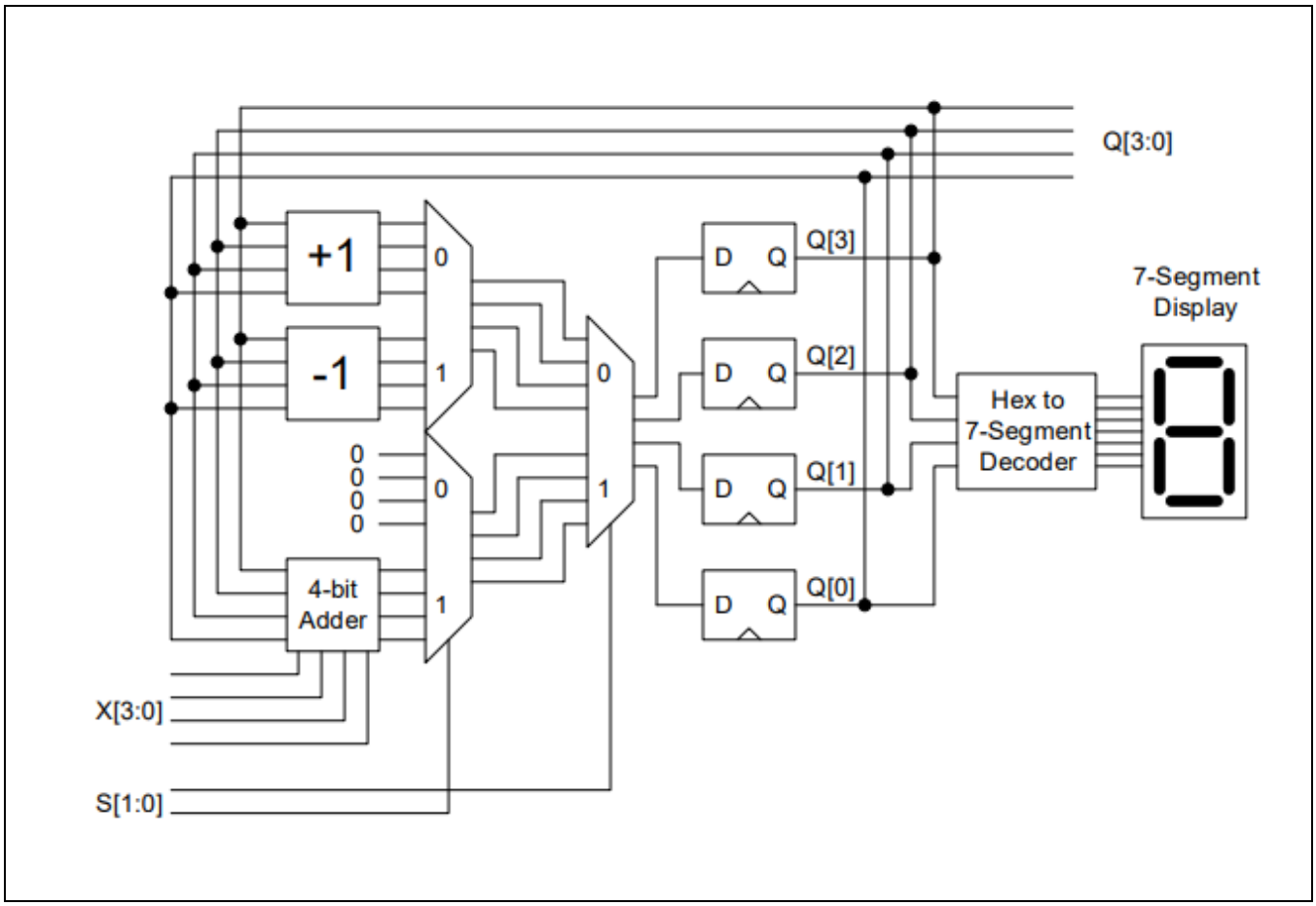
Joshua Pollock

EE 310 – Lab 2 Report

NAU, 9 February 2020

Problem Description

In this lab, we have been asked to design a 4-bit multi-function register circuit. This circuit will be capable of clearing, incrementing, decrementing and adding a value onto the current value of the register. The S input will provide a way to select which operation will take place. When S is 0, it will add 1 to the Q register. When S is 1, it will subtract 1 from the Q register. When S is 2, it will reset the Q register to 0. Finally, when S is 3, it will add the register X to the Q register. The circuit will wait for the clock signal to update the Q register.



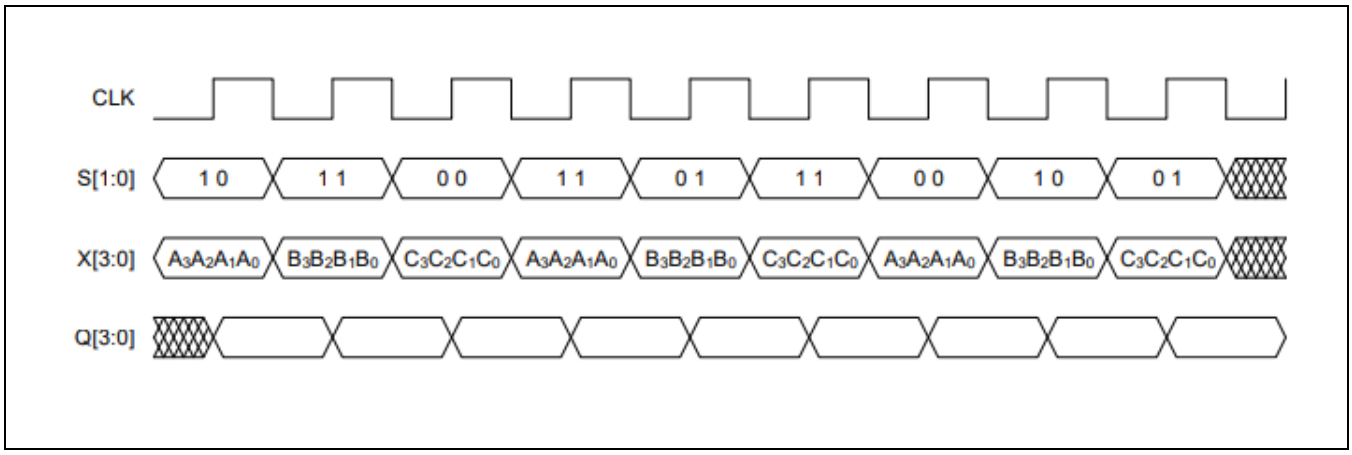


Figure 1. Expected behavior of the circuit

Solution Plan

In order to solve the problem explained above, I created two blocks of code for different tasks. The first block would simply wait for the positive edge of the clock to begin. This block would set Q to muxResult, which is calculated from the other block. It would then also use muxResult and a 7-segment display decoder case statement to output the correct result into SS. The other block of code was a single case statement that would check what S was and complete operations based on that. If S was 0, it would add 1 to the Q register. Else if S was 1, it would subtract 1 from the Q register. Else if S was 2, it would reset the Q register to 0. Finally, else S had to be 3, then it would add the register X to the Q register. The reason these two blocks were separated was because of the given expected behavior of the circuit. Q would only ever change on the positive edge on the clock, while S and X could change at any time.

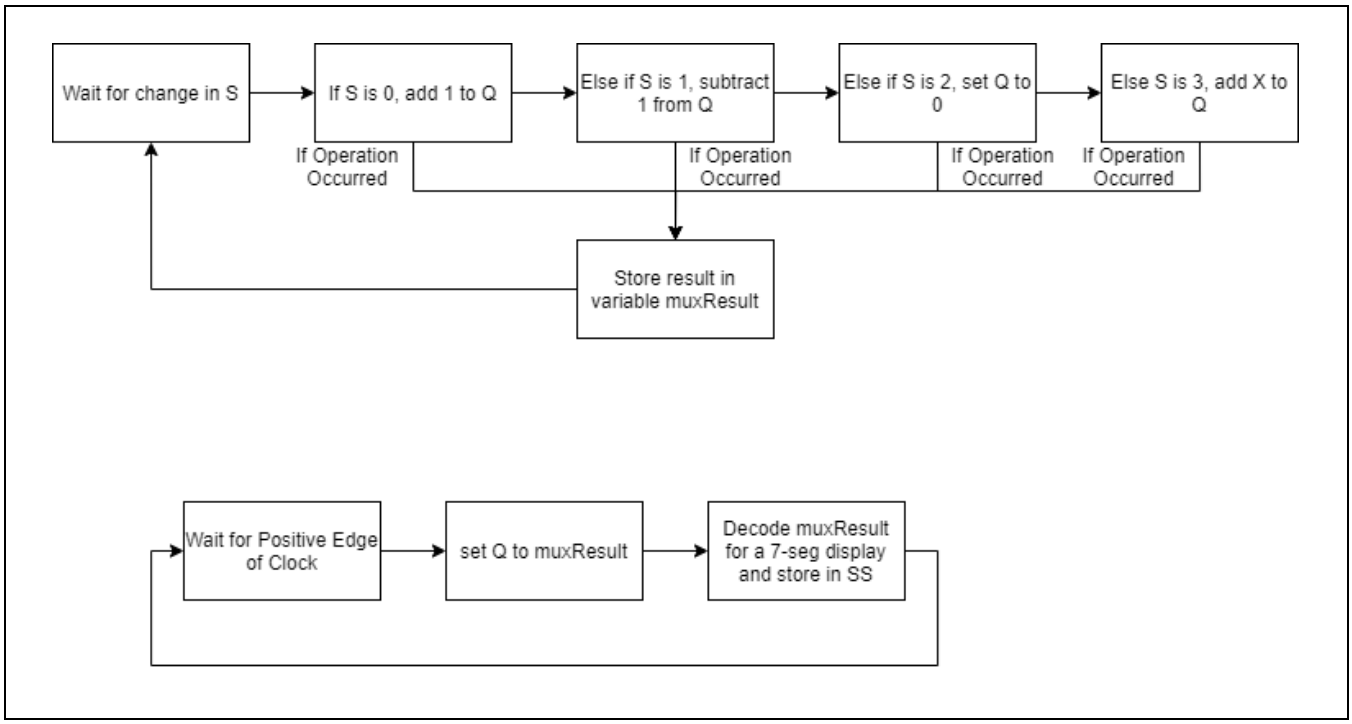


Figure 2. State diagram for the proposed solution

Implementation and Test Plan

I have implemented the solution plan explained above, by first completing the prelim of the lab, creating the main program and a testbench to simulate the program. The testbench simulated a clock and then using the S and X variables I was able to test all the different operations possible on the output of Q and SS. To complete the prelim of the lab, I needed to change my student ID into binary which resulted in "10 0101 1110 0110" where A= 0101, B= 1110, and C=0110. Using these and the given S values from Figure 1, I was able to simulate and complete the preliminary lab. From here, the lab was as simple as plugging in the FPGA board and simulating it using the baseline file. For the baseline, clk was key 0, S was switches 4 and 5, X was switches 3 through 0, Q was the red LEDs 3 through 0, and finally SS was displayed on HEX0.

One error I had in my testbench was setting Q_tb to 0. I thought this would help with ensuring the proper inputs and outputs but caused weird behaviors instead. After fixing this small bug, the code worked flawlessly.

Lab3.v

```
module Lab3 ( clk , S , X , Q , SS ) ;

        input clk;
        input [1:0] S;
        input [3:0] X;
        output reg [3:0] Q;
```

```
        output reg [6:0] SS;

        reg [3:0] muxResult;


        always  @ (S, X, Q)  begin
                case (S)
                        2'b00: muxResult = Q + 4'b0001;
                        2'b01: muxResult = Q - 4'b0001;
                        2'b10: muxResult = 0;
                        2'b11: muxResult = X + Q;
                endcase
        end

        always  @ (posedge clk)  begin

                Q = muxResult;


                case  ( muxResult )
                                4'h0  :  SS =  7'b1000000 ;
                                4'h1  :  SS =  7'b1111001 ;
                                4'h2  :  SS =  7'b0100100 ;
                                4'h3  :  SS =  7'b0110000 ;
                                4'h4  :  SS =  7'b0011001 ;
                                4'h5  :  SS =  7'b0010010 ;
                                4'h6  :  SS =  7'b0000010 ;
                                4'h7  :  SS =  7'b1111000 ;
                                4'h8  :  SS =  7'b0000000 ;
                                4'h9  :  SS =  7'b0010000 ;
                                4'hA  :  SS =  7'b0001000 ;
                                4'hB  :  SS =  7'b0000011 ;
                                4'hC  :  SS =  7'b1000110 ;
                                4'hD  :  SS =  7'b0100001 ;
                                4'hE  :  SS =  7'b0000110 ;
                                4'hF  :  SS =  7'b0001110 ;
                endcase
        end
endmodule
```

Lab3_tb.v

```
module Lab3_tb;

parameter  PER  =  10 ;
```

```
reg  clk_tb ;
reg [1:0] S_tb;
reg [3:0] X_tb;
wire [3:0] Q_tb;
wire [6:0] SS_tb;

// 5209702 -> 9702 -> 10 0101 1110 0110 -> A= 0101 b= 1110 c=0110
reg [3:0] A = 4'b0101;
reg [3:0] B = 4'b1110;
reg [3:0] C = 4'b0110;

Lab3  dut  ( clk_tb , S_tb , X_tb , Q_tb , SS_tb ) ;

always  begin
  clk_tb  =  0 ;
  #(PER/2) ;
  clk_tb  =  1 ;
  #(PER/2) ;
end

initial begin

S_tb = 2'b10;
X_tb = A;
#PER;
S_tb = 2'b11;
X_tb = B;
#PER;
S_tb = 2'b00;
X_tb = C;
#PER;
S_tb = 2'b11;
X_tb = A;
#PER;
S_tb = 2'b01;
X_tb = B;
#PER;
S_tb = 2'b11;
X_tb = C;
#PER;
S_tb = 2'b00;
X_tb = A;
#PER;
S_tb = 2'b10;
X_tb = B;
#PER;
```

```
S_tb = 2'b01;
X_tb = C;
#PER;
$stop;

end
endmodule
```

baseline_c5gx.v (Only one line added)

```
Lab3 dut ( KEY[0], SW[5:4] , SW[3:0] , LEDR[3:0] , HEX0 ) ;
```

Figure 3. Verilog code for the proposed solution

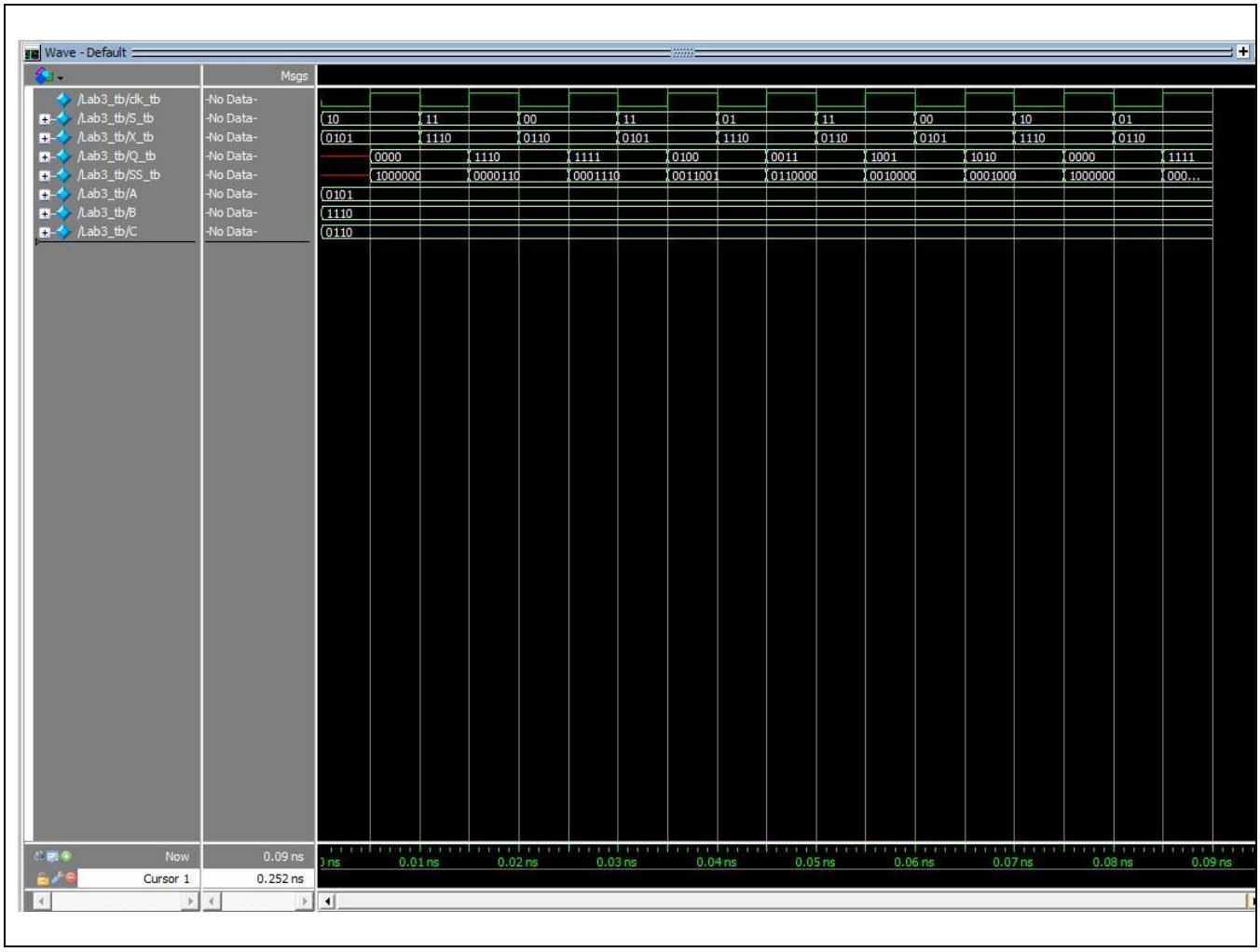


Figure 4. Lab pictures of the running solution