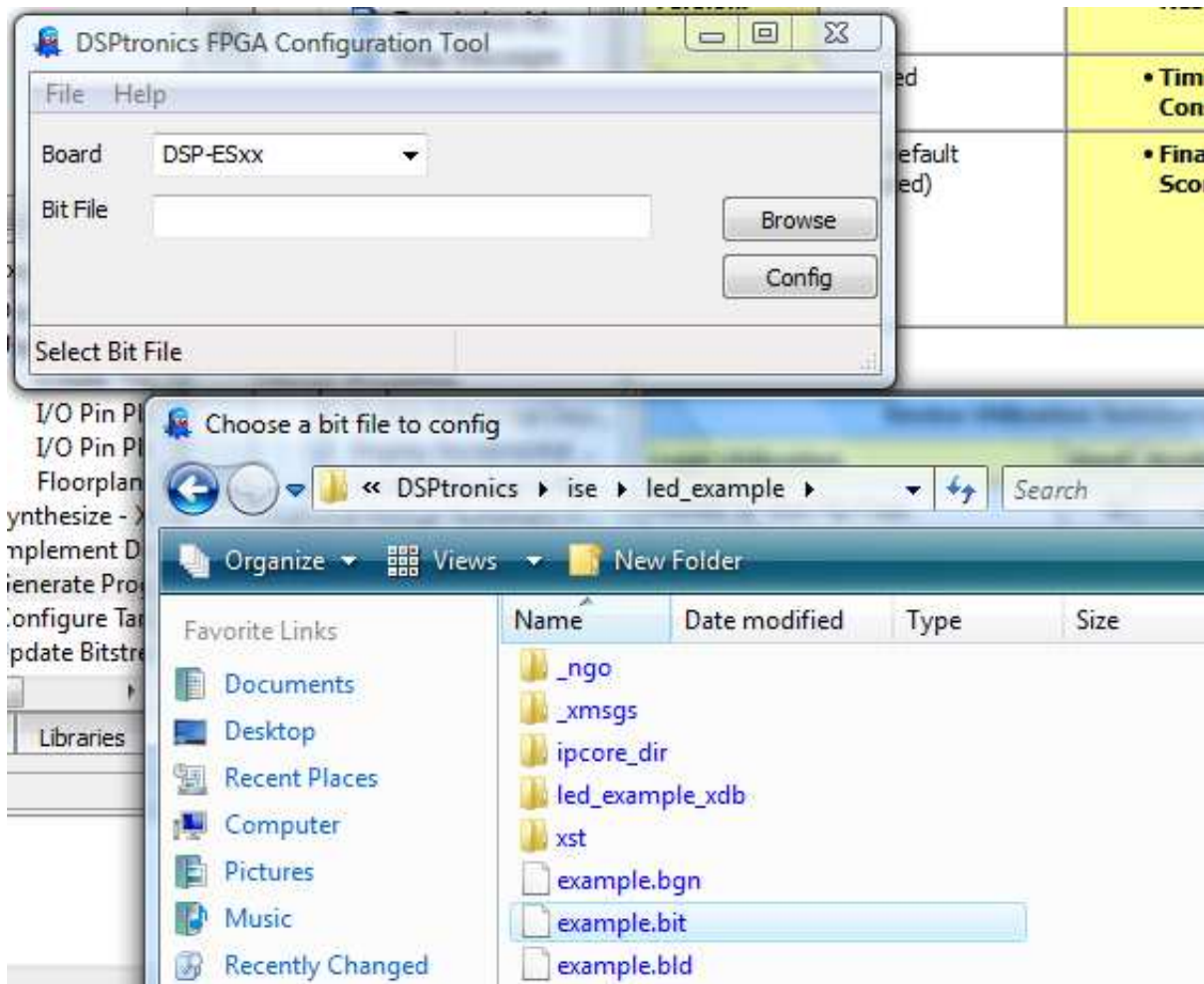


Figure 15: DSPtronics FPGA Programmer