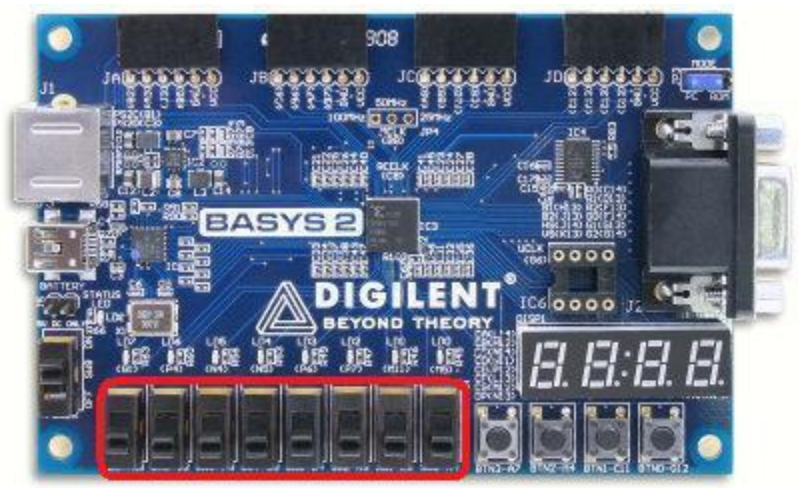
Getting Started With FPGAs



So today, I thought I would talk about something other than microcontrollers. Today, I will talk about how to get started with FPGAs.

What is a FPGA?

An FPGA is a field programmable gate array. Instead of programming a FPGA like your standard microcontroller, you actually "implement" digital circuits using a hardware description language (HDL). The two common hardware description languages includes VHDL and Verilog. For today's tutorial, we'll implement our design using Verilog.

FPGA vs Microcontrollers

So what are the advantages of using a FPGA rather than a microcontroller? The problem with microcontrollers is that they cannot run co-currently, or only run sequentially, and very time limited. A FPGA runs in parallel, or able to run multiple things at once, and much faster than microcontrollers. However, FPGAs are very space limited and initially does not come with all of the fancy peripherals that microcontrollers have. Things like timers, UART, and SPI must be implemented by hand on the FPGA. Also, to have your design stay with the FPGA, you have to use a re-programmable flash memory IC.

Parts List

To get started, you need only the following items

Hardware

- Basys 2 Board (100 die)

Software

- Xilinx Web ISE
- Digilent Adept

Installing Xilinx Web ISE

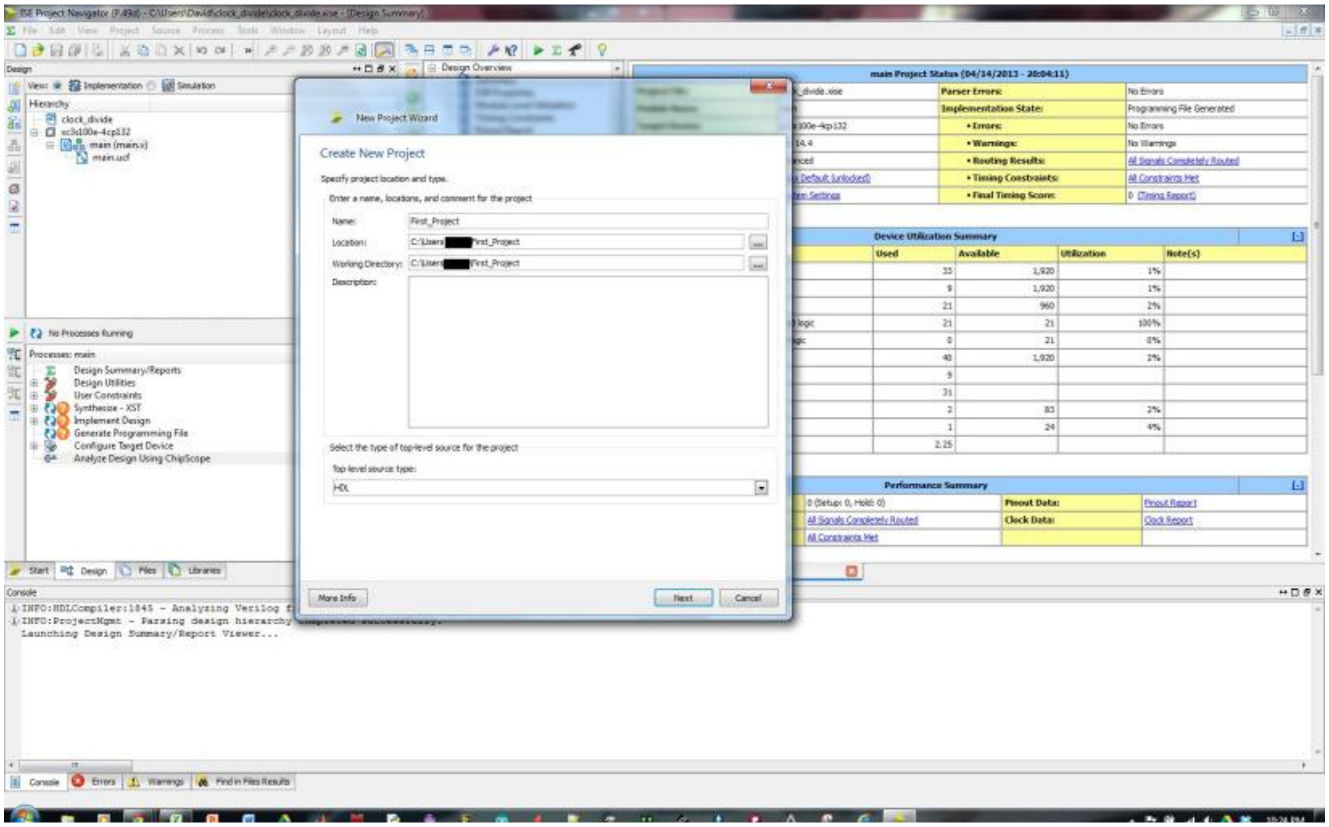
- To learn how to install the Xilinx Web ISE, I highly recommend visiting this URL .They have all the necessary information on properly installing the ISE.

Installing Digilent Adept

- To program the Basys 2 board, you need to download Adept from Digilent's website. Just follow the on screen prompts for proper installation.

Starting A Project

- To create a new project, go to file->new project
- At the create project screen, select a name and location for the project. For this example, let's name the project "First_Project" and set the location to any location on your computer. Make sure you select "HDL" as your Top-level source type.


New Project Wizard

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name: First_Project

Location: C:\Users\[redacted]\First_Project

Working Directory: C:\Users\[redacted]\First_Project

Description:

Select the type of top-level source for the project

Top-level source type: HDL

main Project Status (04/14/2013 - 20:04:11)			
_divide.xise	Parser Errors:	No Errors
	Implementation State:	Programming File Generated
100e-4cp132	• Errors:	No Errors
14.4	• Warnings:	No Warnings
nced	• Routing Results:	All Signals Completely Routed
Default (unlocked)	• Timing Constraints:	All Constraints Met
em Settings	• Final Timing Score:	0 (Timing Report)

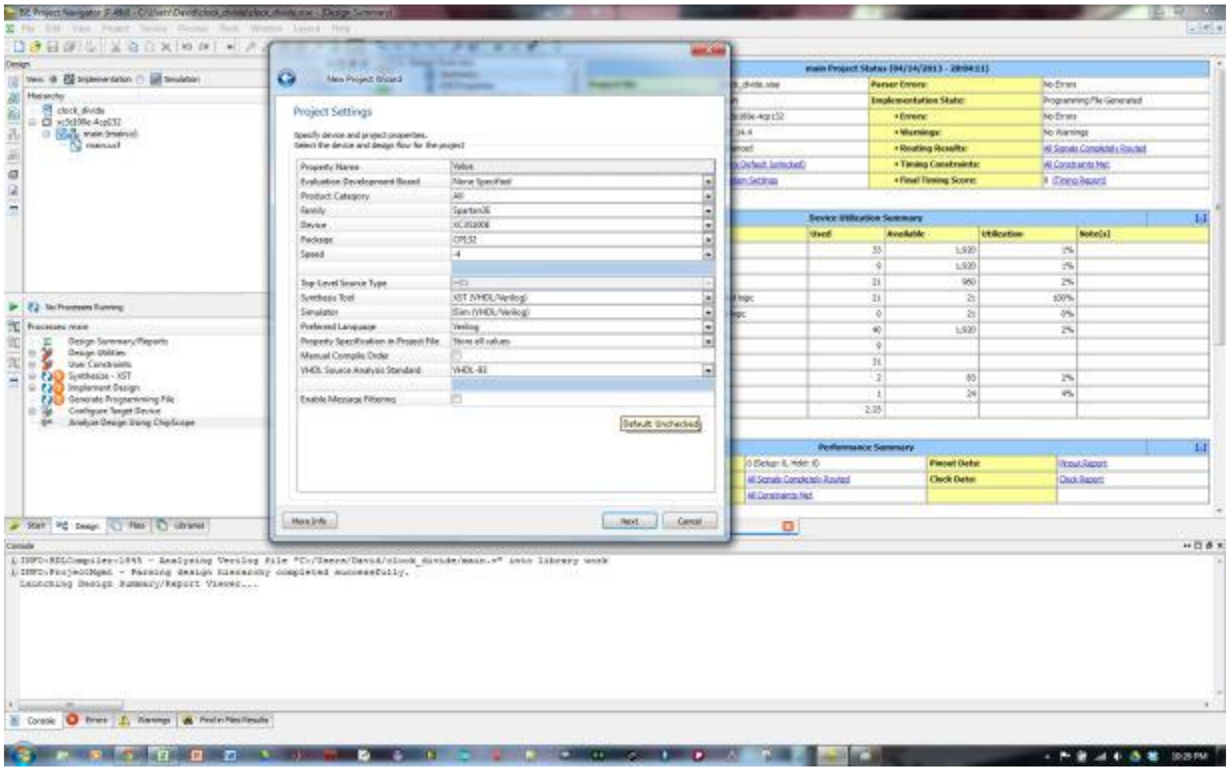
Device Utilization Summary				
	Used	Available	Utilization	Note(s)
	33	1,920	1%	
	9	1,920	1%	
	21	960	2%	
logic	21	21	100%	
gic	0	21	0%	
	40	1,920	2%	
	9			
	31			
	2	83	2%	
	1	24	4%	
	2.25			

Performance Summary		
0 (Setup: 0, Hold: 0)	Pinout Data:	Pinout Report
All Signals Completely Routed	Clock Data:	Clock Report
All Constraints Met		

```
INFO:HDLCompiler:1845 - Analyzing Verilog f
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```

- For project settings, make sure you have the following set…

1. Family is set to Spartan3E
2. Device is XC3S100E
3. Package is CP132
4. Speed is ~4
5. Synthesis Tool is XST (VHDL/Verilog)
6. Simulator is ISim (VHDL/Verilog)
7. Preferred Language is Verilog



Project Settings

Specify device and project properties.
Select the device and design flow for the project

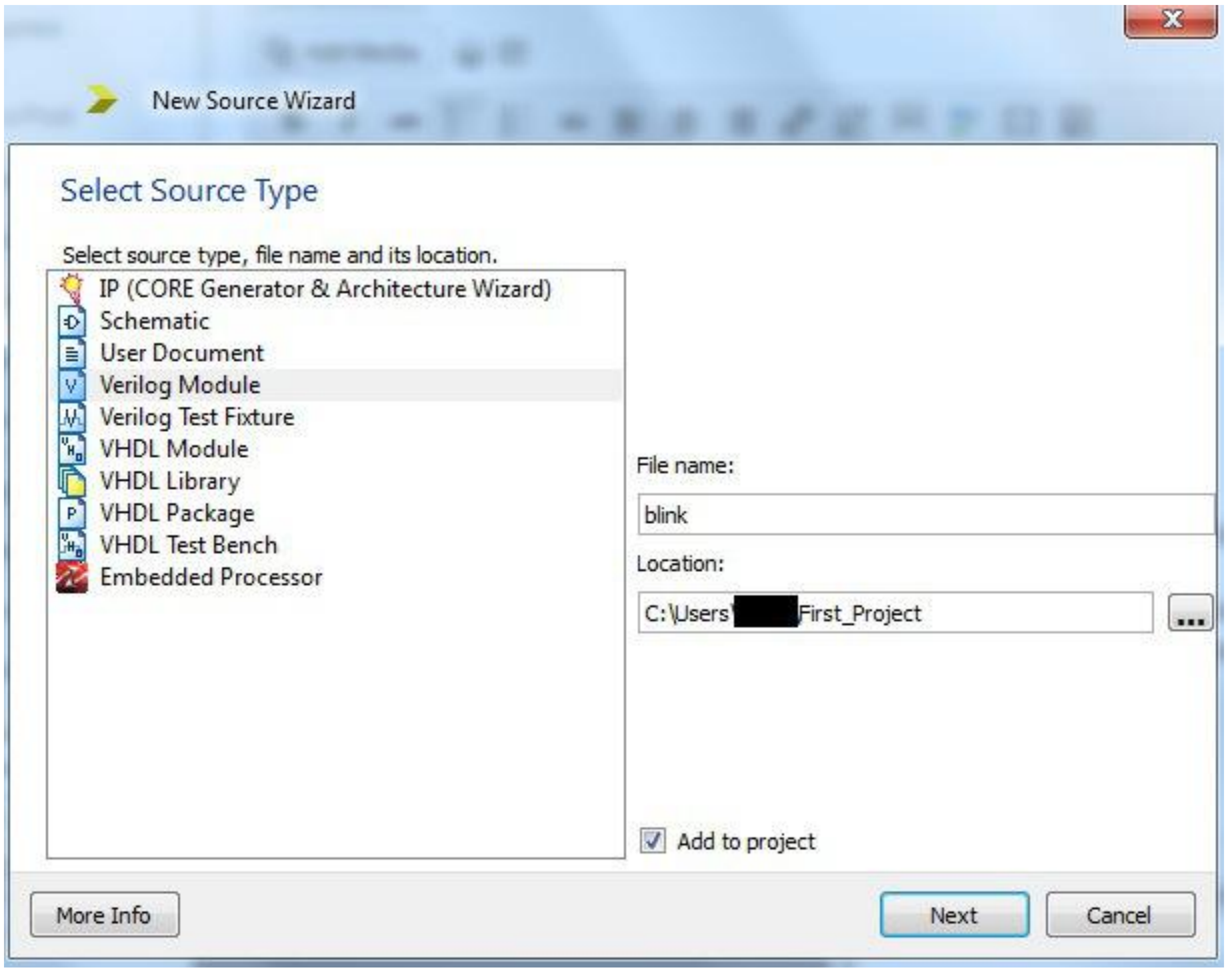
Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	CP132
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

```
INFO:HDLCompiler:1845 - Analyzing Verilog file "C:/Users/David/clock_divide/main.v" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```

- At Project Summary, select Finish.

Creating Verilog Module

- At the project menu, you should see x3s100e-4cp132. Right click on this and select “New Source”



Select source type, file name and its location.

File name: blink

Location: C:\Users\[redacted]First_Project

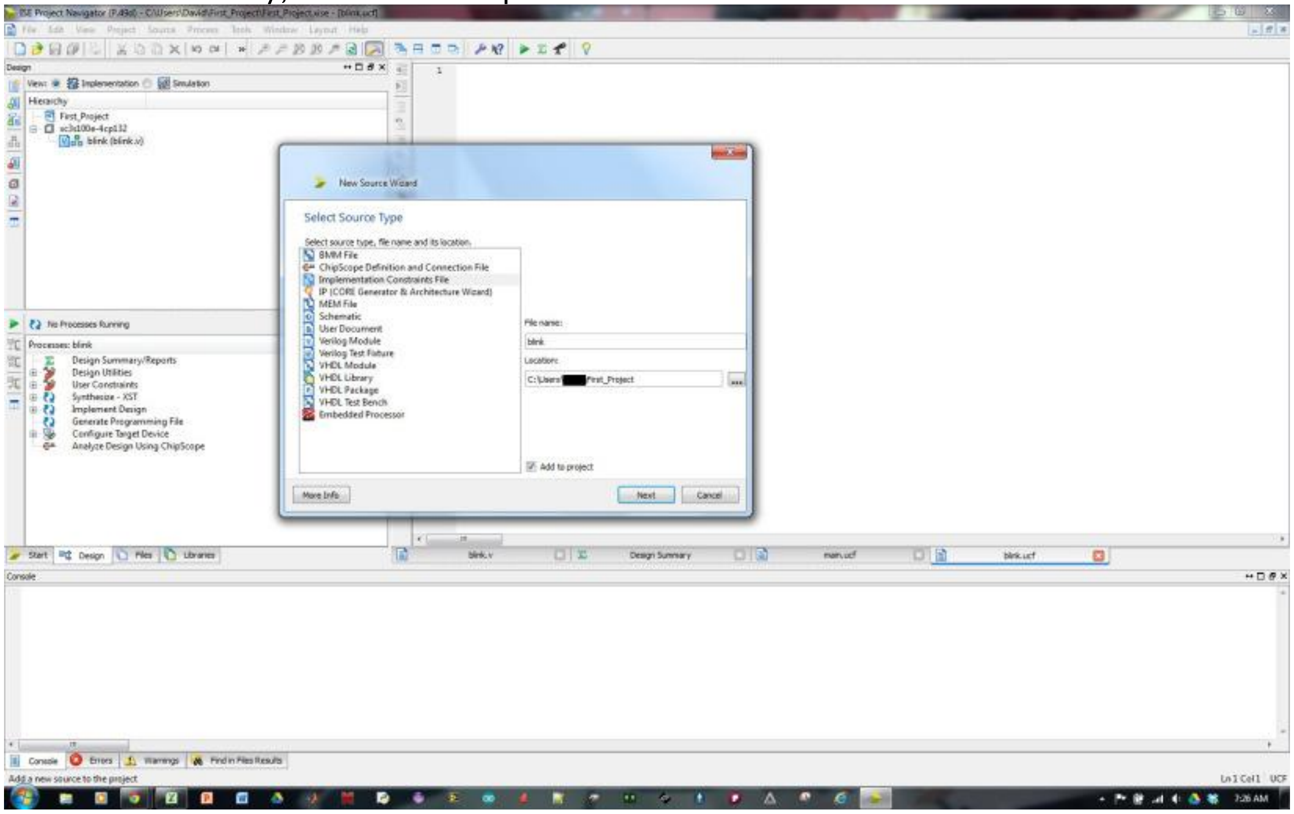
[✓] Add to project

- At the new source menu, select Verilog module and set the name to blink.
- When prompted at the define module, just click next. We’ll define the inputs and outputs later on. Afterwards select finish.
- In your blink.v file, copy and paste the following code into the file.
- Make sure to hit save.

Creating UCF File

Although we created the module, the FPGA does not know which pins on it should be connected to the module’s inputs and outputs. This is where the UCF file comes in.

- Right click you blink.v file and select “New Source”
- In your new source menu, select ” Implementation Constraints File.” We’ll name this file “blink”. Don’t worry, this will not replace the blink.v file.



File name: blink

Location: C:\Users\[redacted]First_Project

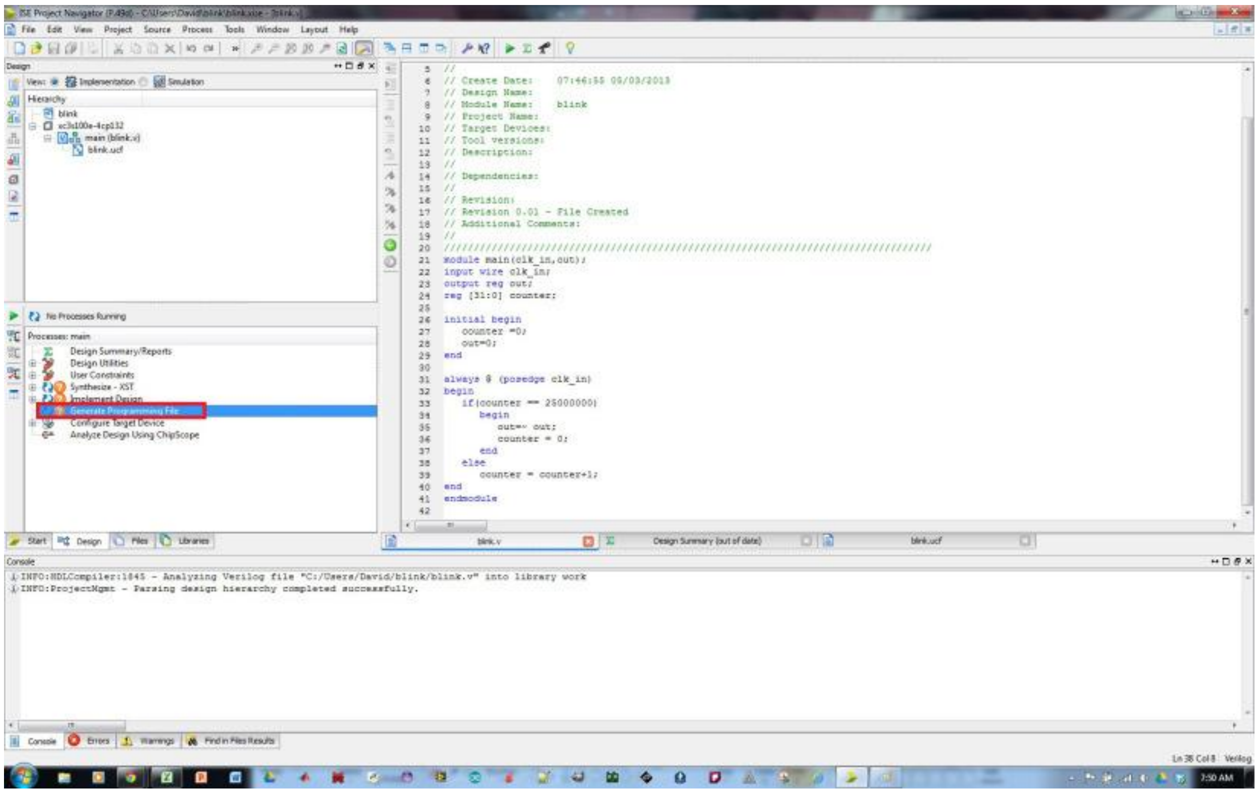
[✓] Add to project

- Hit next and select finish
- In your blink.ucf file, copy and paste the following code.
- Make sure to hit save.

Creating Bit File

Okay, so we’re almost ready to program- I mean ” implement”- our digital design on the FPGA. FPGAs are usually programmed using a bit file. We’re going to create a bit file using the Web ISE.

- Click on blink.v in your project navigator
- Your should see ” Generate Programming File” in the Processes menu. Click on it.



```
//
// Create Date:    07:46:55 09/09/2013
// Design Name:
// Module Name:    blink
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module main(clk_in,out);
input wire clk_in;
output reg out;
reg [31:0] counter;

initial begin
   counter =0;
   out=0;
end

always @ (posedge clk_in)
begin
   if(counter == 25000000)
      begin
         out=~ out;
         counter = 0;
      end
   else
      counter = counter+1;
end
endmodule
```

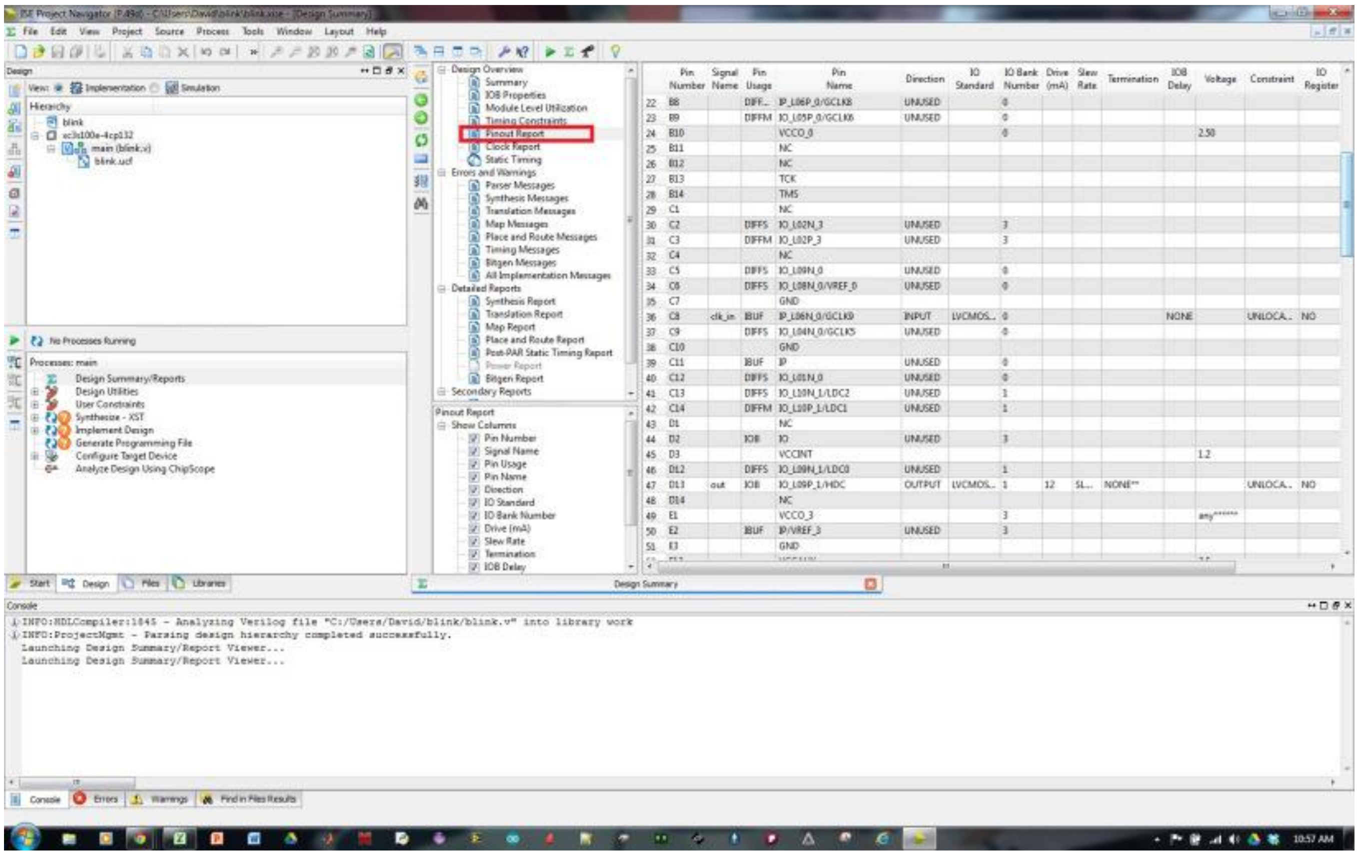
```
INFO:HDLCompiler:1845 - Analyzing Verilog file "C:/Users/David/blink/blink.v" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

- Wait for a little bit for the ISE to finish creating the bit file.

Checking Connections

It is very important to check to see if the FPGA really routed the inputs and outputs correctly. To do this…

- Click on your blink.v file
- In the Processes menu, select " Design Summary/Reports"
- In Design Overview click on " Pinout Report"
- Check to see if "clk_in" is connected to B8 and "out" is connected to M5.



	Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termination	IOB Delay	Voltage	Constraint	IO Register
22	B8		DIFF...	IP_L06P_0/GCLK8	UNUSED		0							
23	B9		DIFFM	IO_L05P_0/GCLK6	UNUSED		0							
24	B10			VCCO_0			0					2.50		
25	B11			NC										
26	B12			NC										
27	B13			TCK										
28	B14			TMS										
29	C1			NC										
30	C2		DIFFS	IO_L02N_3	UNUSED		3							
31	C3		DIFFM	IO_L02P_3	UNUSED		3							
32	C4			NC										
33	C5		DIFFS	IO_L09N_0	UNUSED		0							
34	C6		DIFFS	IO_L08N_0/VREF_0	UNUSED		0							
35	C7			GND										
36	C8	clk_in	IBUF	IP_L06N_0/GCLK9	INPUT	LVCMOS...	0				NONE		UNLOCA...	NO
37	C9		DIFFS	IO_L04N_0/GCLK5	UNUSED		0							
38	C10			GND										
39	C11		IBUF	IP	UNUSED		0							
40	C12		DIFFS	IO_L01N_0	UNUSED		0							
41	C13		DIFFS	IO_L10N_1/LDC2	UNUSED		1							
42	C14		DIFFM	IO_L10P_1/LDC1	UNUSED		1							
43	D1			NC										
44	D2		IOB	IO	UNUSED		3							
45	D3			VCCINT								1.2		
46	D12		DIFFS	IO_L09N_1/LDC0	UNUSED		1							
47	D13	out	IOB	IO_L09P_1/HDC	OUTPUT	LVCMOS...	1	12	SL...	NONE**			UNLOCA...	NO
48	D14			NC										
49	E1			VCCO_3			3					any******		
50	E2		IBUF	IP/VREF_3	UNUSED		3							
51	E3			GND										

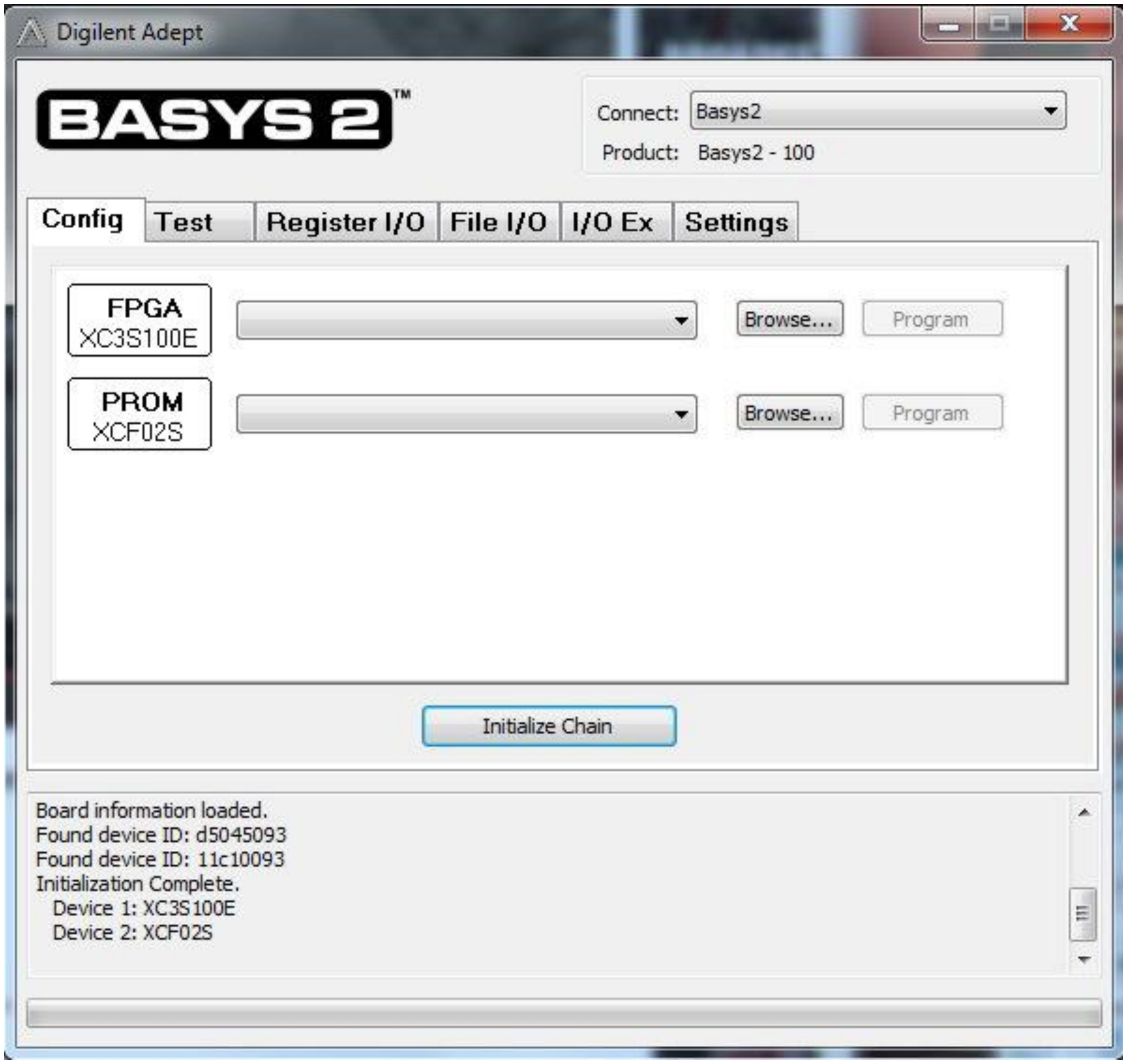
```
INFO:HDLCompiler:1845 - Analyzing Verilog file "C:/Users/David/blink/blink.v" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
Launching Design Summary/Report Viewer...
```

- As shown above, the ISE did not correctly route the connections. This can be solved by resaving your UCF file and clicking the "Generate Programming File" option again.

Uploading the Digital Design To The FPGA

We're at the final part of the tutorial!

- Open Digilent Adept
- Connect the Basys 2 to your computer and turn on it on.



```
Board information loaded.
Found device ID: d5045093
Found device ID: 11c10093
Initialization Complete.
   Device 1: XC3S100E
   Device 2: XCF02S
```

- You can upload the bit file to the main FPGA or the board's re-programmable flash memory. Since we want the board to remember the bit file whenever we turn on or off the board, we'll program the bit file to the reprogrammable flash (PROM)
- Click browse button next to the PROM option
- Navigate to your project folder. The bit file should be inside the folder.
- Hit "program" next to browse
- After the bit file is uploaded to the board, turn off then turn on your board. You should see LD0 blink every 500ms.

How does it work?

First thing, we need to use the 50MHZ clock on the board to tell the FPGA when 500ms has occurred. But how? Well, frequency is really counts per second. Using unit conversion and an onboard counter , we can find how many counts it takes for 500ms to occur.

$$\frac{50{,}000{,}000}{seconds} * 500ms = 25,000,000$$

Now that we know the counts, how do we use it in our design? We can use a register, a variable inside verilog, to function as our counter. During the positive edge of the clock, we tell the design to increment the counter by 1. When counts reaches 25 million, the design resets the counter to 0 then invert the output of the LED, which is initially set to off.

Source: http://coolcapengineer.wordpress.com/2013/05/06/tutorials-getting-started-with-fpgas/