
Seven-Segment Display Applications Using PYNQ

Objectives

After completing this laboratory assignment you will be able:

- To have a better understanding of behavioral modeling and hardware implementation
- To become familiar with modular top-down design methodology
- To become familiar with designing hardware from block designs
- To design a seven-segment decoder

Part I – Behavioral Modeling of a Seven-Segment Display

In this part, you will simulate a pattern decoder using the buttons, switches, and a single seven-segment display. Using switches, ***SW0*** and ***SW1*** as the pattern input**,** write a Verilog program that takes the binary input combinations of the switches and displays the decimal value of the binary switch combination on the seven-segment display. Table 1 shows the switch combinations with the expected output value on the seven-segment display.

NOTE: You will be using common-cathode 7-segment display which means that the segment needs to be HIGH to be turned ON.

Table 1: Input Combinations and Expected Output

SW1	SW0	BTN0	Display
0	0	0	OFF
0	0	1	0
0	1	1	1
1	0	1	2
1	1	1	3

When the ***SW*** combination is set and ***BTN0*** is pressed, the seven-segment display should show the decimal equivalent of the binary combination. When ***BTN0*** is released, the display should turn **OFF** (i.e., the display should not latch)**.**

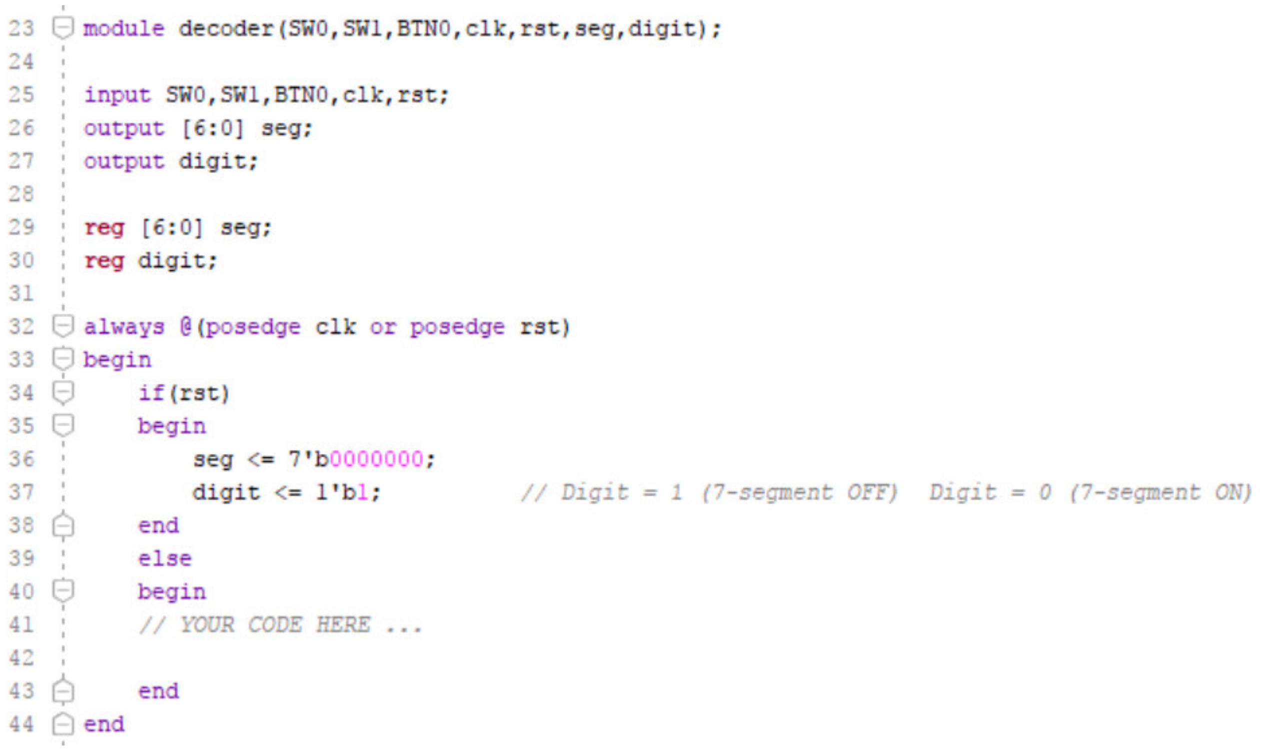
1. Create a new project using **Vivado**. Name the project ***decoder*** and select the device ***xc7z020clg400-1***. You can refer to Lab #1 on how to create a new project in Vivado.

2. Now you will create a new Verilog module and name it ***decoder***. At the very beginning of the newly created **decoder.v** file, change ***'timescale 1ns/1ps*** to ***'timescale 1ns/1ns***.

3. Now declare five input signals as ***"SW0", "SW1", "BTN0", "clk"*** and ***"rst"*** and two output signals namely "***seg***" of width 7, and ***"digit"***.

4. Make sure to save and save often **(CTRL+S).**

5. Now, write your own Verilog code for the seven-segment display, using Figure 1 as a template.



```
module decoder(SW0,SW1,BTN0,clk,rst,seg,digit);

input SW0,SW1,BTN0,clk,rst;
output [6:0] seg;
output digit;

reg [6:0] seg;
reg digit;

always @(posedge clk or posedge rst)
begin
    if(rst)
    begin
        seg <= 7'b0000000;
        digit <= 1'b1;          // Digit = 1 (7-segment OFF)  Digit = 0 (7-segment ON)
    end
    else
    begin
    // YOUR CODE HERE ...

    end
end
```

Figure 1: Verilog template for seven-segment display simulation

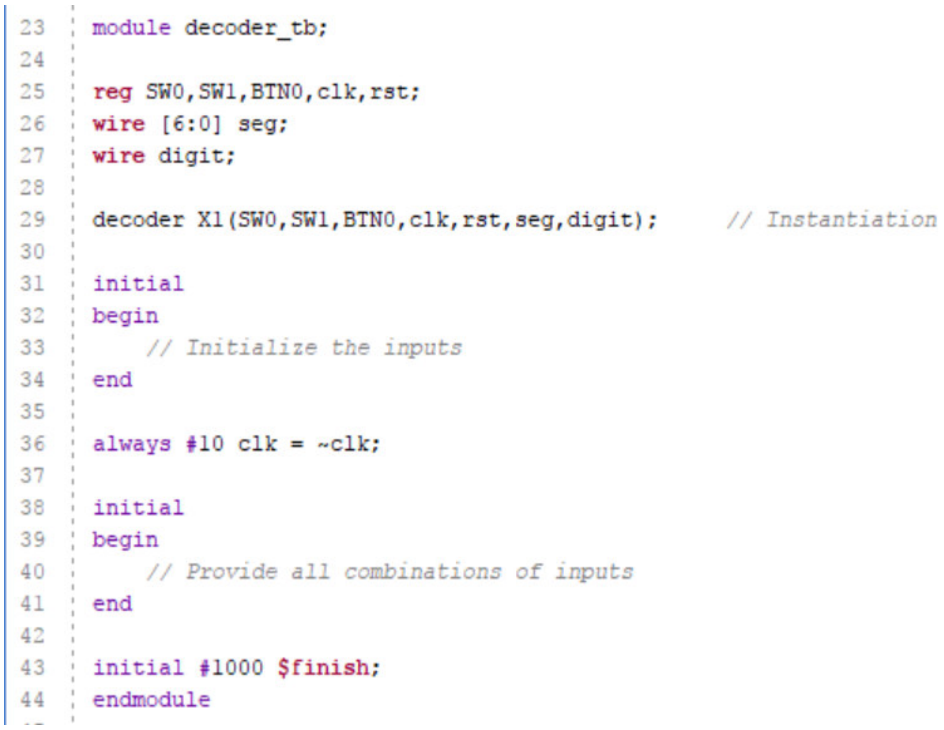
6. Now, synthesize your design by clicking on the ***Run Synthesis*** option. Make sure your design synthesizes successfully. Correct any errors and repeat step 6 if needed.

7. Once you have successfully synthesized your design, you need to create a simulation testbench by clicking the ***Add Sources*** option.

8. Select ***Add or Create Simulation Sources*** option and click **Next.** Name your testbench ***decoder_tb.*** You can refer to Lab #1 on how to create a simulation source.

9. Design your testbench to simulate all possible ***SW*** combinations by using the template in Figure 2.

10. Once your testbench is complete, click on the ***Run Simulation*** option, then select ***Run Behavioral Simulation*** from the given options.



```verilog
module decoder_tb;

reg SW0,SW1,BTN0,clk,rst;
wire [6:0] seg;
wire digit;

decoder X1(SW0,SW1,BTN0,clk,rst,seg,digit);      // Instantiation

initial
begin
    // Initialize the inputs
end

always #10 clk = ~clk;

initial
begin
    // Provide all combinations of inputs
end

initial #1000 $finish;
endmodule
```

Figure 2: Testbench template

11. Your simulation output should look like Figure 3 below.

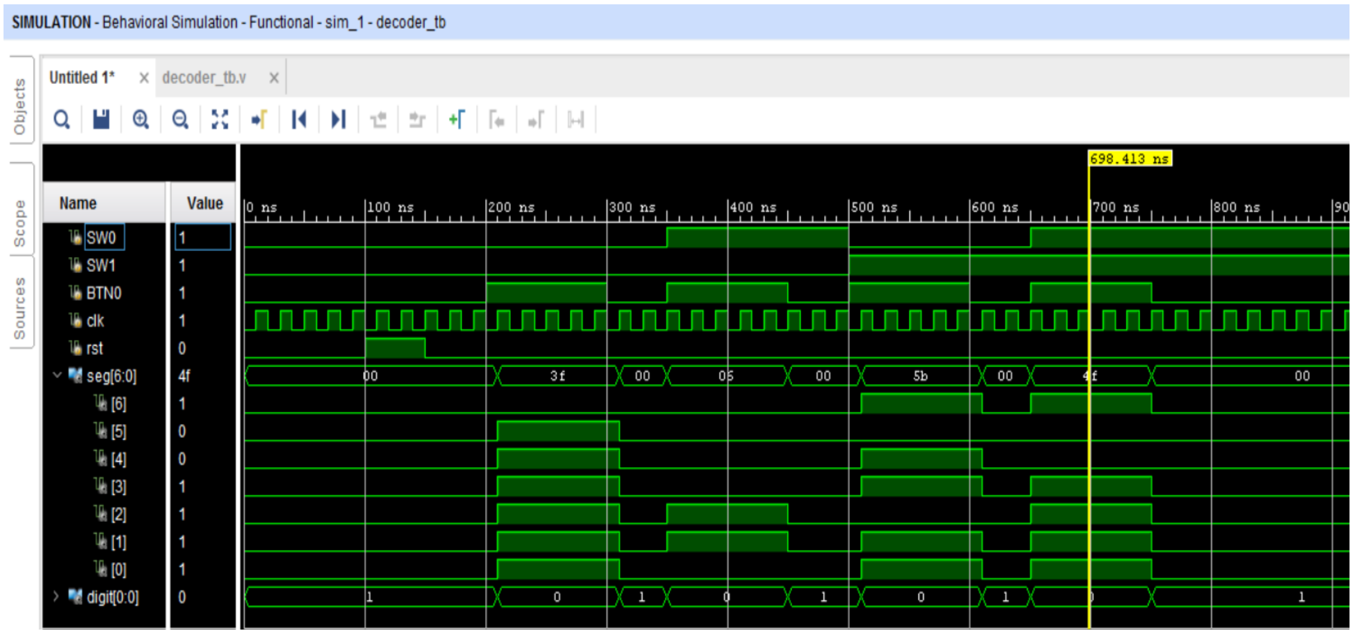


Figure 3: Seven-segment decoder simulation waveforms

12. You can now close the Behavioral Simulation. The select **File → Close Project** to close the current project.

Part II – Hardware Implementation & Modular Design

In this part, you will modify your code from Part I. Instead of only using one button and one switch, you will design your Verilog code to display on four seven-segment displays by pressing the four buttons on the PYNQ board. Each seven-segment display will correspond to one button (i.e., **BTN0** lights up the right-most seven-segment display and **BTN1** will light up the adjacent seven-segment display and so on). Make sure that when a button is released, the corresponding seven-segment display turns **OFF.** Only one seven-segment display should be on at any given time.

You will be using two Digilent Pmod (Peripheral Module) seven-segment displays like the one shown in fFgure 4. The schematic for this Pmod is shown in Figure 5. You can read the Pmod specification by downloading the document from the course website.

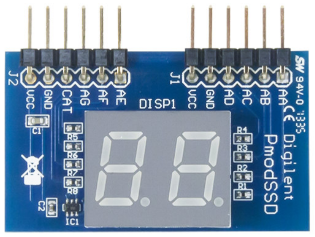


Figure 4: Digilent Pmod seven-segment display

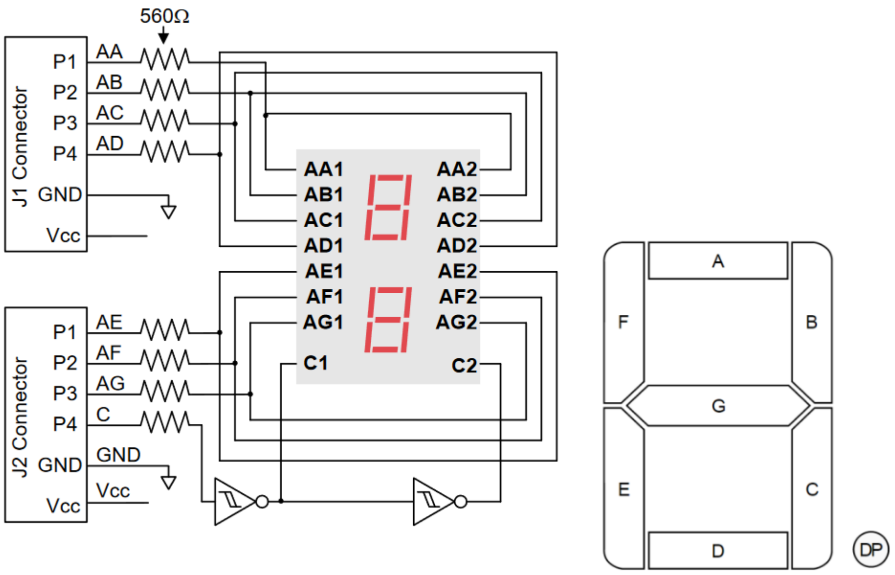


Figure 5: Pmod seven-segment display schematic

Each Pmod has two seven-segment displays, and each seven-segment display is selectable by setting the ***C*** pin high or low.

<u>NOTE</u>: These seven-segment displays are common cathode. More information is available in the datasheet: *https://reference.digilentinc.com/_media/reference/pmod/pmodssd/pmodssd_rm.pdf*
The Pmod specification can be found here:
https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf

The seven-segment displays will be connected to the PYNQ board by using two Pmod ***2x6 pin to dual 6-pin cables*** as shown in Figure 6.

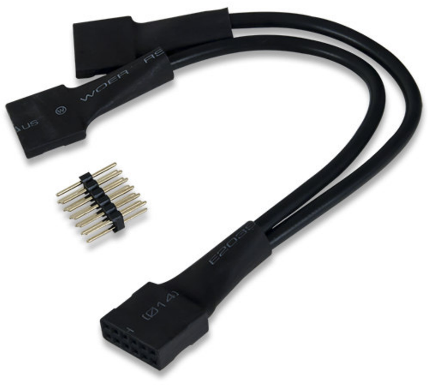


Figure 6: Pmod interface cable

Pay careful attention while connecting the Pmods to the PYNQ board. **CONNECTING THE DISPLAYS INCORRECTLY COULD DAMAGE THE HARDWARE.** See Figure 7 below for the proper way to connect the Pmods to the PYNQ board using the interfacing cables. The pin mapping to the PYNQ board for this connection configuration is shown in Table 2.

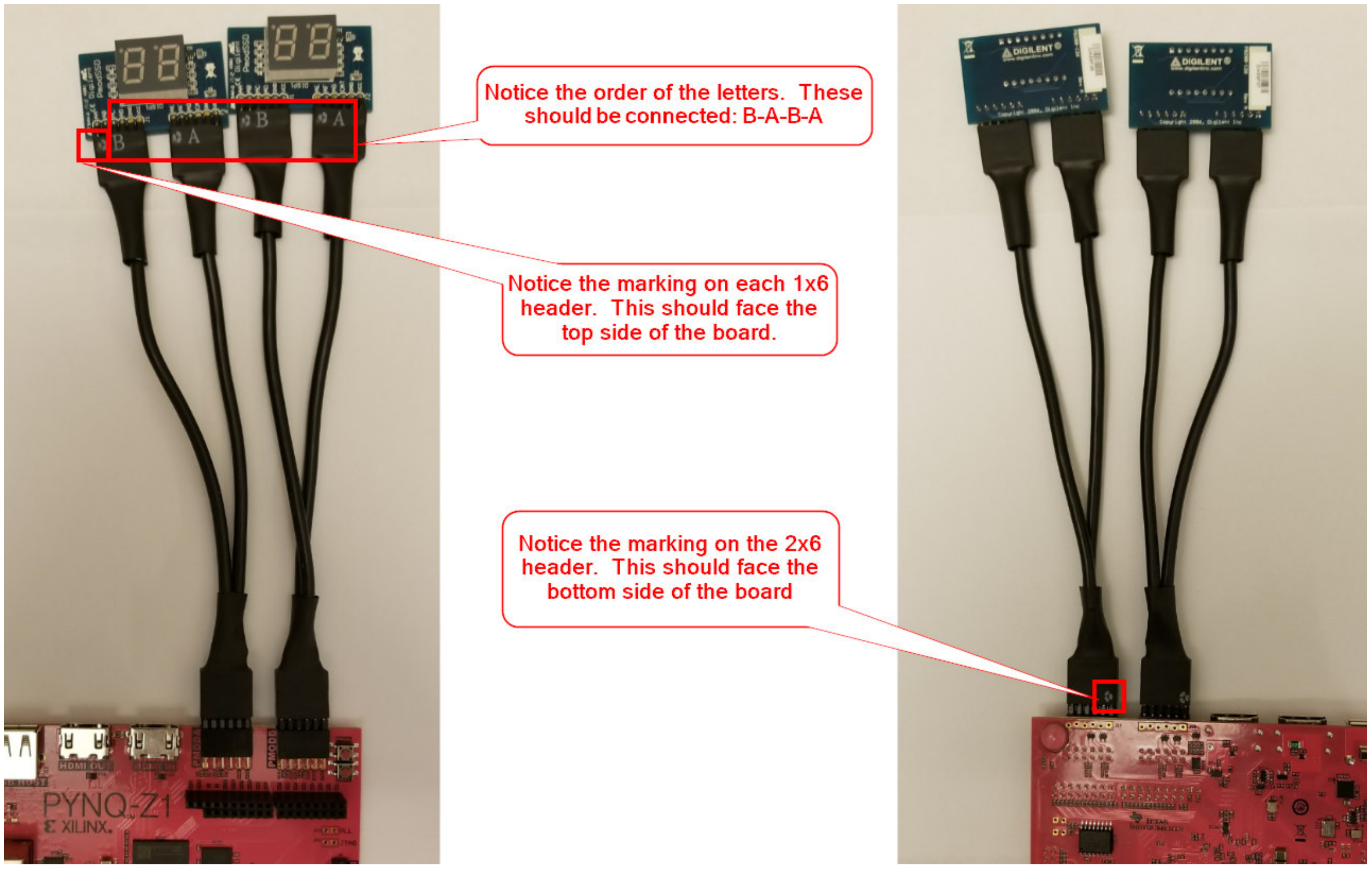

Notice the order of the letters. These should be connected: B-A-B-A
Notice the marking on each 1x6 header. This should face the top side of the board.
Notice the marking on the 2x6 header. This should face the bottom side of the board
B
A
B
A

Figure 7: Pmod connections to the PYNQ board

Table 2: Pmod to PYNQ pin mapping

Pmod A	PYNQ	Pmod B	PYNQ
AA	Y18	**AA**	W14
AB	Y19	**AB**	Y14
AC	Y16	**AC**	T11
AD	Y17	**AD**	T10
GND	GND	**GND**	GND
VCC	VCC	**VCC**	VCC
AE	U18	**AE**	V16
AF	U19	**AF**	W16
AG	W18	**AG**	V12
CAT	W19	**CAT**	W13
GND	GND	**GND**	GND
VCC	VCC	**VCC**	VCC

Previously, you wrote your Verilog code all in one module. This was fine before, but as your projects become more complex, you will find it easier and more organized to design using the modular top-down approach. From now on, you will be creating your Verilog code with a modular top-down approach. Your project will have a ***top*** level module from which all of the other modules are called.

1. Create a new Vivado project and name it ***decoder_p2***. Within this project, create two new Verilog files named ***"decoder_top", and "decoder".***

2. Write your own code for the seven-segment decoder described above. Don't forget to add and create a ***constraints*** file to your project. All inputs and outputs **MUST** be constrained!

3. Next, ***Run Synthesis, Run Implementation, Generate Bitstream,*** and then upload the ***Bitstream*** to the PYNQ board. Refer to Lab #1 for help with this process. If your ***.bit*** file is not already loaded in the browser, the ***.bit*** file can be found by navigating to the ***.runs*** folder within your **decoder** project.

4. Lastly, verify your design works as specified in the description.

5. Once you are done, slide the POWER switch to **OFF** position and then disconnect the USB cable from the computer. Close the **Hardware Manager** and **Vivado** applications.

Part III – Laboratory Exercise

In this part, you will create a ***time multiplexed*** seven-segment decoder using the block diagram shown in Figure 8 as a guide. You will need to create a multiplexer in order to display more than one seven-segment simultaneously. Design the multiplexer to enable each digit for ***8ms,*** or a ***125Hz*** refresh rate. Remember, these Pmods have a shared digit select pin. Most multi-digit displays would have a separate digit select pin for each digit. Table 3 shows the required outputs for the required input combinations.

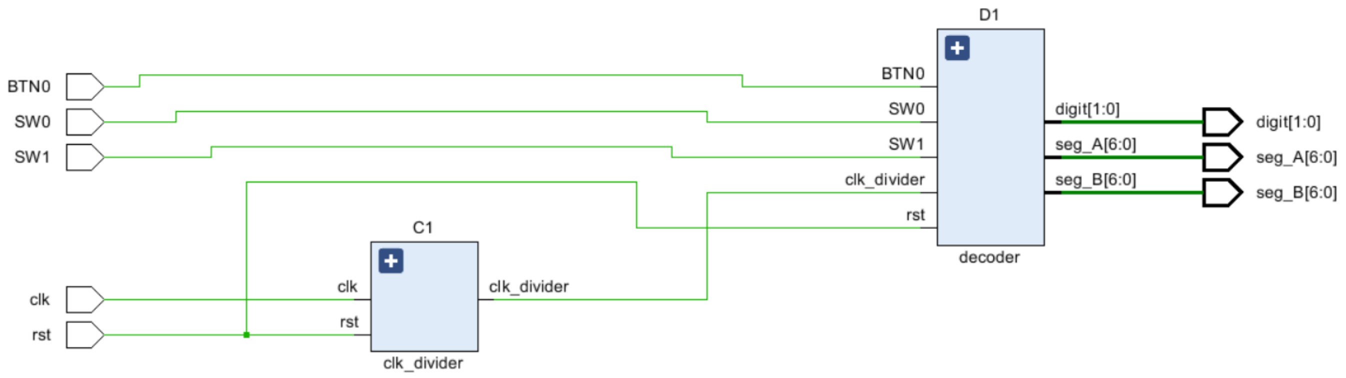


Figure 8: Part III block diagram

Table 3: Inputs and outputs

SW1	SW0	BTN0	Display
0	0	0	OFF
0	0	1	PYNQ
0	1	1	FPGA
1	0	1	IS
1	1	1	COOL

The design should work as follows:

1. Your design should show all digits **OFF** upon startup.

2. When you press **BTN0,** the appropriate word should display based on the switch configuration shown in table 3.

3. The word should remain on the display until a different switch configuration is set and **BTN0** is pressed, or **rst** is pressed. A reset should clear the display.

Laboratory Deliverables

You are to submit a brief technical report by the beginning of the next laboratory. Your report must contain:

- Cover page containing your name, the date, the laboratory number, the course name, your instructor's name
- A brief description on how your design works
- Simulation results
- Your entire project
- Demonstration of your design

This laboratory assignment has to be performed individually.