**EENG2131 – Digital Systems Lab#5 – 7 Segment Displays**

Antonio Veguilla Hernandez Student ID: P000166790

**Part 0 -Background**

A 7 – segment display has one pin for each of the segments, and one pin that is common. That common pin is known as the “common-anode”, meaning that the anodes of all the LED segments are connected. When a digit’s common-anode pin is connected high and an individual segment’s pin is connected to ground through a resistor, that segment lights up.

A picture containing text, receipt, screenshot

Description automatically generated

Diagram

Description automatically generated

Reference provided for this lab.: <https://digilent.com/reference/programmable-logic/basys-3/reference-manual#seven_segment_display>

**Part 1 – BCD to 7 -Segment Decoder**

In this part of the lab, we had to design a combinational logic Verilog module to convert a 4-bit BCD value in the range of 0-9 into the seven logic signals that will drive the 7 cathodes of the segments.

A truth table was also required to help us with our implementation

|  |  |
| --- | --- |
| BCD | segN  (Active low bits!)  7’bGFEDCBA (G thru A) |
| 0 | 7’b1000000 |
| 1 | 7’b1111001 |
| 2 | 7’b0100110 |
| 3 | 7’b0110000 |
| 4 | 7’b0011001 |
| 5 | 7’b0010010 |
| 6 | 7’b0000010 |
| 7 | 7’b1111000 |
| 8 | 7’b0000000 |
| 9 | 7’b0011000 |
| A | 7’b0001000 |
| B | 7’b0000011 |
| C | 7’b1000110 |
| D | 7’b0100001 |
| E | 7’b0000110 |
| F | 7’b0001110 |

6 : begin

segN = 7'b0000010; //6

end

7 : begin

segN = 7'b1111000; //7

end

8 : begin

segN = 7'b0000000; //8

end

9 : begin

segN = 7'b0010000; //9

end

10 : begin

segN = 7'b0001000; //A

end

11 : begin

segN = 7'b0000011; //b

end

12 : begin

segN = 7'b1000110; //C

end

13 : begin

segN = 7'b0100001; //d

end

14 : begin

segN = 7'b0000110; //E

end

default : begin

segN = 7'b0001110; //F

end

endcase

end

endmodule

**Part 1 -BCD to 7 – Segment Decoder**

The next part task was to use the following module to help us with our implementation:

module BCDtoSevenSegN(  
 input [3:0] bcd;  
 output [6:0] segN; // Active low!  
 always @ ( ??? )  
 begin  
 // TODO – Create implementation here  
 end )  
endmodule

**My implemented module:**

module BCDtoSevenSegN(

input [3:0]bcd,

output [6:0]segN); //Active low!

reg [6:0]segN;

always @ bcd

begin

case(bcd)

0 : begin

segN = 7'b1000000; //0

end

1 : begin

segN = 7'b1111001; //1

end

2: begin

segN = 7'b0100100; //2

end

3 : begin

segN = 7'b0110000; //3

end

4 : begin

segN = 7'b0011001; //4

end

5 : begin

segN = 7'b0010010; //5

end

else if(count1 == 1)

begin

AN <= 4'b1101;

count1 <= count1 +1; //count 1 is set to 2

count0 <= 0;

end

else if(count1 == 2)

begin

AN <= 4'b1011;

count1 <= count1 +1; //count 1 is set to 3

count0 <= 0;

end

else //if(count1 == 3)

begin

AN <= 4'b0111;

count1 <= 0; //resets the counter count1

count0 <= 0;

end

end

else

begin

count0 <= count0+1;

end

end

endmodule

**Part 2 -Digit timing controller**

On this section, we need to rapidly display each digit in sucession, which means, create a Verilog module to control the four timing signals that enable each of the 7-seghment display digits.

The following module was provide to help us with our implementation:

module DigitController(clk, AN);  
 parameter numDigits = 4;  
 parameter refreshRate\_ms = 10;  
 input clk; // 100 MHz  
 output [numDigits-1:0] AN; // Active low!  
 )  
 // TODO – Create implementation here  
endmodule

**My implemented module:**

module DigitController(

input clk,

output [3:0]AN

);

parameter refresh\_ms = 0.010; //10ms

parameter numDigits = 4;

reg [20:0] count0 = 0; //counter with a magnitude of 2^19

reg [1:0]count1 = 0; //this will be used to increment 1 everytime count 0 is reset

reg [numDigits - 1:0]AN = 4'b0111;

always @ (posedge clk)

begin

if(count0 == 125000)//refresh\_ms/numDigits)\*50000000

begin

if(count1 == 0)

begin

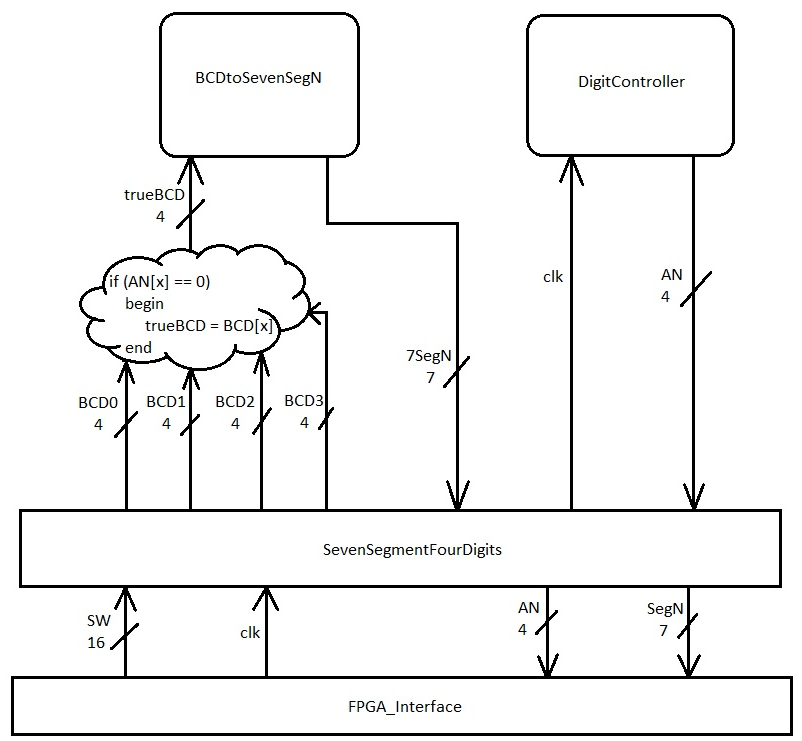
AN <= 4'b1110;

count1 <= count1 +1; //count 1 is set to 1

count0 <= 0;

end

Here’s the block diagram developed as part of this section:



My two implemented modules:

module FPGA\_Interface(sw,an,seg,clk,led);

output [3:0]an;

input clk;

input [15:0]sw;

output [6:0]seg;

output [15:0]led;

SevenSegmentFourDigits(clk, sw[3:0], sw[7:4], sw[11:8], sw[15:12], an, seg);

assign led[15:0] = sw[15:0];

endmodule

**Part 3 -Put it all together**

On this section, we had to create a new top -level module that will let us easily drive the four digit 7-segment display using four separate BCD values. The BCD values will come from the slides switches on the Basys3 FPGA device board, and those numbers will be translated into the correct signals ro drive the four 7-segmane displays.

The following module was preovide to helps with our implementation:

module SevenSegmentFourDigits(clk, bcd0, bcd1, bcd2, bcd3, AN, segN);  
 input clk; // Standard 100 MHz clock  
 input [3:0] bcd0; // right-most digit  
 input [3:0] bcd1;  
 input [3:0] bcd2;  
 input [3:0] bcd3; // left-most digit  
 output [3:0] AN; // digit enable, active low  
 output [7:0] segN; // segment enable, active low  
 // TODO – Create implementation here  
endmodule

For FPGA synthesis, I used the following connections for the inputs and outputs of SevenSegmentFourDigits module:

clk: Normal 100 MHz clock  
BCD input value 0: Switches 3:0  
BCD input value 1: Switches 7:4  
BCD input value 2: Switches 11:8  
BCD input value 3: Switches 15:12  
Digit enable AN: 7-segment digit pins (see constraints file)  
Digit enable AN: 7-segment segment pins (see constraints file)

Actually, 2 new modules were required, the first module was “SevenSegmentFourDigits” and the 2nd (my true top level module) “FPGA\_Interface”.

The module called “SevenSegmentFourDigits”is the module that provides the clk signal to my “DigitController” module, also the “DigitController” provide the input control of the 4 anodes “AN[3:0]”. Also, the “SevenSegmentFourDigits” receives the the 7-segment inputs from the module called “BCDtoSevenSegN”, while the “SevenSegmentFourDigits” module sends the inputs for the BCDs to the module “BCDtoSevenSegN”.

**Part 3 -Put it all together**

module SevenSegmentFourDigits(clk, bcd0, bcd1, bcd2, bcd3, AN, segN);

input clk; //Standard 100MHz clock

input [3:0]bcd0; //right most digit

input [3:0]bcd1;

input [3:0]bcd2;

input [3:0]bcd3; //left-most digit

output [3:0]AN; //digit enable active low

output [7:0]segN; //segment enable, active low

BCDtoSevenSegN dut(trueBCD, segN);

DigitController dut1(clk, AN);

reg [3:0]trueBCD = 0;

always@(AN)

begin

if(AN[0] == 0)

begin

trueBCD = bcd0;

end

else if(AN[1] == 0)

begin

trueBCD = bcd1;

end

else if(AN[2] == 0)

begin

trueBCD = bcd2;

end

else if(AN[3] == 0)

begin

trueBCD = bcd3;

end

end

endmodule

Conclusion: This laboratory was very challenging and satisfying, the professor support and guidance was a key factor for me to finish this laboratory. I can say that the Verilog platform is a powerful tool for development and for behavioral troubleshooting, however it has so many details to consider and decipher what the error log really mean is an art. Overall, it takes a lot of effort to truly understand how to make things work within Verilog, but once you get a good grasp on the software, watching the simulations and see your logic working within the FPGA board, it’s worth it!.