University of Hawaii
EE 361L

Getting Started with Artix-7
Digilent Basys3 Board

Lab 4.1

I. Create a Project

The project is to build a combinational circuit called a decoder which has

- Two inputs sw[1] and sw[0]
- Four outputs led[3], led[2], ... led[0]

and the following truth table

sw	led
00	0001
01	0010
10	0100
11	1000

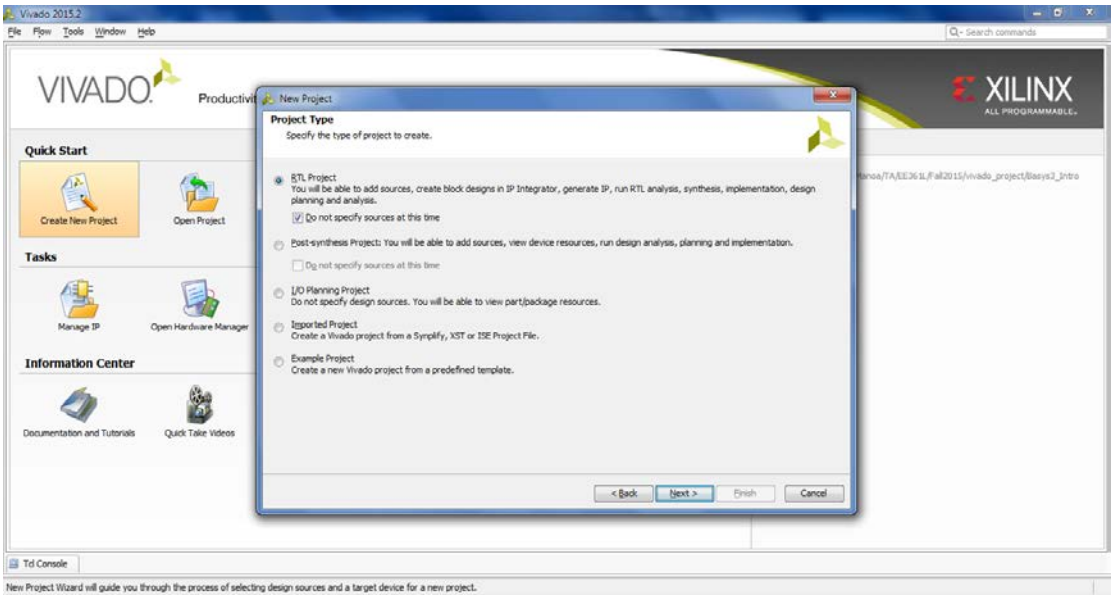
First, we'll create and simulate a verilog module for the decoder, and then we'll synthesize and download into the Xilinx Artix-7 FPGA of the Basys3 board:

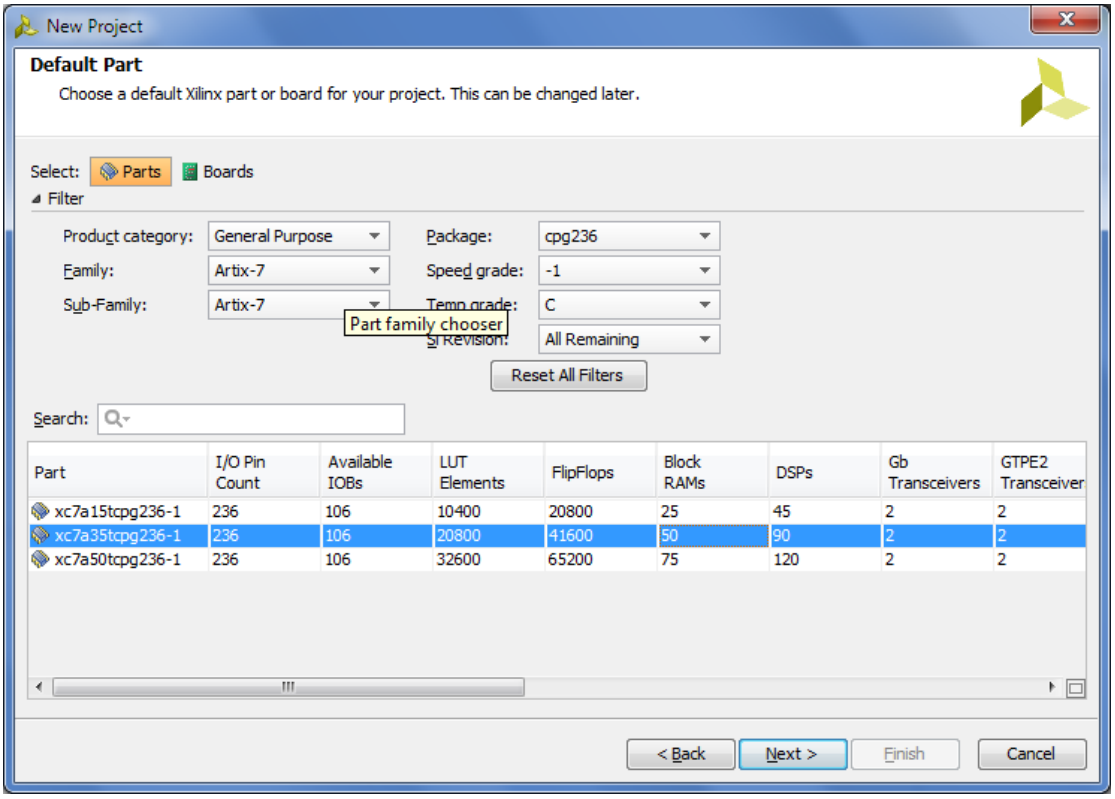
- Input "sw" will be connected to two slide switches
- Output "led" will be connected to four LEDs.

Step 1. Start Vivado Software (version 2015.2)

Step 2. Start a New Project

- Go to File and select New Project
- Name it Lab4.1A
- Select RTL Project as below



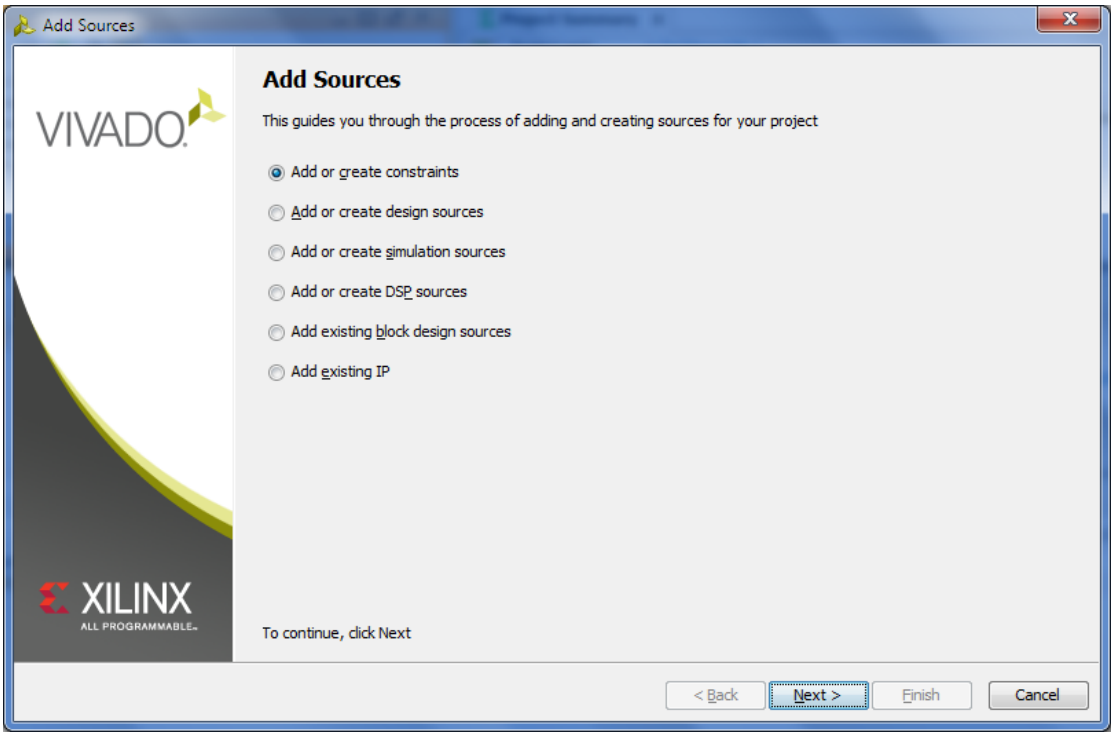


Step 3. Fill in the properties in the table as shown below:

- Product category: **General Purpose**
- Family: **Artix-7**
- Sub-Family: **Artix-7**
- Package: cpg236
- Speed Grade: **-1**
- Term grade: **C**

Then choose the 2nd item, go on and Finish

Add Constraints: Under left column “Project Manager”, click “Add Sources”:



If you do not have a constraint file, you can download at:
https://reference.digilentinc.com/doku.php?id=basys3:basys3#basys3_resource_center
where you will see the GitHub link of Master XDC file~~a ZIP file~~:

Documentation

- **Schematic** – PDF
  - PDF Schematic of the PCB generated by Altium
- **Reference Manual** – Wiki PDF
  - Technical description of the Basys 3 and all of its features. The Wiki may contain more up-to-date information than the PDF.
- **Datasheet** – PDF

Design Resources

- **Master XDC** – ZIP
  - This package contains the master XDC that defines the pin constraints for every device on the Basys 3.

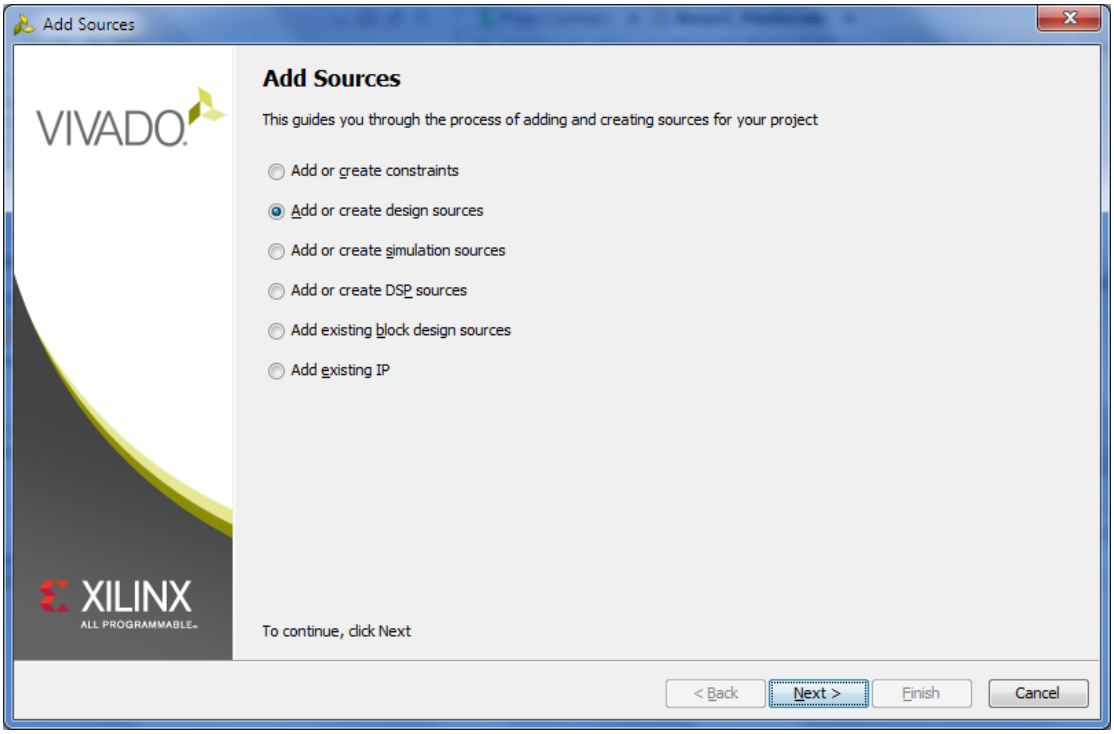
Design Resources	
Vivado Board Files	2015.1 and newer 2014.4 and older
Master XDC	GitHub
Documentation	
Primary IC	Artix-7 (XC7A35T-1CPG236C)
Reference Manual	
Schematic	Rev. C

FYI: here are two links of the tutorial and reference manual
Video Tutorial: https://www.youtube.com/watch?v=6_GxkslqbcU
Reference Manual: https://reference.digilentinc.com/_media/basys3:basys3_rm.pdf

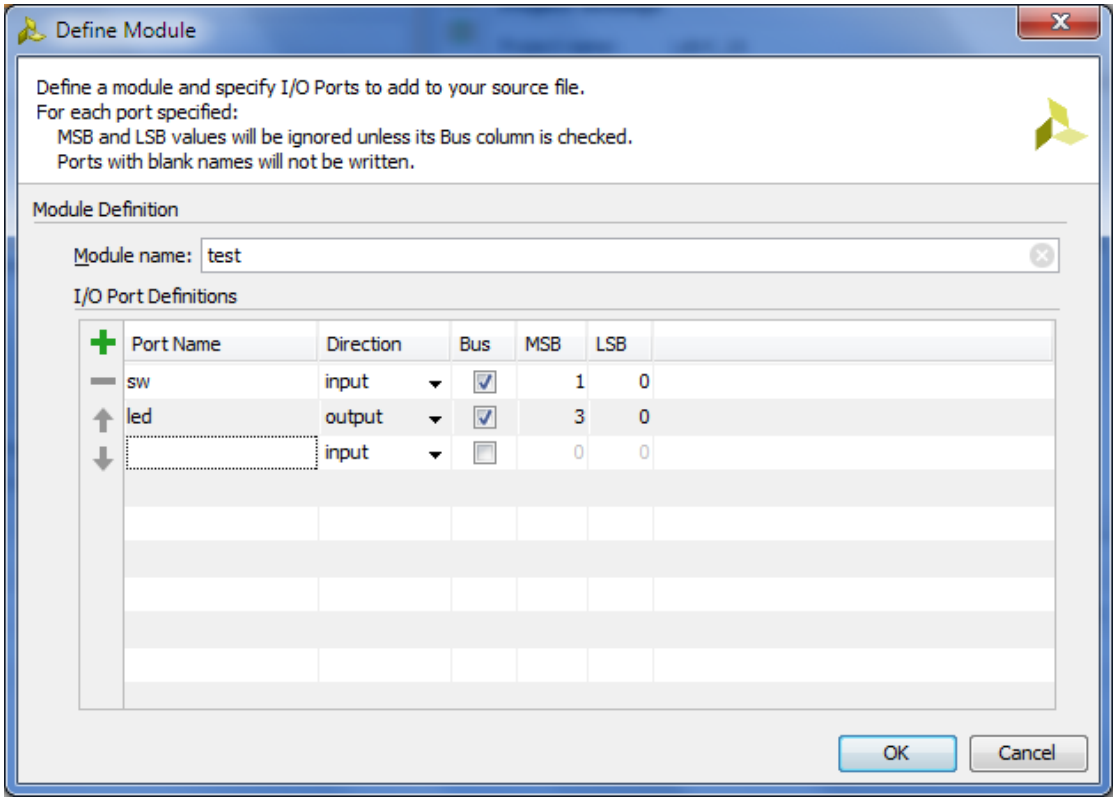
You can watch the tutorial for a few issues not covered in this lab.

III. Create an HDL Source

Step 1. In the created project, click “Add Sources”.



Step 2. We can add or create Verilog files. Here we will create a file by clicking the green-cross symbol and select “Create File…”. You can give it a name, and setup the input and output.

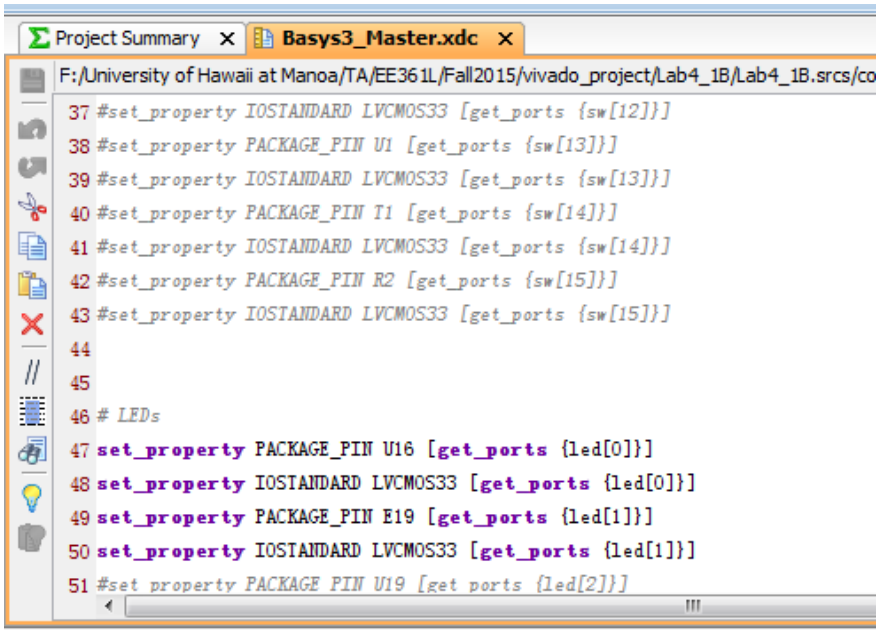


Once you have created a ".v" file, this file will be automatically set as "Top Module". You can right click the top module, then click "Add Sources…" to add or create additional ".v" files for other modules.

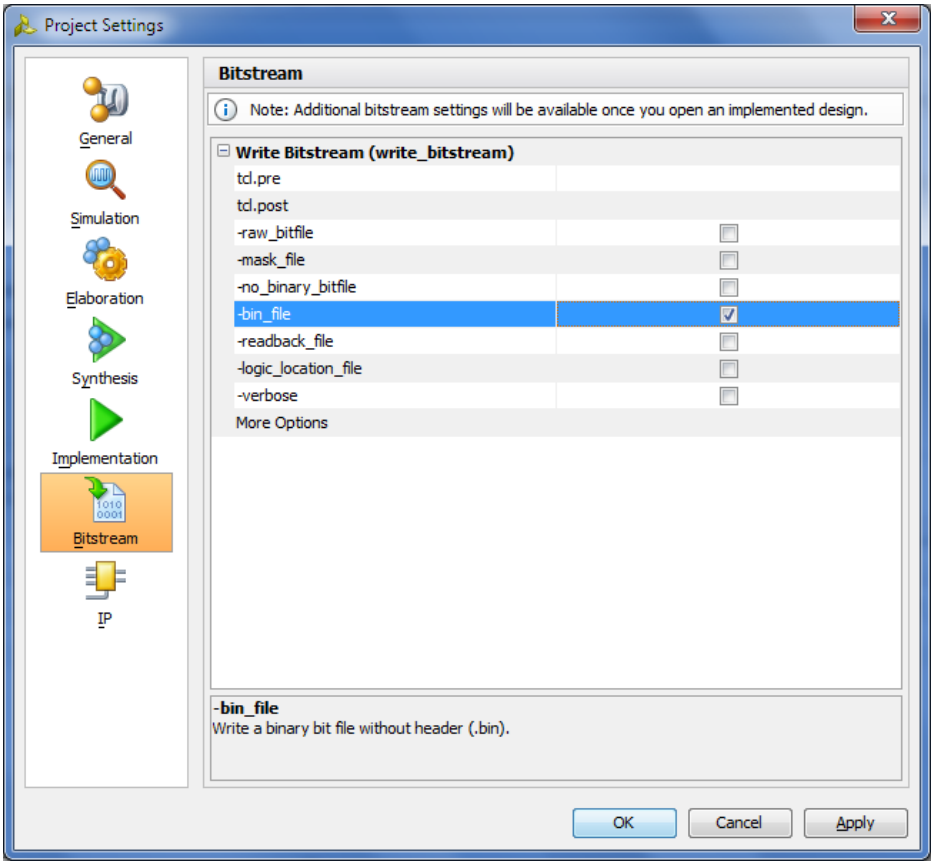
Step 3. Then a template of a verilog module will be created. It will have the ports of the module already written. Complete the module by fill in the following shown in blue

```
module decoder(
   input [1:0] sw,
   output [3:0] led
   );

reg [3:0] H;

assign led = H;

always @(sw)
        case(sw)
        2'b00: H=4'b0001;  // Note that 4'b0001 means "4" bits of "0001"
        2'b01: H=4'b0010;
        2'b10: H=4'b0100;
        2'b11: H=4'b1000;
        endcase

endmodule
```

Make sure the name of the inputs and outputs in the top ".v" file match the pin names in the constraint file (.xdc). You can uncomment the pins needed in the lab, from the constraint file (select a block of codes and press ctrl + /). Make sure that there is no unneeded symbols in the constraint file (e.g. as a reminder, we usually write "LED" before a block of codes to notify us the following codes setup LED pins, it should be written in the commented "#LED").

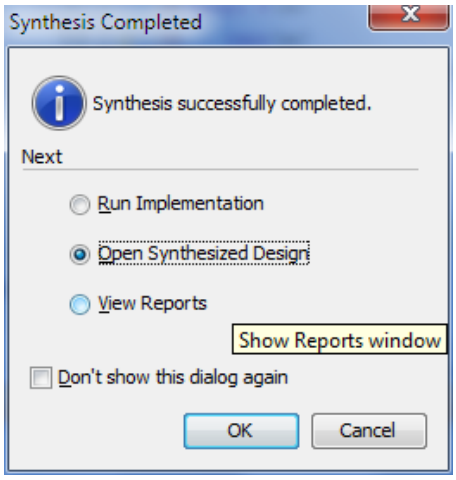


Step 4. We can setup to generate the ".bin" file as following:
Tools->Project Settings:



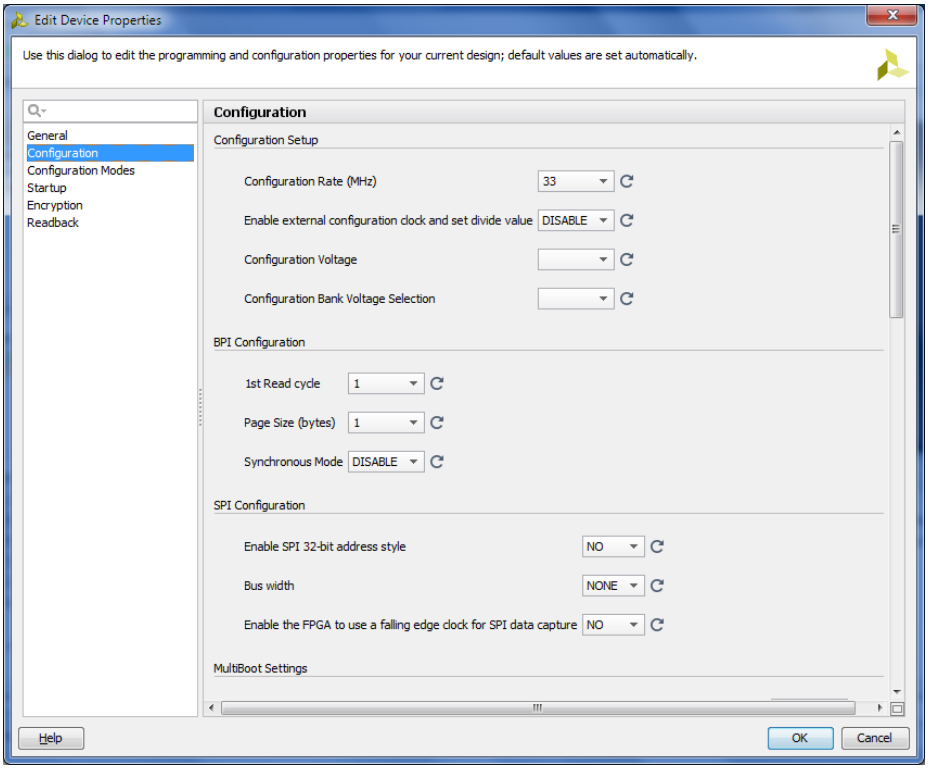
The -bin_file allows you to add memory configuration file to the FPGA flash device. If you do this, then when you power on the device, you can directly run the program. But here we just program the FPGA.

Step 5. Setup Device Properties:
In the left column (Flow Navigator), click "Run Synthesis" under the "Synthesis" section, this will generate the ".bit" and ".bin" file. When it finishes, select "Open Synthesized Design"

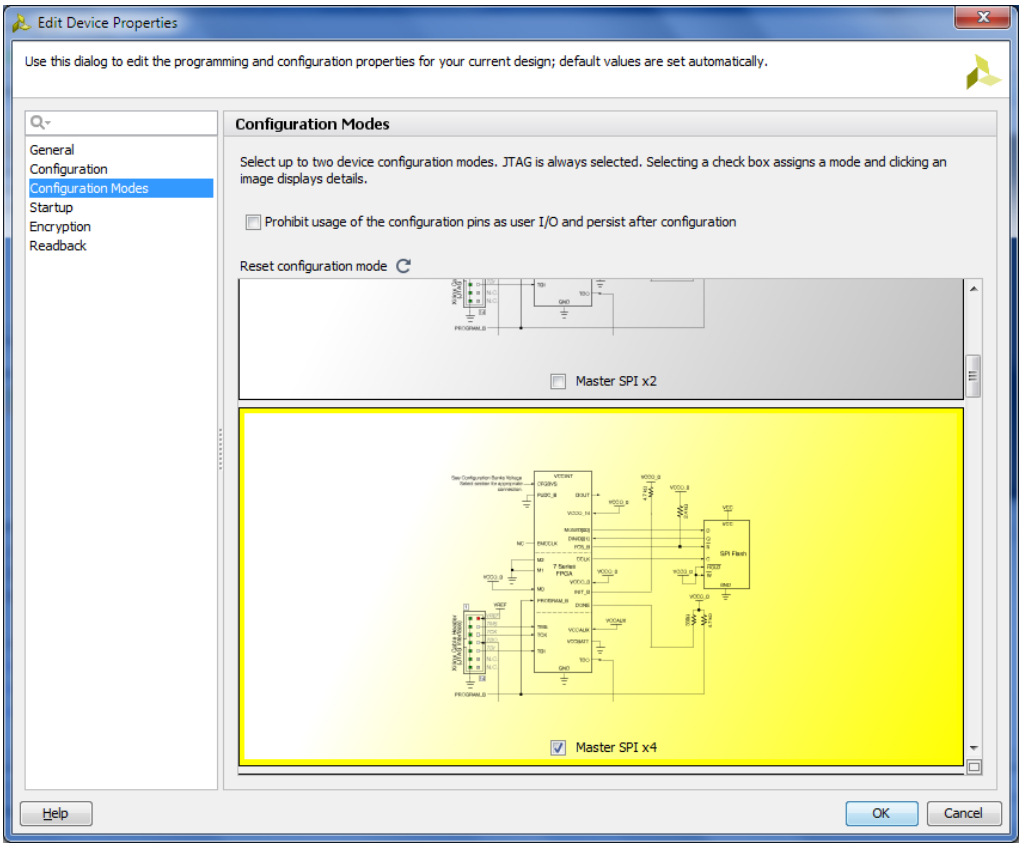


In the new window, select Tools->Edit Device Properties, configure as follows:

Set configuration rate to be 33:

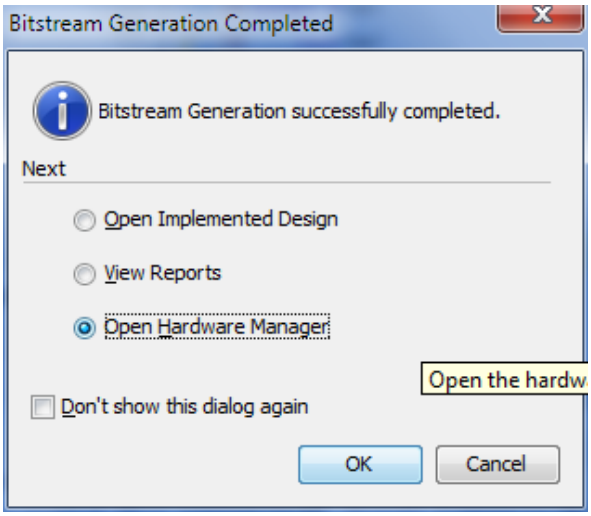


Choose the modes:

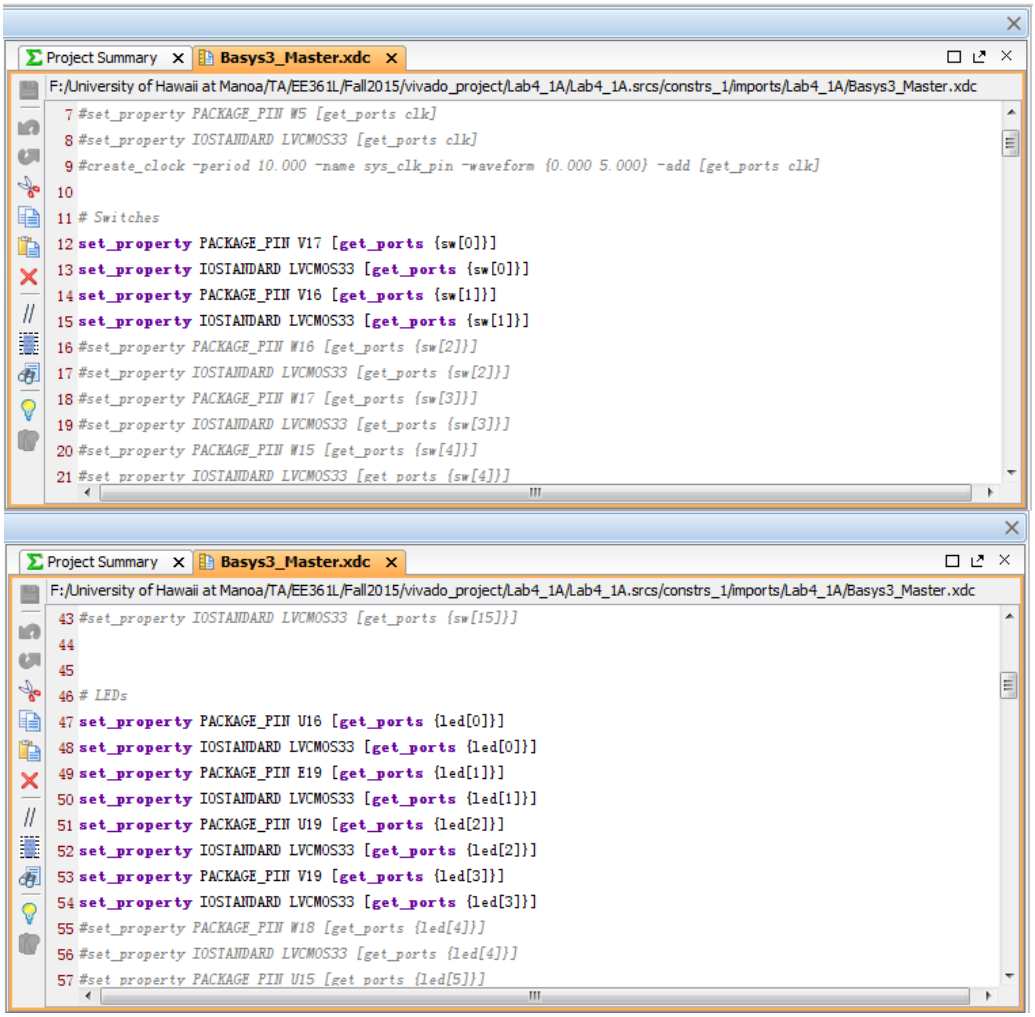


Then you can close the synthesis design and click “save” on the message box if it appears.

Step 6. Go to “Program and Debug” section, click “Generate Bitstream”, click “yes” on the message box. When the process completes, if you select “.bin” file before, both “.bit” and “.bin” files will be generated. Then select “Open Hardware Manager”.



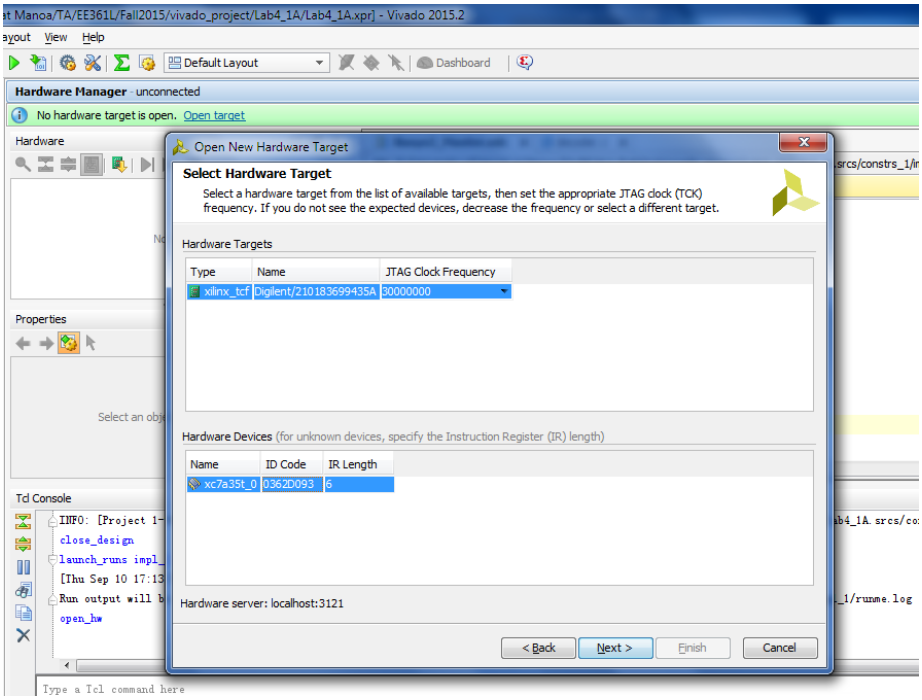
Uncomment the blocks shown below:



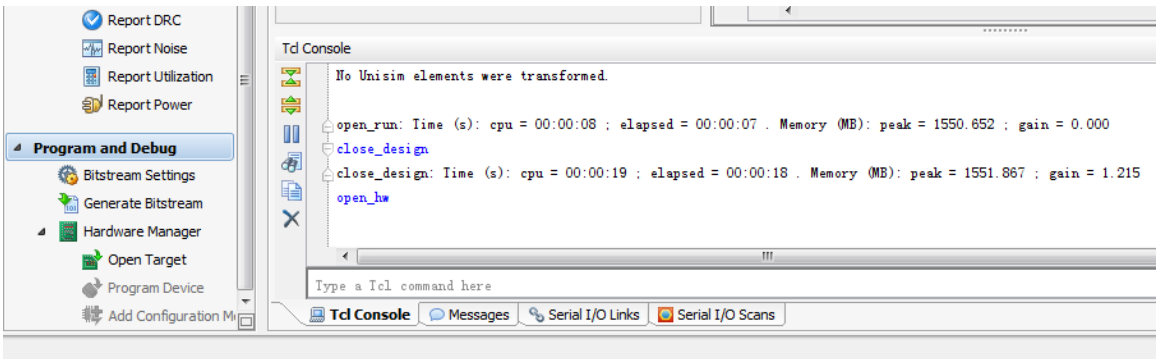
The I/O ports “sw” and “led” should be connected to pins as the above figures. Note that if you want to use the constraint file, the input and output name in the top “.v” file and the constraint file should be identical.

Decoder Signals	FPGA pins	Comments
sw[1]	V16	Sliding Switch SW1
sw[0]	V17	Sliding Switch SW0
led[3]	V19	LED LD3
led[2]	U19	LED LD2
led[1]	E19	LED LD1
led[0]	U16	LED LD0

If no window shows up, click “Open target” at the top, choose “Local server”, set up the “JTAG Clock Frequency”.



Make sure the jumper on the FPGA board is set to “JTAG” programming mode. Click “Program device”, then “Program”.



The following shows how the pins are connected to devices on the board

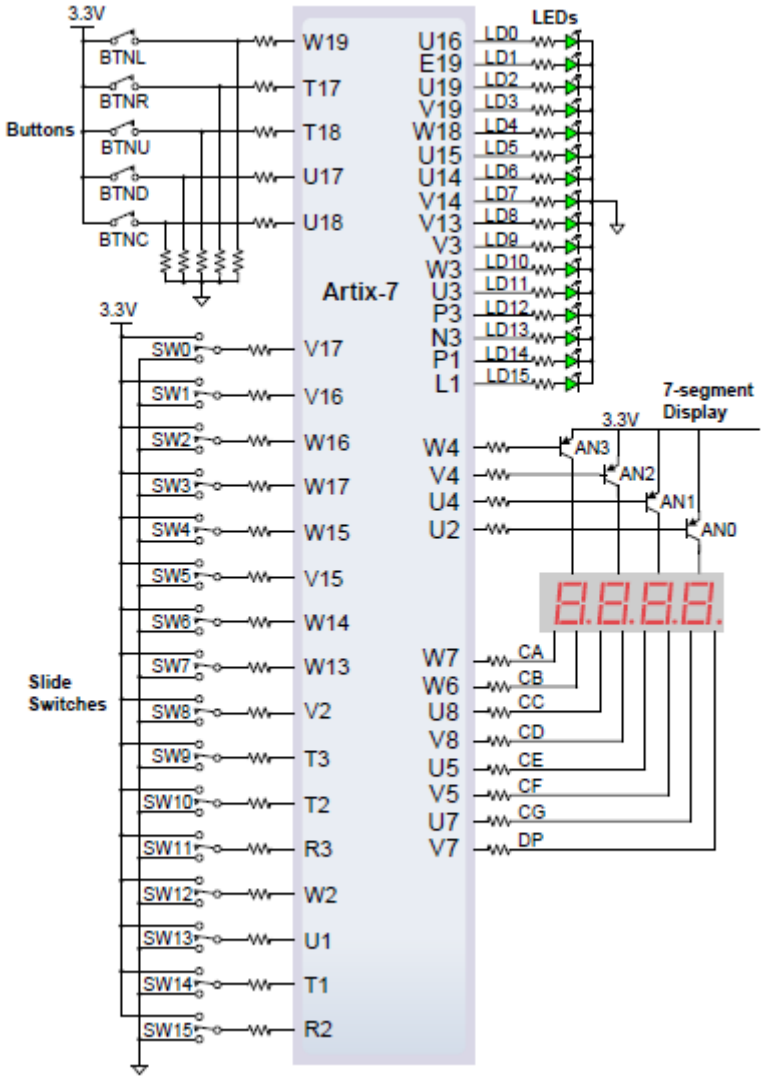


Figure 1. How switched and LEDs are connected to the FPGA.

Show the TA that it works.

IV. Sequential Circuit

We will implement a simple 2-bit counter circuit with a clock input and 2-bit output "led". It has a single Reset input, which when enabled will cause the counter to reset to led = 00 at the next clock transition. If the Reset = 0 then the counter increments every half-second.

The Basys3 board has an onboard clock at pin W5. We specify the counter's clock input in the (.xdc) file.

The outputs of "led" are connected to LED1 and LED0. We will connect the Reset (named btnC here) to push button 0.

Step 1. Create another Project and name it Lab4.1B.

Step 2. Add the following two new verilog modules.

First, add the new module "counter.v".
Then add the new "halfsec.v" module, which will be a submodule of counter. This can be done by right clicking "counter.v" and selecting to add a new verilog module.

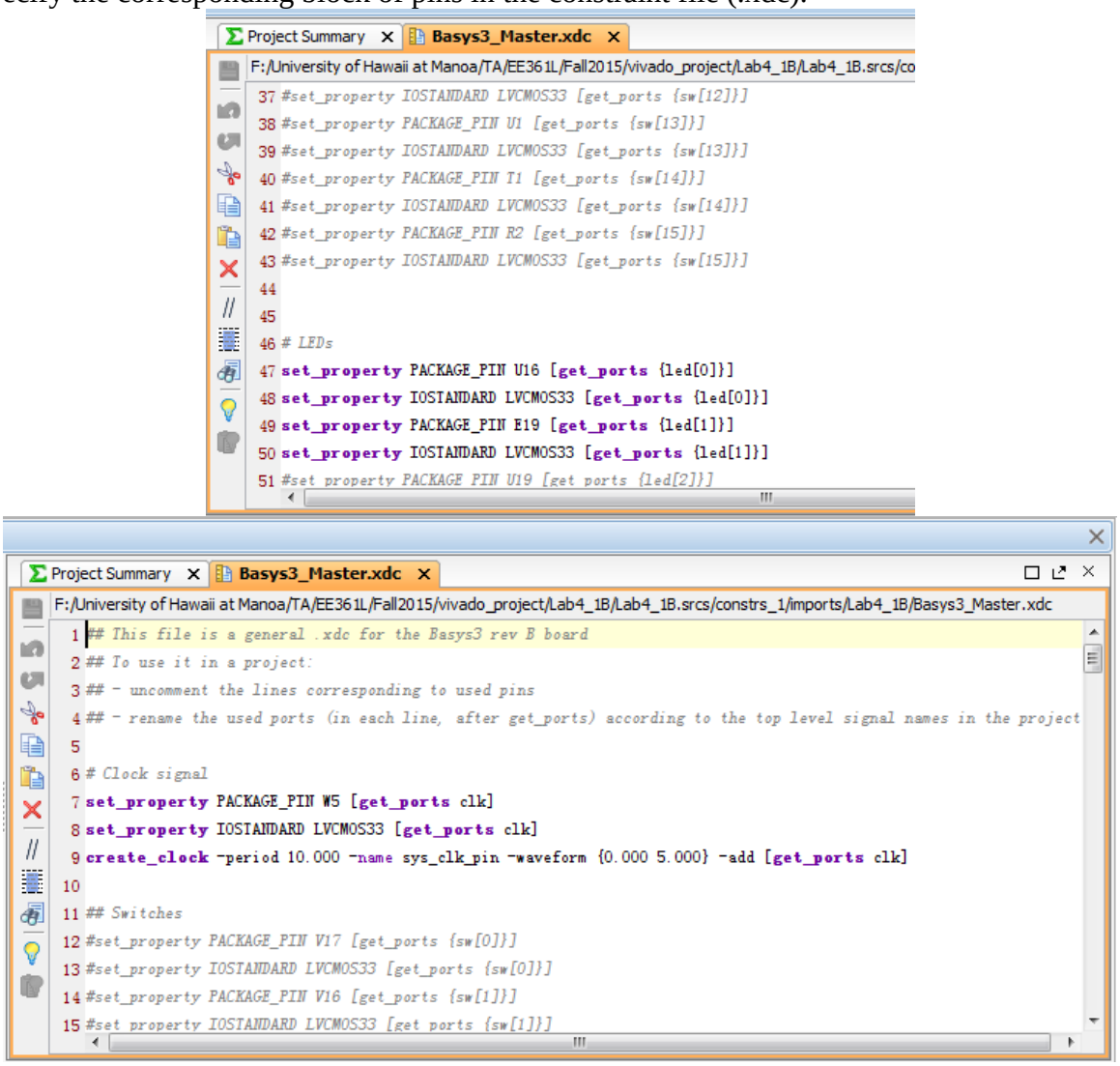
```
module counter(
   output [1:0] led,
   input clk,
   input btnC
   );

reg [1:0] state;
wire halfsec_elapsed;

halfsec  timer(halfsec_elapsed, clk, btnC);

always @(posedge clk)
  begin
  if (btnC == 1) state <= 0;
  else if (halfsec_elapsed == 1) state <= state+1;
  end

assign led = state;

endmodule
```

Add the following codes to “halfsec.v”

```
// Timer that indicates a half-second duration
module halfsec(
   output Y,
   input clock,
   input reset
   );

reg elapsed; // indicates that half second elapsed

// State of the timer.  Must be able to have more than 25 million values
// So we choose it to have 26 bits.  26 bits is a bit of an overkill but it'll work.
reg [25:0] state;

always @(posedge clock)
  if (reset == 1) state <= 0;
        else if (state == 25000000~~25000000~~50000000) state <= 0;
        else state <= state + 1;

always @(state)
  if (state == 25000000~~25000000~~50000000) elapsed = 1;
        else elapsed = 0;

assign Y = elapsed;

endmodule
```

Now synthesize as in Lab4.1A

Step 3. Specify the corresponding block of pins in the constraint file (.xdc).



In Lab4.1B we only need clock and LEDs on the board. Here is the table to connect the signals of the counter to the pins of the FPGA.

Counter Signals	**FPGA pins**	**Comments**
Clk (as clock)	W5	On-board clock
btnC(as Reset)	U18	Push button 0
led[1]	E19	LED LD1
led[0]	U16	LED LD0

Don’t forget to save the your project files

Step 4. Implement and Generate bit file. Then program the Basys3 board. Show the TA.