3TB4 Lab 1 Report
02/06/2021

Group 22

Brian Chiu:
MacID: chiuh1
Student #: 400054774

Christian Mavely:
MacID: mavelyc
Student #: 400057598

3TB4 Lab 1 Report

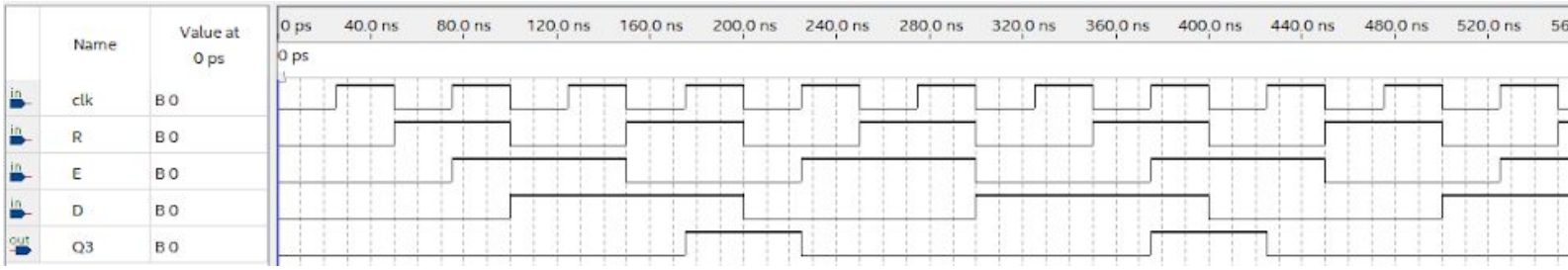
In this lab we studied the behavior of various modules by running simulations and we implemented the custom 7-segment display decoder module we created in the prelab. We mapped the inputs and outputs to physically operate on our FPGAs. We then combined our decoder with a multiplexer and a few different sources of inputs which allowed the display to change its output depending on what the multiplexer gave it.

Part 1

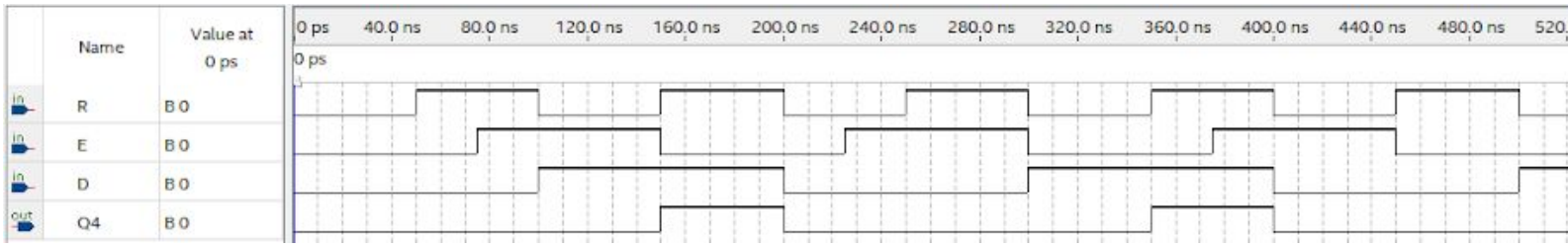
In part 1 we ran a simulation on the following modules.

- D-flipflop with low reset and low enable
- D-latch
- 4 bit counter with reset and enable
- 4 bit multiplexer

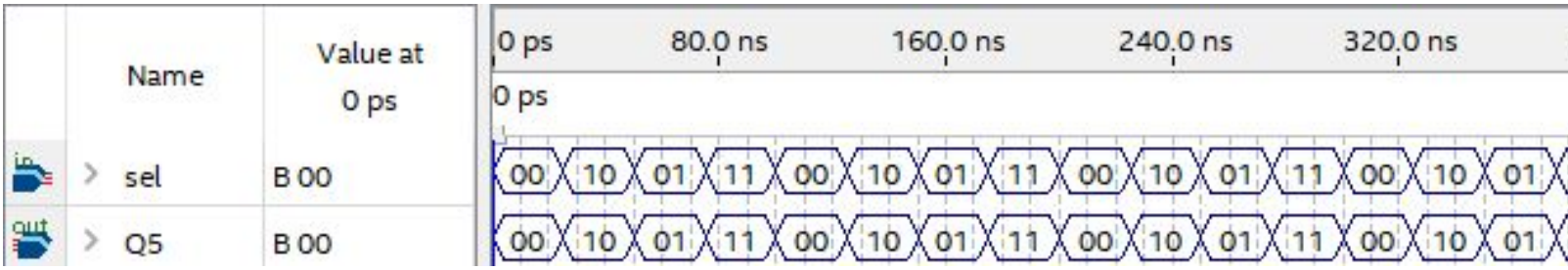
D-flipflop::



D-latch:

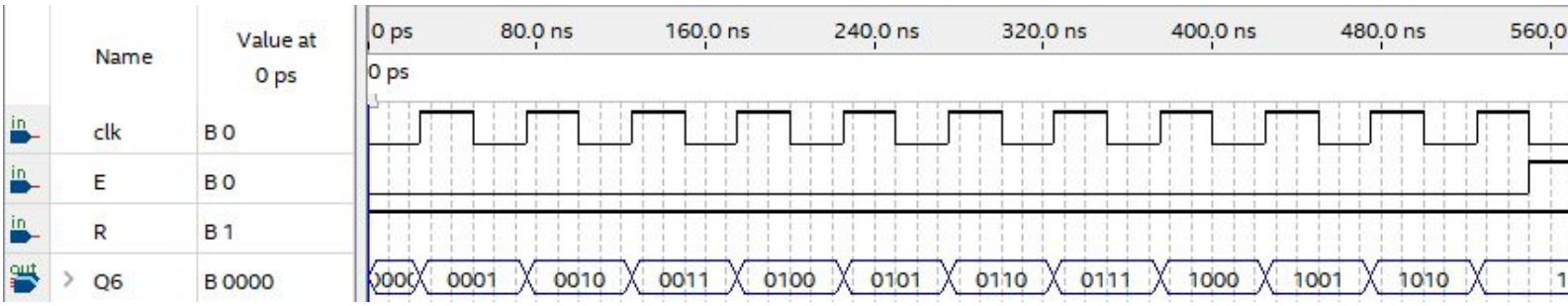


Multiplexer with hard coded values 00-01-10-11:



3TB4 Lab 1 Report

4 bit Counter



Part 2:

In part 2 we created a module to map the inputs and outputs of our seven segment decoder to the switches and seven segment display on our FPGAs. The input was 4 bits from SW[0] to SW[3]. The output was to HEX0. This module was demonstrated to our TA's during our lab time.

Part 3:

In part 3 we combined our decoder with a few counters and a multiplexer to make the project behave in the following manner:

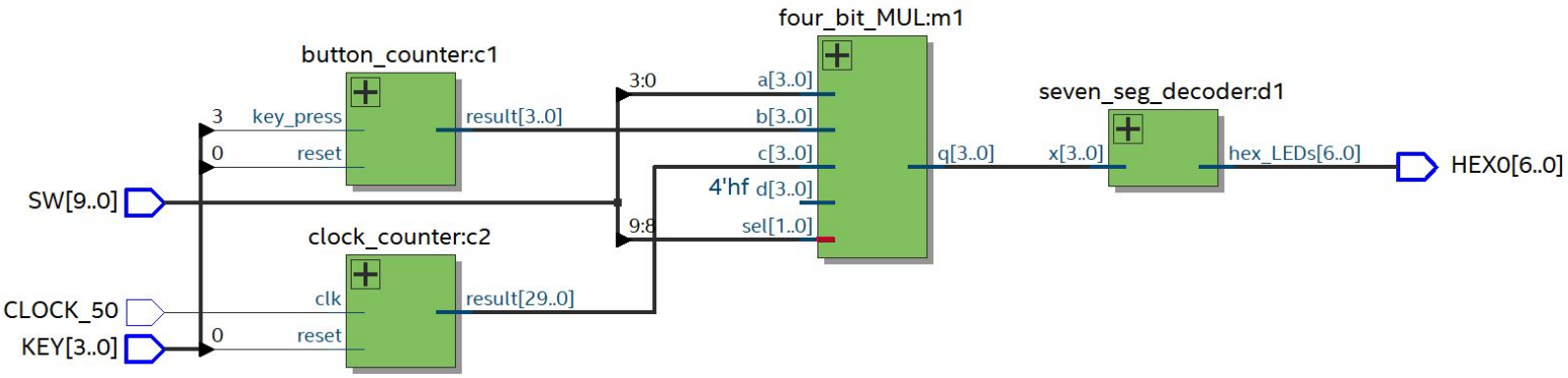
- The decoder outputs to display HEX0
- The multiplexer chose the input to the decoder
- Switches SW[9] and SW[8] is the 2 bit sel input for the multiplexer

S[9], S[8]	Input to decoder
00	4 bit Binary input from switches SW[0] to SW[3]
01	4 bit binary counter incremented by key[3]
10	4 most significant bits of 30 bit counter increment by CLOCK50
11	OFF

In addition to this behavior KEY[0] would reset both the key 3 counter and the clock counter.

3TB4 Lab 1 Report

RTL Viewer



Pin Planner:

Named: * Edit:

Node Name	Direction	Location	I/O Bank	/REF Group	tter Locatic	'O Standar	Reserved	rent Stren	Slew Rate	ffe
CLOCK_50	Input	PIN_AF14	3B	B3B_N0	PIN_AF14	2.5 V		12mA...ult)		
HEX0[6]	Output	PIN_AH28	5A	B5A_N0	PIN_AH28	2.5 V		12mA...ult)	1 (default)	
HEX0[5]	Output	PIN_AG28	5A	B5A_N0	PIN_AG28	2.5 V		12mA...ult)	1 (default)	
HEX0[4]	Output	PIN_AF28	5A	B5A_N0	PIN_AF28	2.5 V		12mA...ult)	1 (default)	
HEX0[3]	Output	PIN_AG27	5A	B5A_N0	PIN_AG27	2.5 V		12mA...ult)	1 (default)	
HEX0[2]	Output	PIN_AE28	5A	B5A_N0	PIN_AE28	2.5 V		12mA...ult)	1 (default)	
HEX0[1]	Output	PIN_AE27	5A	B5A_N0	PIN_AE27	2.5 V		12mA...ult)	1 (default)	
HEX0[0]	Output	PIN_AE26	5A	B5A_N0	PIN_AE26	2.5 V		12mA...ult)	1 (default)	
KEY[3]	Input	PIN_Y16	3B	B3B_N0	PIN_Y16	2.5 V		12mA...ult)		
KEY[2]	Input	PIN_W15	3B	B3B_N0	PIN_W15	2.5 V		12mA...ult)		
KEY[1]	Input	PIN_AA15	3B	B3B_N0	PIN_AA15	2.5 V		12mA...ult)		
KEY[0]	Input	PIN_AA14	3B	B3B_N0	PIN_AA14	2.5 V		12mA...ult)		
SW[9]	Input	PIN_AE12	3A	B3A_N0	PIN_AE12	2.5 V		12mA...ult)		
SW[8]	Input	PIN_AD10	3A	B3A_N0	PIN_AD10	2.5 V		12mA...ult)		
SW[7]	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V		12mA...ult)		
SW[6]	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V		12mA...ult)		
SW[5]	Input	PIN_AD12	3A	B3A_N0	PIN_AD12	2.5 V		12mA...ult)		
SW[4]	Input	PIN_AD11	3A	B3A_N0	PIN_AD11	2.5 V		12mA...ult)		
SW[3]	Input	PIN_AF10	3A	B3A_N0	PIN_AF10	2.5 V		12mA...ult)		
SW[2]	Input	PIN_AF9	3A	B3A_N0	PIN_AF9	2.5 V		12mA...ult)		

All Pins

Questions

1. The role of the DE1-SoC.qsf file is to map switch, button, key, etc. assignments to those that are present on the FPGA. These assignments can now be referenced within the .v file to use as input/output from the FPGA and compute or act out any programmable logic.
2. Compilation report numbers:
    a. Total number of logic elements: 24
    b. Total number of registers: 38
    c. Total number of pins: 22
    d. The maximum number of logic elements that can fit on the FPGA: 32070

Code used:

Flip-flops, latch, multiplexer and counter are already in prelab

Seven seg decoder:

```
module seven_seg_decoder(input [3:0] x, output [6:0] hex_LEDs);
     reg [6:0] reg_LEDs;

     assign hex_LEDs[0]= (x[3]&x[1]) | (x[2]&!x[1]&!x[0]) |
(!x[3]&!x[2]&!x[1]&x[0]);
     assign hex_LEDs[1]= (x[3]&x[1]) | (x[3]&x[2]&!x[0]) |
(x[2]&x[1]&!x[0]) | (!x[3]&x[2]&!x[1]&x[0]);

     assign hex_LEDs[6:2]=reg_LEDs[6:2];

     always @(*)
     begin
          case (x)
               4'b0000: reg_LEDs[6:2]=5'b10000; //7'b1000000
decimal 0
               4'b0001: reg_LEDs[6:2]=5'b11110; //7'b1111001
decimal 1
               4'b0010: reg_LEDs[6:2]=5'b01001; //7'b0100100
decimal 2
               4'b0011: reg_LEDs[6:2]=5'b01100; //7'b0110000
decimal 3
               4'b0100: reg_LEDs[6:2]=5'b00110; //7'b0011001
decimal 4
```

```
                4'b0101: reg_LEDs[6:2]=5'b00100; //7'b0010010
decimal 5
                4'b0110: reg_LEDs[6:2]=5'b00000; //7'b0000010
decimal 6
                4'b0111: reg_LEDs[6:2]=5'b11110; //7'b1111000
decimal 7
                4'b1000: reg_LEDs[6:2]=5'b00000; //7'b0000000
decimal 8
                4'b1001: reg_LEDs[6:2]=5'b00100; //7'b0010000
decimal 9
                4'b1010: reg_LEDs[6:2]=5'b00000; //7'b0000011
char    b
                4'b1011: reg_LEDs[6:2]=5'b01011; //7'b0101111
char      r
                4'b1100: reg_LEDs[6:2]=5'b10011; //7'b1001111
char    I
                4'b1101: reg_LEDs[6:2]=5'b00010; //7'b0001000
char    A
                4'b1110: reg_LEDs[6:2]=5'b01010; //7'b0101011
char    n
                4'b1111: reg_LEDs[6:2]=5'b11111; //7'b1111111
          OFF
          endcase
     end
endmodule
```

lab1part2:

```
module lab1part2 (input [3:0] SW, output [6:0] HEX0);

     wire [6:0]hex_LEDs;
     seven_seg_decoder decode(.x(SW), .hex_LEDs(hex_LEDs));

     assign HEX0 = hex_LEDs;

endmodule
```

Button activated counter:

```
module button_counter(input key_press, input reset, output
reg[3:0] result);

     always@(negedge key_press, negedge reset) begin
     //buttons activate on negedge

          if(!reset)begin
               result <= 4'b0;

          end else begin
               result <= result + 1'b1;
          end

     end
endmodule
```

Clock activated counter:

```
module clock_counter(input clk, reset,  output reg[WIDTH-1:0]
result);

     parameter WIDTH=30;
     //2's power of 26 is 67,108,864.
     // that is 1 second (22^6/50e6 = 1.34), if count is using
CLOCK_50

     always@(posedge clk, negedge reset)

     if(!reset)begin
          result<=30'b0;

     end else begin
          result<=result+1'b1;

     end
endmodule
```

4 bit Multiplexer:

```
module four_bit_MUL (sel, a, b, c, d, q);
     // ==============in out dec=============//
     input [1:0]sel;
     input [3:0]a;       // 4 bits for inputs and output
     input [3:0]b;
     input [3:0]c;
     input [3:0]d;

     output reg [3:0]q;

     always @ (sel) begin
      case (sel)
         2'b00 : q <= a;
         2'b01 : q <= b;
         2'b10 : q <= c;
         2'b11 : q <= d;
      endcase
   end
endmodule
```

Lab1part3:

```
module lab1part3 (input [9:0] SW,
                              input [3:0] KEY,
                              input CLOCK_50,
                              output [6:0] HEX0);

     wire [3:0]off = 4'b1111;                //hard code off
     wire [3:0]key_counter_val;                   //output of
button counter
     wire [29:0]clock_counter_val;           //output of clock
counter

     wire [3:0]display_in;                        //multiplexer
output
     wire [6:0]display_out;                       //display


     assign HEX0 = display_out;
```

```
     button_counter c1(  .key_press(KEY[3]),
                                    .reset(KEY[0]),

.result(key_counter_val));

     clock_counter c2(        .clk(CLOCK_50),
                                    .reset(KEY[0]),

.result(clock_counter_val));

     four_bit_MUL m1(         .sel(SW[9:8]),
                                    .a(SW[3:0]),
                                    .b(key_counter_val),

.c(clock_counter_val[29:26]),          //4 most significant
bits
                                    .d(off),
                                    .q(display_in));

     seven_seg_decoder d1(.x(display_in),
                                    .hex_LEDs(display_out));


endmodule
```