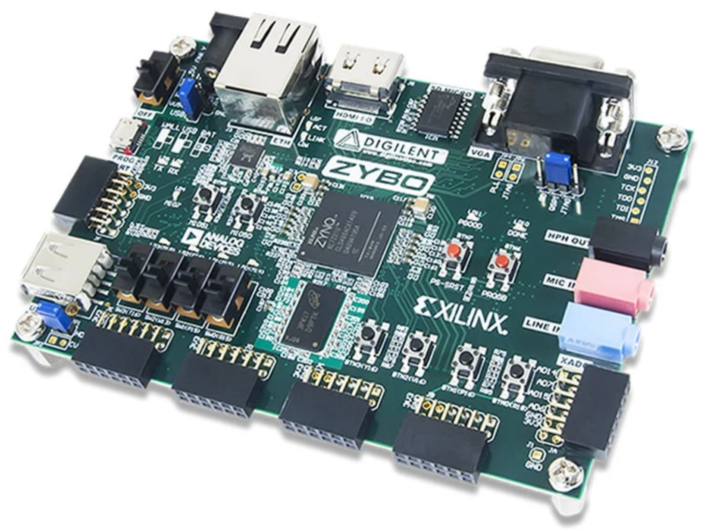
ECE 520

Final Project

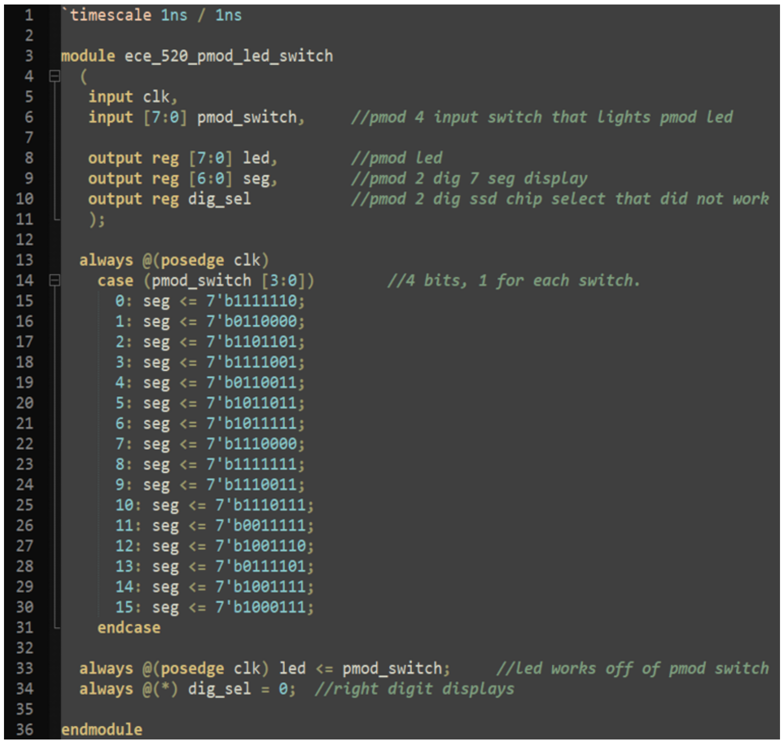
“Pmod LED and Pmod Switches”

By: David Chun 5/11/2022



Introduction

This project is a demonstration of how to use a Pmod LED (8) and a Pmod Switches (4) to show a binary counter. In order to begin this experiment, I first had to understand what Pmods are and how they work. Pmod's are a Digilent made line of products that work with microprocessors and FPGA boards. They are easy to use attachments that are relatively affordable and are well documented. The data sheets of the Pmods and the FPGA board are all that are needed to map the pins. After having a decent understanding of what I must do, I began creating my module.



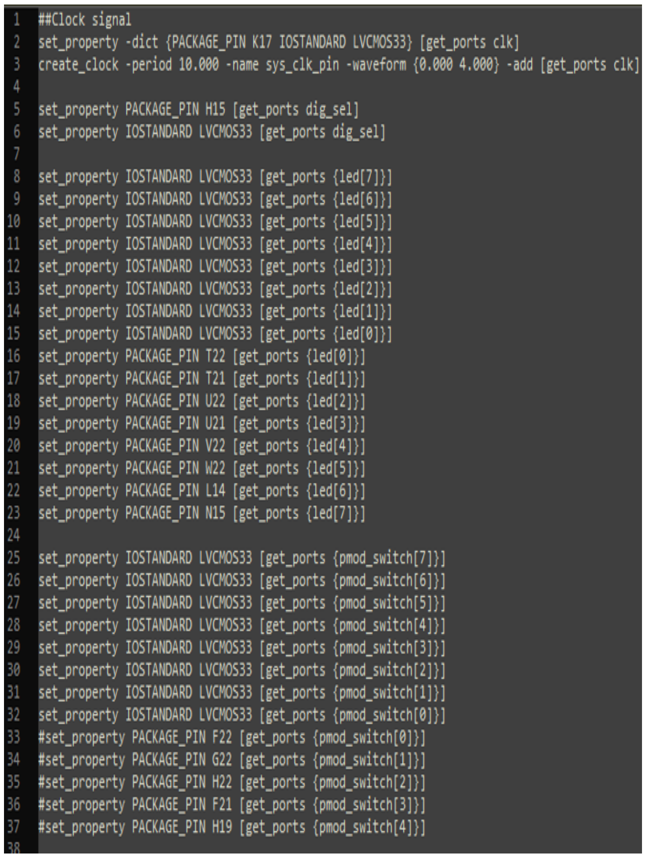
```verilog
`timescale 1ns / 1ns

module ece_520_pmod_led_switch
  (
   input clk,
   input [7:0] pmod_switch,     //pmod 4 input switch that lights pmod led

   output reg [7:0] led,        //pmod led
   output reg [6:0] seg,        //pmod 2 dig 7 seg display
   output reg dig_sel           //pmod 2 dig ssd chip select that did not work
   );

  always @(posedge clk)
    case (pmod_switch [3:0])        //4 bits, 1 for each switch.
      0: seg <= 7'b1111110;
      1: seg <= 7'b0110000;
      2: seg <= 7'b1101101;
      3: seg <= 7'b1111001;
      4: seg <= 7'b0110011;
      5: seg <= 7'b1011011;
      6: seg <= 7'b1011111;
      7: seg <= 7'b1110000;
      8: seg <= 7'b1111111;
      9: seg <= 7'b1110011;
      10: seg <= 7'b1110111;
      11: seg <= 7'b0011111;
      12: seg <= 7'b1001110;
      13: seg <= 7'b0111101;
      14: seg <= 7'b1001111;
      15: seg <= 7'b1000111;
    endcase

  always @(posedge clk) led <= pmod_switch;     //led works off of pmod switch
  always @(*) dig_sel = 0;  //right digit displays

endmodule
```

Figure 1: Design Code

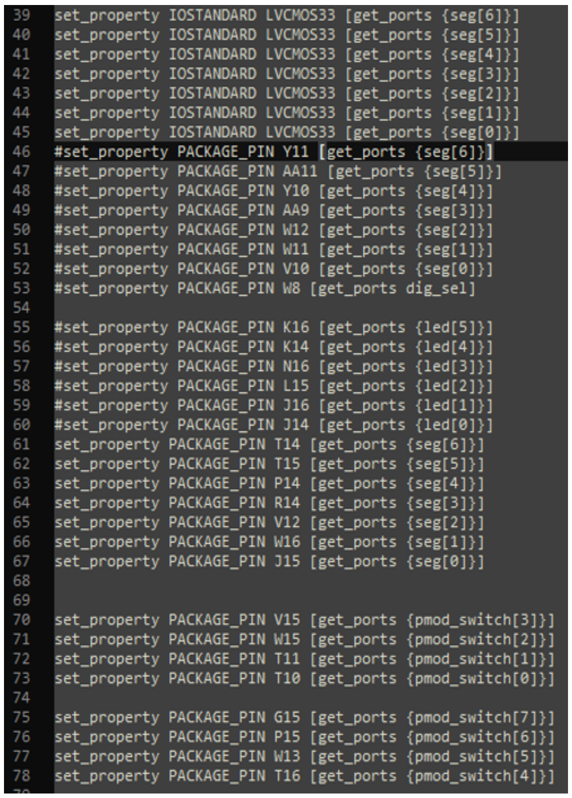
Methodology

I decided to write my code in verilog since I had created a counter beforehand. My module was filled with two inputs: one of single bit size for clock and one of 8 bit size for the switches. I also added 3 output registers: one of 8 bit size for the pmod leds, one of

7 bit size for the segments, and one of one bit size for the digit select. The digit select would determine which digit is being lit up when logic is 0 for the right digit or 1 for the left digit. I used these inputs and outputs for connecting the pmods to the fpga board. I then instantiated a case where all the numbers from 0 - 15 are displayed on the ssd Numbers 10 - 15 are displayed in hexadecimal.



```
##Clock signal
set_property -dict {PACKAGE_PIN K17 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 4.000} -add [get_ports clk]

set_property PACKAGE_PIN H15 [get_ports dig_sel]
set_property IOSTANDARD LVCMOS33 [get_ports dig_sel]

set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN T22 [get_ports {led[0]}]
set_property PACKAGE_PIN T21 [get_ports {led[1]}]
set_property PACKAGE_PIN U22 [get_ports {led[2]}]
set_property PACKAGE_PIN U21 [get_ports {led[3]}]
set_property PACKAGE_PIN V22 [get_ports {led[4]}]
set_property PACKAGE_PIN W22 [get_ports {led[5]}]
set_property PACKAGE_PIN L14 [get_ports {led[6]}]
set_property PACKAGE_PIN N15 [get_ports {led[7]}]

set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pmod_switch[0]}]
#set_property PACKAGE_PIN F22 [get_ports {pmod_switch[0]}]
#set_property PACKAGE_PIN G22 [get_ports {pmod_switch[1]}]
#set_property PACKAGE_PIN H22 [get_ports {pmod_switch[2]}]
#set_property PACKAGE_PIN F21 [get_ports {pmod_switch[3]}]
#set_property PACKAGE_PIN H19 [get_ports {pmod_switch[4]}]

```



```
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
#set_property PACKAGE_PIN Y11 [get_ports {seg[6]}]
#set_property PACKAGE_PIN AA11 [get_ports {seg[5]}]
#set_property PACKAGE_PIN Y10 [get_ports {seg[4]}]
#set_property PACKAGE_PIN AA9 [get_ports {seg[3]}]
#set_property PACKAGE_PIN W12 [get_ports {seg[2]}]
#set_property PACKAGE_PIN W11 [get_ports {seg[1]}]
#set_property PACKAGE_PIN V10 [get_ports {seg[0]}]
#set_property PACKAGE_PIN W8 [get_ports dig_sel]

#set_property PACKAGE_PIN K16 [get_ports {led[5]}]
#set_property PACKAGE_PIN K14 [get_ports {led[4]}]
#set_property PACKAGE_PIN N16 [get_ports {led[3]}]
#set_property PACKAGE_PIN L15 [get_ports {led[2]}]
#set_property PACKAGE_PIN J16 [get_ports {led[1]}]
#set_property PACKAGE_PIN J14 [get_ports {led[0]}]
set_property PACKAGE_PIN T14 [get_ports {seg[6]}]
set_property PACKAGE_PIN T15 [get_ports {seg[5]}]
set_property PACKAGE_PIN P14 [get_ports {seg[4]}]
set_property PACKAGE_PIN R14 [get_ports {seg[3]}]
set_property PACKAGE_PIN V12 [get_ports {seg[2]}]
set_property PACKAGE_PIN W16 [get_ports {seg[1]}]
set_property PACKAGE_PIN J15 [get_ports {seg[0]}]


set_property PACKAGE_PIN V15 [get_ports {pmod_switch[3]}]
set_property PACKAGE_PIN W15 [get_ports {pmod_switch[2]}]
set_property PACKAGE_PIN T11 [get_ports {pmod_switch[1]}]
set_property PACKAGE_PIN T10 [get_ports {pmod_switch[0]}]

set_property PACKAGE_PIN G15 [get_ports {pmod_switch[7]}]
set_property PACKAGE_PIN P15 [get_ports {pmod_switch[6]}]
set_property PACKAGE_PIN W13 [get_ports {pmod_switch[5]}]
set_property PACKAGE_PIN T16 [get_ports {pmod_switch[4]}]
```

Figure 2: Constraints/Mapping

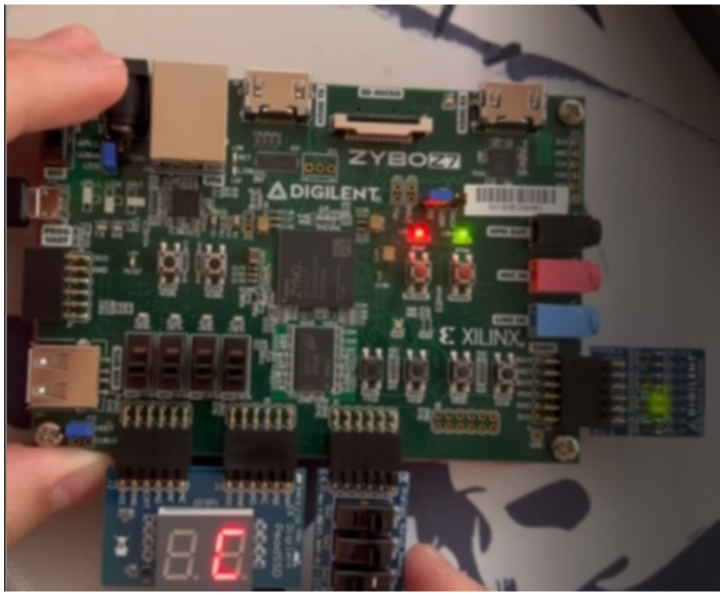


Figure 3: Physical Model

Analysis

After creating my module, I was able to test it by first generating synthesis. I ran the synthesis to make sure that there are no errors and so that I can map my inputs and outputs. After assigning the inputs as stated in the data sheets of my Pmods and my Zybo Z7 10 board, I ran the implementation. Implementation will only pass if all ports are mapped and there are no overlaps. If both synthesis and implementation pass and are updated to the newest module, then I can run bitstream generation. Bitstream generation will allow me to open the hardware manager where I can connect my FPGA board as well as program my board with the bit file generated. I was successfully able to connect my board and when I ran the program, the result was exactly as I had expected. The switches acted as binary digits with up being a ‘1’ and down being a ‘0.’ It counted all the way to 15 and I mapped the onboard switches to increase the amount of leds for future implementations.

Conclusion

Overall, this project was pretty fun to do. Although it was unfortunate that I was unable to get my 00 to 99 counter to function originally, I was able to practice i/o mapping and get a successful binary counter using external leds and switches. Future iterations to this project can be made by fixing my binary counter, and adding more switches/leds for higher binary bits.

https://github.com/chundew/ECE-520-Final-Project