

University of Hawaii

EE 361L

Getting Started with Spartan 3E

Digilent Basys2 Board

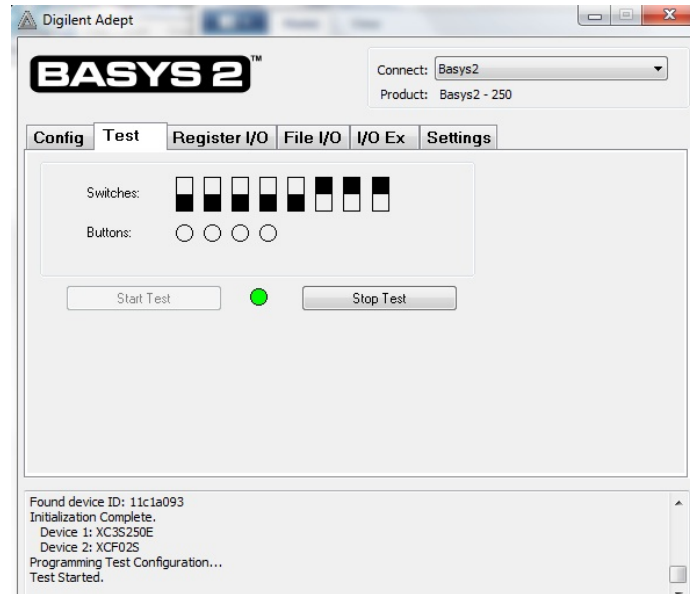
Lab 4.1

I. Test Basys2 Board

Attach the Basys2 board to the PC or laptop with the USB connector.

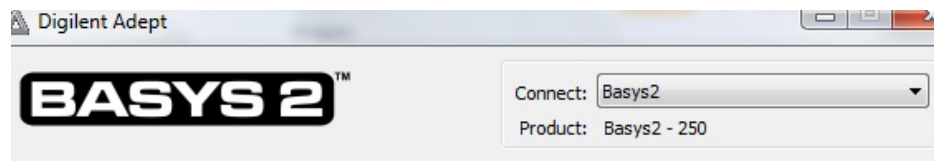
- Make sure the blue jumper (a blue plastic connector) is configured to PC rather than ROM.
- Also be sure the sliding power switch is on power “ON”

Start the Digilent Adept software tool as shown below. This can be found in the Digilent folder.



Select “Test” and “Start Test”. The board should light up, cause the 7 segment displays to count 0, 1, 2,...., and connect the sliding switches to LEDs, causing them to go on and off.

The Adept software will indicate the FPGA you have in the Basys2 board. For example, the following figure indicates that the FPGA is the 250 type (roughly 250 thousand gates). The boards that we have will be either the 100 or 250 type.



II. Create a Project

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

- Two inputs A[1] and A[0]
- Four outputs Y[3], Y[2], ... Y[0]

and the following truth table

A	Y
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 Spartan 3E FPGA of the Basys2 board. The input A will be connected to 2 slide switches and the output Y will be connected to four LEDs.

Step 1. Start ISE Software (version 13.1)

Step 2. Start a New Project

- Go to File and select New Project
- Name it Lab4.1A
- Top level source type: HDL

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

- Evaluation Development Board: **None specified**
- Product Category: **All**
- Family: **Spartan3E**
- Device: **XC3S100E or it's the 200 device depending on the Basys2 board**
- Package: CP132
- Speed Grade: **-4**
- Top-Level Source Type: **HDL**
- Synthesis Tool: **XST (VHDL/Verilog)**
- Simulator: **ISim (VHDL/Verilog)**
- Preferred Language: **Verilog**

Then go on and Finish

III. Create an HDL Source

Step 1. Under Project, select New Source -> Verilog Module and name it “decoder.v”, and be sure “Add to the Project” is checked.

Step 2. Configure the ports as shown below

Specify ports for module.
Module name: decoder

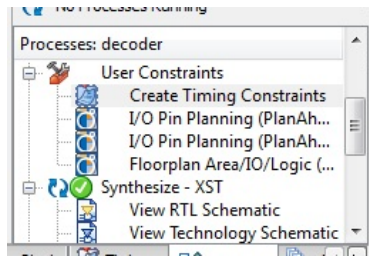
Port Name	Direction	Bus	MSB	LSB
A	input	<input checked="" type="checkbox"/>	1	0
Y	output	<input checked="" type="checkbox"/>	3	0
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		

More Info Next Cancel

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

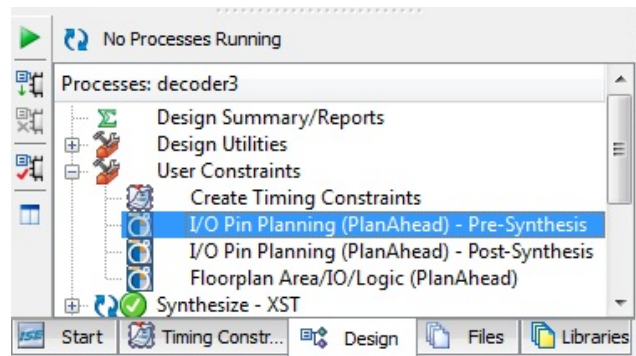
```
module decoder(  
    input [1:0] A,  
    output [3:0] Y  
);  
  
reg [3:0] H;  
  
assign Y = H;  
  
always @(A)  
    case(A)  
        2'b00: H=4'b0001; // Note that 4'b0001 means "4" bits of "0001"  
        2'b01: H=4'b0010;  
        2'b10: H=4'b0100;  
        2'b11: H=4'b1000;  
    endcase  
  
endmodule
```

Step 4. In the project window select Synthesize -XST



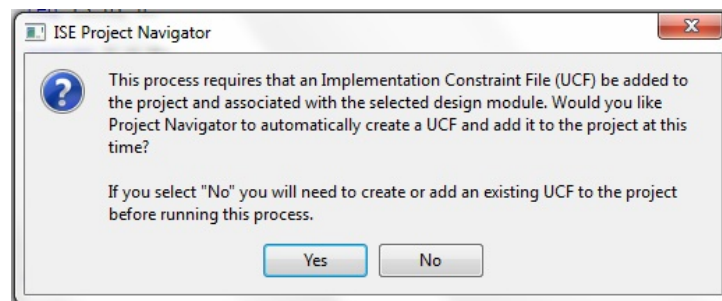
It'll try to synthesize. If you have any bugs, edit the verilog module source. You can always rerun the synthesizer by right clicking "Synthesize-XST" and selecting rerun. This is true for any of the processes, e.g., I/O Pin Planning (which is discussed next) – just right click and rerun.

Step 5. Go back to your Design



and select I/O Pin Planning (PlanAhead) – Pre-Synthesis. This will open the PlanAhead software, which will fill in your UCF with information about connecting your signals to pins.

It may also ask you for a UCF, which is the universal constraint file. Select "Yes" in the window below:

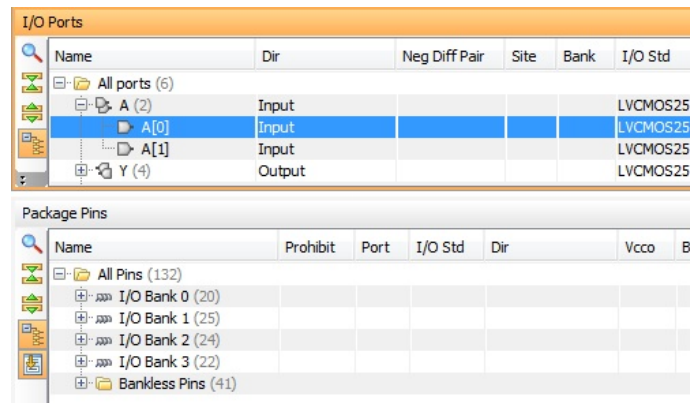


A UCF will specify, among other things, how the signals (inputs and outputs) of your circuit are connected to the pins of the FPGA. For example, your decoder circuit should have its input *A* connected to two sliding switches, and its output *Y* should be connected to four LEDs.

The software will complain that you have no clocks, so just select okay to get rid of that little window.

Note it may take a while for PlanAhead to get loaded, perhaps a few minutes.

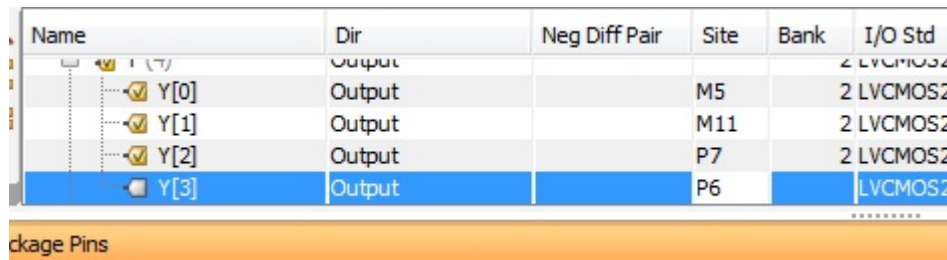
Step 6. The PlanAhead software will show the signals of your circuit and the pins.



The I/O ports A and Y should be connected to pins as follows. Note that a description of these pins and what devices they are connected to is given on page 11 of the Basys2 Reference Manual.

Decoder Signals	FPGA pins	Comments
A[1]	L3	Sliding Switch SW1
A[0]	P11	Sliding Switch SW0
Y[3]	P6	LED LD3
Y[2]	P7	LED LD2
Y[1]	M11	LED LD1
Y[0]	M5	LED LD0

In PlanAhead, under I/O ports, select the pins for the ports under “Site”, as shown below



In PlanAhead, after setting all the ports, then “Save Project”. This will save the UCF information into a file. If you make any changes, then “Save Project” so that the information can be used by the other software and in particular the Xilinx ISE.

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

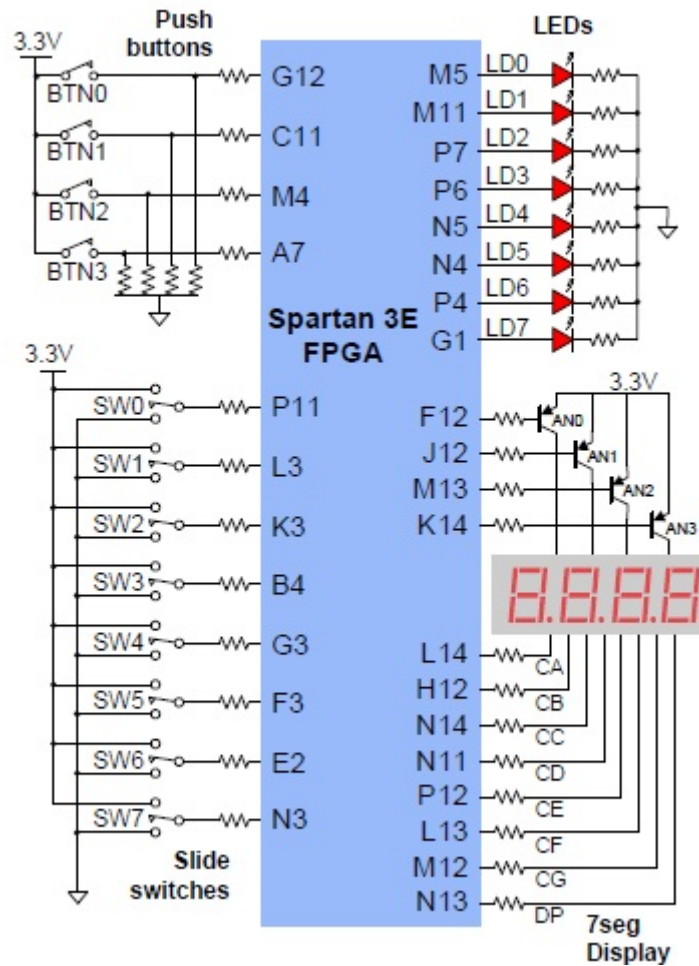
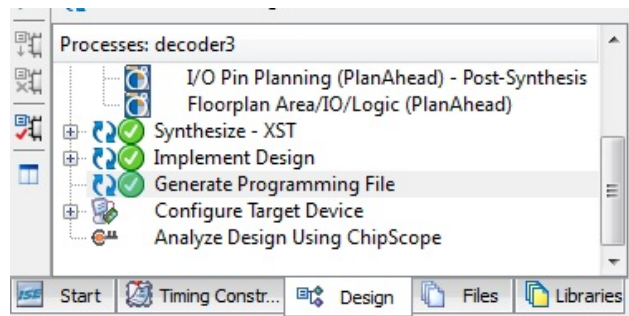


Figure 1. How switched and LEDs are connected to the FPGA.

Step 7. Double click “Implement Design”

Step 8. Double click “Generate Programming File”



This should create a bit file in the root folder of your project.

Step 9. Use the Adept software to download the bit file into the FPGA. Choose “Config” and find the bit file. It should be in the root directory of your project. Then “Program”.

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 Q. It has a single Reset input, which when enabled will cause the counter to reset to $Q = 00$ at the next clock transition. If the Reset = 0 then the counter increments every half-second.

The Basys2 board has an onboard clock at 50 MHz at pin B8. We must connect the counter's clock input to pin B8.

The outputs of Q are connected to LED1 and LED0. We will connect the Reset 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] Q,  
    input clock,  
    input reset  
);  
  
    reg [1:0] state;  
    wire halfsec_elapsed;  
  
    halfsec timer(halfsec_elapsed, clock, reset);  
  
    always @(posedge clock)  
        begin  
            if (reset == 1) state <= 0;  
            else if (halfsec_elapsed == 1) state <= state+1;  
        end  
  
    assign Q = state;  
  
endmodule  
  
// More on next page
```



```

// 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) state <= 0;
    else state <= state + 1;

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

assign Y = elapsed;

endmodule

```

Now synthesize

Step 3. Under User Constraints, select I/O Planning (PlanAhead) – Pre-Synthesis

Here is the table to connect the signals of the counter to the pins of the FPGA.

Counter Signals	FPGA pins	Comments
clock	B8	On-board clock 50 HHZ
Reset	G12	Push button 0
Q[1]	M11	LED LD1
Q[0]	M5	LED LD0

Don't forget to save the PlanAhead project.

Step 4. Implement and Generate bit file. Then program the Basys2 board. Show the TA.