Project Report:

CECS 301
Louis Monfiero
Project 3 - Switch Debounce
March 28, 2019

- When a switch is pushed, the “output” bounces for a period of time. By bouncing, it’s outputting either a 1 or a 0 repeatedly until it stabilizes and outputs just the 1
- The project led to the creation of a circuitry that will detect when the switch has been released and create a single one-clock wide signal to be used
- The switch tends to settle (stabilize) after 10 to 20 milliseconds
- We will assume that each bounce lasts about >1 microsecond
- We will be creating a simple implementation of a debounce circuit in this project
- Wait until the output is dependable; in other words, when it has finished bouncing and has stabilized. Sort of a pseudo-delay
- We will shift samples of our output into a shift register until all the bits have the same state of 0/1
- We will detect when the switch changes from a 0 to a 1. This applies to the momentary switches on our board
- After the switch has been stable at logic 1 for a set amount of time, we will change the output to switch to 1 for a single clock period
- Use a 1ms pulse generator so we know how long to count
- Utilize a counter and the seven segment decoder from Project 2
- The 32-bit counter increments for one clock cycle

Pulse.v

```
21  module pulse(clk, rst, pls);
22     input clk, rst;
23     output pls;
24     reg [16:0] counter;
25
26     assign pls = (counter == 17'd99_999);
27
28     always @(posedge clk, posedge rst)
29        if (rst) counter <= 17'b0; else
30        if (pls) counter <= 17'b0;
31        else counter <= counter + 17'b1;
32
33
34  endmodule
35
```

ShiftTenBit.v

```
module ShiftTenBit(clk, rst, sdi, out);
   input clk, rst, sdi;
   output [9:0] out;
   reg [9:0] out, nout;

   pulse PULSE (.clk(clk),
                .rst(rst),
                .pls(pls));

   always @ (posedge clk, posedge rst)
      if (rst) out <= 10'b0;
      else if (pls) out <= nout;

   always @ (*)
      nout = {sdi, out [9:1]};

endmodule
```

PED.v

```
module PED(clk, rst, inc, sd);
   input clk, rst, sd;
   output inc;
   reg nq;
   reg [1:0] next;
   wire [9:0] out;

   ShiftTenBit STB (.clk(clk),
                    .rst(rst),
                    .out(out),
                    .sdi(sd));

   assign inc = next[0] & !next[1];

   always @ (posedge clk, posedge rst)
      if (rst) next <= 2'b0;
      else next <= {next[0], nq};

   always @ (*)
      if (out == 10'h3FF) nq = 1;
      else nq = 0;


endmodule
```

Button.v

```
module Button(clk, rst, btn, out);
   input clk, rst, btn;
   output out;

   PED PED1 (.clk(clk),
             .rst(rst),
             .inc(test),
             .sd(btn));

   assign out = test;

endmodule
```

Count32.v

```
module count32(clk, rst, cnt, q);
   input clk, rst, cnt;
   output [31:0] q;
   reg [31:0] q;
   wire inc;
   wire [9:0] out;

   Button BT (.clk(clk),
              .rst(rst),
              .btn(cnt),
              .out(inc));

   always @(posedge clk, posedge rst)
      if (rst) q <= 32'b0; else
      if (inc) q <= q + 32'b1;

endmodule
```

Rotation.v

```
//////////////////////////////////////////
module Rotation(clk, rst, rotate);
   input clk, rst;
   output rotate;
   reg [17:0] count;

   assign rotate = (count == 18'd199_999);

   always @(posedge clk, posedge rst)
      if (rst) count <= 18'b0; else
      if (rotate) count <= 18'b0;
      else count <= count + 18'b1;

endmodule
```

Cathode.v

```
module Cathode(clk, rst, cath);
   input clk, rst;
   output reg [2:0] cath;

   Rotation RT1(.clk(clk), .rst(rst), .rotate(rotate));

   always @ (posedge clk, posedge rst)
      if (rst) cath <= 3'b0; else
      if (rotate) cath <= cath + 3'b1;
      else cath <= cath;

endmodule
```

Multiplexer.v

```
module Multiplexer(clk, rst, out, d2);
   input clk, rst, d2;
   output reg [3:0] out;

   wire [2:0] cath;
   wire [31:0] Ci;

   Cathode CA (.clk(clk),
               .rst(rst),
               .cath(cath));
   count32 CO (.clk(clk),
               .rst(rst),
               .cnt(d2),
               .q(Ci));

   always @ (*)
       begin
           case (cath)
               3'b000  :   out = Ci[3:0];
               3'b001  :   out = Ci[7:4];
               3'b010  :   out = Ci[11:8];
               3'b011  :   out = Ci[15:12];
               3'b100  :   out = Ci[19:16];
               3'b101  :   out = Ci[23:20];
               3'b110  :   out = Ci[27:24];
               3'b111  :   out = Ci[31:28];
               default :   out = 4'h0;
           endcase
       end


endmodule
```

Hex_7.v

```
module Hex_7(clk, rst, Ca, d3);
   input clk, rst, d3;
   output reg [7:0] Ca;
   wire [3:0] HexVal;

   Multiplexer M81 (.clk(clk), .rst(rst), .out(HexVal), .d2(d3));

   always @ (*)
      case(HexVal)
         4'h0: Ca = 8'hC0;
         4'h1: Ca = 8'hF9;
         4'h2: Ca = 8'hA4;
         4'h3: Ca = 8'hB0;
         4'h4: Ca = 8'h99;
         4'h5: Ca = 8'h92;
         4'h6: Ca = 8'h82;
         4'h7: Ca = 8'hF8;
         4'h8: Ca = 8'h80;
         4'h9: Ca = 8'h98;
         4'hA: Ca = 8'h88;
         4'hB: Ca = 8'h83;
         4'hC: Ca = 8'hC6;
         4'hD: Ca = 8'hA1;
         4'hE: Ca = 8'h86;
         4'hF: Ca = 8'h8E;
      default: Ca = 8'hFF;
      endcase

endmodule
```

Anodes.v

```
module Anodes(clk, rst, Anode);
   input clk, rst;
   output [7:0] Anode;
   reg [7:0] Anode,nAnode;

   Rotation RT2 (.clk(clk), .rst(rst), .rotate(rotate));

   always @(posedge clk, posedge rst)
      if (rst) Anode <= 8'hfe;   else
      if (rotate) Anode <= nAnode;

   always @(*)
      case (Anode)
         8'hFE:   nAnode = 8'hFD;
         8'hFD:   nAnode = 8'hFB;
         8'hFB:   nAnode = 8'hF7;
         8'hF7:   nAnode = 8'hEF;
         8'hEF:   nAnode = 8'hDF;
         8'hDF:   nAnode = 8'hBF;
         8'hBF:   nAnode = 8'h7F;
         8'h7F:   nAnode = 8'hFE;
      default:    nAnode = 8'hFF;
      endcase


endmodule
```

TLD.v

```
module TLD(clk, rst, Anode, Cathode, signal);
    input clk, rst, signal;
    output [7:0] Anode, Cathode;
    wire [7:0] Anode, Cathode;

    Anodes AN (.clk(clk),
              .rst(rst),
              .Anode(Anode));

    Hex_7 HX (.clk(clk),
              .rst(rst),
              .Ca(Cathode),
              .d3(signal));

endmodule
```

TLD_tf.v

```
module TLD_tf;

   // Inputs
   reg clk;
   reg rst;

   // Outputs
   wire [7:0] Anode;
   wire [7:0] Cathode;

   // Instantiate the Unit Under Test (UUT)
   TLD uut (
      .clk(clk),
      .rst(rst),
      .Anode(Anode),
      .Cathode(Cathode));

   initial begin
      // Initialize Inputs
      clk = 0;
      rst = 0;
      i = 0;


      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here
      for (i=0; i < 256; i = i + 1)
         begin
            clk = $random;
            rst = $random;
            $write(".");
            end
   end

endmodule
```

.ucf

```
NET "clk"   LOC = "E3"  | IOSTANDARD = "LVCMOS33";             #Bank = 35, Pin name = #IO_L12P_T1_MRCC_35,

NET "signal"           LOC=N17 | IOSTANDARD=LVCMOS33; #IO_L9P_T1_DQS_14
NET "rst"           LOC=P17 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_14

NET "Cathode<0>"             LOC=T10 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_A00_D16_14
NET "Cathode<1>"             LOC=R10 | IOSTANDARD=LVCMOS33; #IO_25_14
NET "Cathode<2>"             LOC=K16 | IOSTANDARD=LVCMOS33; #IO_25_15
NET "Cathode<3>"             LOC=K13 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_A26_15
NET "Cathode<4>"             LOC=P15 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_14
NET "Cathode<5>"             LOC=T11 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A10_D26_14
NET "Cathode<6>"             LOC=L18 | IOSTANDARD=LVCMOS33; #IO_L4P_T0_D04_14
NET "Cathode<7>"             LOC=H15 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A21_VREF_15

NET "Anode<0>"          LOC=J17 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_FOE_B_15
NET "Anode<1>"          LOC=J18 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_FWE_B_15
NET "Anode<2>"          LOC=T9 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_A01_D17_14
NET "Anode<3>"          LOC=J14 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A22_15
NET "Anode<4>"          LOC=P14 | IOSTANDARD=LVCMOS33; #IO_L8N_T1_D12_14
NET "Anode<5>"          LOC=T14 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_14
NET "Anode<6>"          LOC=K2 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_35
NET "Anode<7>"          LOC=U13 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_A02_D18_14
```