Vivado Walkthrough ECGR 2181 - Fall 2015

Intro

In this walkthrough we're going to go through the process of creating a project, adding sources, writing vhdl, simulating the design, and creating a bitsteam (as if you were going to put it on an FPGA) of a prime number detector. Vivado can be pretty complex if you just stick with the basics, it's pretty simple.

Creating a Project

First open Vivado. You should have something like this in front of you.

Flow <u>T</u> ools <u>₩</u> indow <u>H</u> elp			Q- Search commands
VIVADO.	Productivity. Multiplied.		E XILINX ALL PROGRAMMABLE.
uick Start			Recent Projects
Create New Project	Open Project	Open Example Project	project 1 /home/Jarks/ecgr2181/computer_assignment_a/. project_1 /home/Jarks/project_1 ecgr2181_test
asks			/home/larks/ecgr2181_test
Manage IP	Open Hardware Manager	Xilinx Tcl Store	
Ś	8		
ocumentation and Tutorials	Quick Take Videos	Release Notes Guide	
ci Console			I

Select the "Create New Project" button.

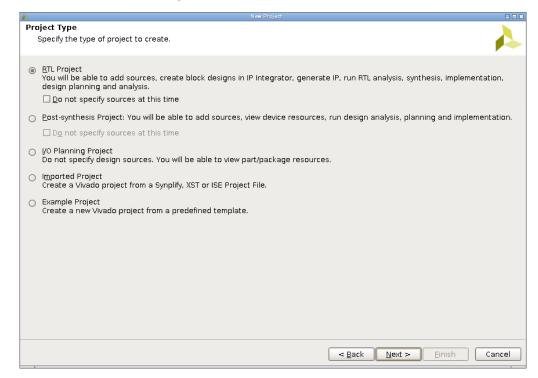
The first window just says that it's a wizard etc and you can press next. The next window you want to input your project name and the path to the project directory.

2	New Project				
Project Name					
Enter a name for your project and specify a directory where the project data files will be stored.					
Project name:	prime_number_detector	0			
Project location:	/home/larks/ecgr2181/	2 []			
🗹 Create projec	t subdirectory				
Project will be cre	eated at: /home/larks/ecgr2181/prime_number_detector				
	Seck Next > Einish Can	cel			

I have Linux directory paths but your's should look something like this. **Name your project** "prime_number_detector" and make sure you've enable "Create project subdirectory." Remember to always give your projects clear meaningful names and make sure that they're placed somewhere meaningful. As a side note, it's generally best to not have white space in directory or file names. In the past spaces in file names or directories has caused errors with Xilinx tools. It's a good idea to have a ecgr2181 directory and then subdirectories in it for each project/lab/etc like:

```
ecgr2181\cad1\
ecgr2181\cad2\
ecgr2181\project1\
etc...
```

Once you've filled out your project name and clicked next, you'll see a screen asking you about about Project Type. You'll want to select "RTL Project."



After clicking selecting "RTL Project" and then next, you'll be at the "Adding Sources" screen. At this point in time, we won't be adding sources, but **you need to make sure that "Target language" and "Simulator language" are both "VHDL."**

New Project	
Add Sources Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source	
file on disk and add it to your project. You can also add and create sources later.	
Press the 🔸 button to Add Files, Add Directories or Create File	
Scan and add RTL include files into project	
Copy sources into project	
☑ Add sources from subdirectories	
Target language: VHDL - Simulator language: VHDL -	
< Back Next > Einish Cancel	31

The next screen should be the "Adding Existing IP (optional)" screen which we can just click next.

The screen after this this is important if you're going to be putting the design on an FPGA (which we assume we will in this example). For this, you'll need to go to Digilent's <u>website</u> and get the xdc file for the Basys3. It'll be in a zip file and you'll need to unzip it. I suggest putting it in the project directory or you can also use the "Copy constraints files into project" checkbox, either one works.

👃 New Project						
Add Constraints (optional)						
Specify or create constraint files for physical and timing constraints.	r					
Constraint File Location Basys3_Master.xdc /home/larks/ecgr2181						
	< Back Next > Einish Cancel					

Next you need to select the (FPGA) part the tools will be targeting. The Basys3 uses a XC7A35T-1CPG236C FPGA.

A				New Proje	±				
Default Part									
Choose a default Xilinx part or board for your project. This can be changed later.									
Select: 🔷 Parts 📓	Boards								
Produ <u>c</u> t category:	General Pu	urpose 🔻	Packaq	ge: C	pg236	•			
<u>E</u> amily:	Artix-7	-	Spee <u>d</u>	grade:	1	-			
S <u>u</u> b-Family:	Artix-7	-	<u>T</u> emp o	grade: 🛛	:	-			
			<u>S</u> i Revi	sion: 🛛	All Remaining	•			
				Reset A	ll Filters				
Search: Q-									
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers	PCle
🔷 xc7a15tcpg236-1	236	106	10400	20800	25	45	2	2	1 5
xc7a35tcpg236-1	236	106	20800	41600	50	90	2	2	1 5 1 5 1 5
🔷 xc7a50tcpg236-1	236	106	32600	65200	75	120	2	2	1 5
4		ITTT							
						< <u>B</u> ack	Next >	<u>E</u> inish	Cancel

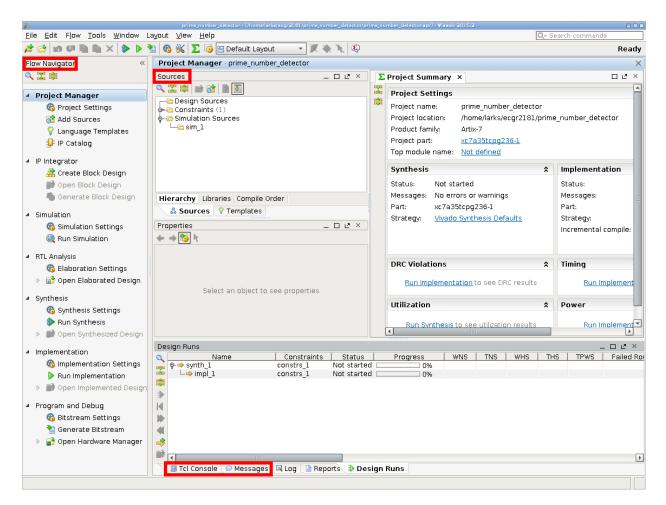
After selecting the right part, click next and that should bring you to a summary window.

A	New Project	
λ.	New Project Summary	
VIVADO.	③ A new RTL project named 'prime_number_detector' will be created.	
	▲ No source files or directories will be added. Use Add Sources to add them later.	
	🛕 No Configurable IP files will be added. Use Add Sources to add them later.	
	① 1 constraints file will be added.	
	 The default part and product family for the new project: Default Part: xc7a35tcpg236-1 Product: Artix-7 Family: Artix-7 Package: cpg236 Speed Grade: -1 	
XILINX ALL PROGRAMMABLE-	To create the project, click Finish	
	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

Then you're done setting up your new project! Now for the fun stuff.

Vivado Interface Overview

So now that the project is set up, let's look at the Vivado interface some.



There's a couple of key things we need to note:

- Flow Navigator
- Sources
- Tcl Console and Messages
- In the Flow Navigator, we'll mostly be using: Add Sources, Simulation and Generate Bitstream. There's a lot of intermediate steps, but those are the main ones that we'll be using. I would like to note that Synthesizing a design can help catch errors that aren't syntax related (like libraries missing etc).

In the sources section, I don't think you'll ever need to switch away from the sources tab or the hierarchy sub tab. It's a pretty good way to look at the code and see how multiple source files are related.

The Tcl Console and Messages are good for figuring out what went wrong. Depending on the error I've found useful information in either/or.

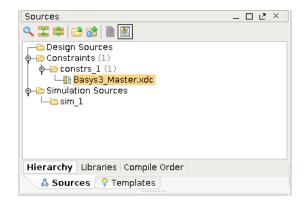
As a note, if there's something you'd prefer to change with Xilinx's built in editor you can access that via "tools" -> "options" in the "general" left vertical tab. There's the "Shortcuts" tab, which you can view/edit shorts for the editor and various other parts of Vivado. (Just an FYI)

We'll learn more about some of the interfaces as we use them in our project.

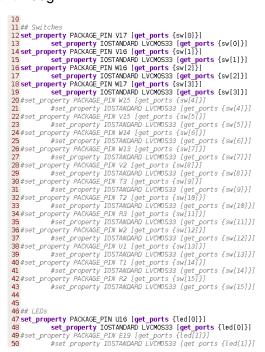
Writing VHDL/Implementation

Now that we've had a short intro the interface, let's start using it.

First we need to figure which FPGA pins we're using for the top level VHDL file. The constraint file will be in the Source section under "Constraints (1)" -> "constrs_1 (1)". Double click on it to open the file.



In the xdc file, there's a section for the clock, switches, leds, etc. We want to uncomment the first 4 switches (0 through 3) and the first led (0). Afterwards the file should look like:



We could change the names from sw and led but they suit our purposes. In the top level VHDL file they'll be "sw: in std_logic_vector(3 downto 0);" and "led: out std_logic_vector(0 downto 0)". Since we know our ports, let's create a VHDL file. Normally you don't use a vector/bus for a 1 bit value, but since led was part of a bus in this cause, we're just using one bit of that bus. In your own designs just use "std_logic" rather that "std_logic_vector (0 downto 0)".

In the "Flow Navigator" Panel, click the "Add Sources" button. You'll want to select "Add or create design sources" in the "Add Sources" window.

el

After that you'll see the "Add or Create Design Source" window. There's a green plus symbol which will create a pop up and then **select** "**Create File...**"

Add Sources
Add or Create Design Sources Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project.
Add Files Add Directories Create File Press the 🔸 button to Add Files, Add Directories or Create File
□ Scan and add RTL include files into project
Copy sources into project
☑ Add sources from subdirectories
< Back Next > Einish Cancel

Input your VHDL file name and click ok.

<u>k</u>	Add Sources	
Add or Create Design Sources		
Specify HDL and netlist files, or directories of and add it to your project.	ontaining HDL and netlist files, to add to your project. Create a new source file on disk	
Pre	Create Bource File Create a new source file and add it to your project. Elle type: File name: top File location: Cancel OK Cancel	
□ Scan and add RTL include files into project		
Copy <u>s</u> ources into project		
Add sources from subdirectories		
	< <u>Back</u> <u>N</u> ext > <u>Einish</u>	Cancel

It should look like:

Add Sources	= = ×
Add or Create Design Sources	
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file o and add it to your project.	n disk 🥂
Index Name Library Location 1 top.vhd xil_defaultlib <local project="" to=""></local>	
Scan and add RTL include files into project	
Copy sources into project	
Add sources from subdirectories	
< Back Next > Ein	nish Cancel

Since we know that **sw will be a 4 bit wide bus and led will be a 0 bit wide bus**, we put that into the VHDL "Module Definition" that is next.

<u>}-</u>					D effi	ne Module			
For M	Define a module and specify I/O Ports to add to your source file. For each port specified: MSB and LSB values will be ignored unless its Bus column is checked. Ports with blank names will not be written.								
Mod	Adule Definition								
[<u>E</u> ntii	ty name:	top				8		
,	A <u>r</u> ch	nitecture name:	Behavioral				8		
Į	I/O F	ort Definitions							
	+	Port Name	Direction	Bus	MSB	LSB			
	<u> </u>	sw	in 💌		3	0			
	+	led	out 💌		0	0			
			in 💌		0	0			
	Ŧ		in 🔽		0	0			
							OK Cancel		

Click ok and you'll have a vhdl source file in your "Sources" panel.

Sources	_ 🗆 🖻 ×					
🔍 🛣 🖨 🔁 🔂 📗 🛃						
စု– 🗁 Design Sources (1)						
🗛 🗁 Constraints (1)						
•						
🛛 🗋 🛏 🚹 Basys3_Master.xdc						
Simulation Sources (1)						
○- im_1 (1)						
Hierarchy Libraries Compile Order						
\delta Sources 🖓 Templates						

Now we need to write some VHDL. Since we have a 4 bit input, that means our input range is 0 to 15. The prime numbers in that range are: 2, 3, 5, 7, 9, 11, 13. So the output (led(0)) needs to be higher when the input is equal to those.

The VHDL file from the entity down (excluding the libraries) is:

```
34 entity top is
35 Port ( sw : in STD_LOGIC_VECTOR (3 downto 0);
36
             led : out STD_LOGIC_VECTOR (0 downto 0));
37 end top;
38
39 architecture Behavioral of top is
40
41 begin
42
          led(0) <=
43
44
                  '1' when (sw = "0010") else
                                                  -- 2
                 '1' when (sw = "0011") else
                                                  -- .3
45
                 'l' when (sw = "0101") else
46
                                                  -- 5
                  '1' when (sw = "0111") else
47
                                                  -- 7
                 'l' when (sw = "1001") else
                                                  -- 9
48
                 '1' when (sw = "1011") else
49
                                                  -- 11
                  '1' when (sw = "1101") else
50
                                                  -- 13
                  '0';
51
52
53 end Behavioral;
```

I run the synthesis tool (in the "Flow Navigator" as "Run Synthesis") after this to make sure there weren't any errors in the code.

Creating a Testbench

Test benches are extremely using for debugging circuit logic. Hardware isn't like software where you can easily run a debugger while it runs, so logic that isn't depending on complex I/O, simulation can useful for testing. Although if you don't use the idioms and try to be "creative" the simulation results can differ from implementation results (aka what gets loaded to the FPGA). Just a note that simulation isn't the final test, running it on the FPGA is.

First click on "Add sources" and select "Add or create simulation source."

A		
	Add Sources	
VIVADO.	This guides you through the process of adding and creating sources for your project	
	○ Add or <u>c</u> reate constraints	
	○ <u>A</u> dd or create design sources	
	Add or create simulation sources	
	○ Add or create DS <u>P</u> sources	
	○ Add existing <u>b</u> lock design sources	
	O Add <u>e</u> xisting IP	
XILINX	To continue, click Next	
	< Back Next > Finish	Cancel

Just like adding a source file, **you need to click the green plus symbol and select "Create File...**" on the "Create Simulation Sources" window.

Add Sources	
Add or Create Simulation Sources Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.	
Specify simulation set: as sim_1	
Add Files Add Directories Create File	
Press the 🔸 button to Add Files, Add Directories or Create File	
□ Scan and add RTL include files into project	
Copy sources into project	
☑ Add sources from subdirectories	
Include all design sources for simulation	
< Back Next > Einish Cance	

Then input "top_tb", make sure the file type is VHDL and then hit ok.

٨	Create Source File							
Create a new source file and add it to your project.								
<u>F</u> ile type:	M VHDL	-						
F <u>i</u> le name:	top_tb	8						
Fil <u>e</u> location:	🔂 <local project="" to=""></local>	-						
	OK Car	ncel						

After creating the VHDL file it should look like:

Add Sources									
Add or Create Simulation Sources									
Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.									
Specify simulation set: in sim_1									
□ Scan and add RTL include files into project									
Copy sources into project									
🗹 Add so <u>u</u> rces from subdirectories									
✓ Include all design sources for simulation									
< Back Next > Einish Canc	el								

Select Finish and then after that the "Module Definition" window will pop up. You don't want to add anything here. With test benches, the goal is to exercise a component and doesn't have anything to do with signals going in or out of the test bench entity. If a test bench had input ports you need another test bench to drive that test bench... **So leave these blank and hit Ok.**

Å	Define Module											
Define a module and specify I/O Ports to add to your source file. for each port specified: MSB and LSB values will be ignored unless its Bus column is checked. Ports with blank names will not be written.												
Module Definition												
<u>E</u> ntity name:	top_tb	8										
A <u>r</u> chitecture name:												
I/O Port Definitions												
+ Port Name	Direction Bus MSB LSB											
-	in 🔽 0 0											
↑ ↓												
+												
		OK Cancel										

Afterwards your source windows should have the top.vhd and the top_tb.vhd.

Sources	_ 🗆 🖻 ×
오 🔀 🖨 🔁 🚼 📓	
စု 📴 Design Sources (1)	
🛛 🗆 🗠 🖉 🕹 🖉 🖉 🖉 🖉 🖉 🖉	
🖕 🗁 Constraints (1)	
🖕 🕞 constrs_1 (1)	
- Basys3_Master.xdc	
• Simulation Sources (2)	
• • • • • • • • • • • • • • • • • • •	
Hierarchy Libraries Compile Order	
💑 Sources 🖓 Templates	

Open the top_tb VHDL file. It should be pretty barren. You'll want to add a component definition for top and add signals internal to the test bench. The internal signals would be sw and led, but also a counter signal. The reason for this is that a counter for n bits will go through all 2^n iterations of the n bits exercising all possibilities of n bits. Since we're using unsigned signals we needs to uncomment line 27 ("use IEEE.NUMERIC_STD.ALL;"). Then after tying the signals together, the architecture should look like:

```
38 architecture Behavioral of top tb is
39
40
           component top
41
                   Port (
42
                            sw : in STD_LOGIC_VECTOR (3 downto 0);
43
                           led : out STD_LOGIC_VECTOR (0 downto 0)
44
                  );
45
           end component;
46
47
           --signals internal to the test bench
48
49
                           std_logic_vector(3 downto 0);
           signal sw:
           signal led:
                           std_logic_vector(0 downto 0);
50
51
           --counter used to provide input to the swtiches
52
           signal counter: unsigned(3 downto 0):="0000";
53
54 begin
55
56
          uut: top port map (
57
                   SW => S₩,
                   led => led
58
59
          ):
60
61
          --since VHDL is strongly typed we need to convert
62
           --unsigned to std logic vecter
63
          sw <= std_logic_vector(counter);</pre>
64
65
          --counter process
           --warning: will NOT work in hardware
66
67
           tb: process
68
69
70
           beain
                   wait for 20ns:
                  counter <= counter + 1;</pre>
71
           end process tb;
72
73 end Behavioral:
```

Running The Testbench

Now for the easy part, running it all. All you need to do is on the "Flow Navigator" panel select "Run Simulation" and when the menu pops up select "Run Behavioral Simulation." That should start up the simulator. Initially it should be really zoomed in so you'll need to zoom out a fair bit to see something reasonable. Another thing that might be helpful is to change the radix of the sw bus. All you have to do is right click on the signal where it has "Name" and "Value", (see picture to get a better idea) and then select "Radix" and then "unsigned decimal." It should look something like:

Name	Value	0 ns		50	ns		100 n:	5	I	150	ns		200 ns	5	12	250 ns		300 ns
•	0	i o		2	× 3	4	5	le l		Ż	8		10			2 13	14	15
•	0	(()	\square	1	0	1	0			0	(1)	0	(1)	¢	1)	Ð
• % cou:0]	0000	0000	0001	0010	0011	0100	0101	0110	01	11 X	1000	1001	1010	1011	(110	00 1101	1110	1111

As you can see, the circuit works as aspected. When the switches form a prime number the LED goes high. Sadly Vivado doesn't have a way to print waveforms so you have to take a screenshot of it to include it in a report/document/etc.

If there's a more complex circuit or multiple VHDL files, you can go into the scope panel and add those signals to the waveform.

Creating the Bitstream

To create the bitstream simple go to the "Flow Navigator" panel and select "Generate Bitstream" under the "Program and Debug" section.

Programming the FPGA

I haven't yet gotten a Basys3 so I can't say for certain if you need to DL the Digilent Adept software/drivers to program the FPGA but once I acquire one this section will be updated.