Class Report 1: Rotating Square Circuit

Maddie Vorhies

September 5, 2023

Github Repository

https://github.com/MaddieVorhies/ELC5396_ClassReport1.git

Summary

The purpose if this exercise was to be able to design a circuit that would circulate a square pattern in the four-digit seven-segment LED display. The square should be able to rotate both clockwise and counter-clockwise around the display. The user should also be able to pause and reset the location.

The SW0, SW1, and SW2 switches on the board are used to enable the rotation, choose the direction of the rotation, and reset the rotation. The circuit consists of a clockwise module that takes care of the clockwise rotation, a counter-clockwise module that takes care of the counter-clockwise rotation, two muxes that allow the user to select the clockwise or the counter-clockwise module, and an overall wrapper that brings the previous modules together.

Design Sketch

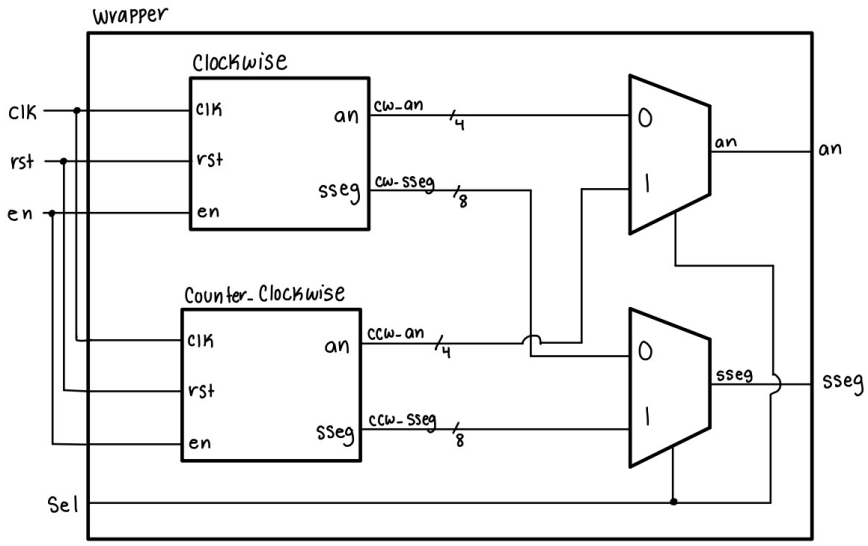

Figure 1: Design Sketch for Rotating Square Circuit

Results

The board was able to successfully rotate a sqaure around the seven-segment display in both the clockwise and counter-clockwise direction. It was also able to pause and reset the rotation of the square.

A testbench was created for clockwise.sv, counter_clockwise.sv, and mux.sv. The results of each of those testbenches are shown below:

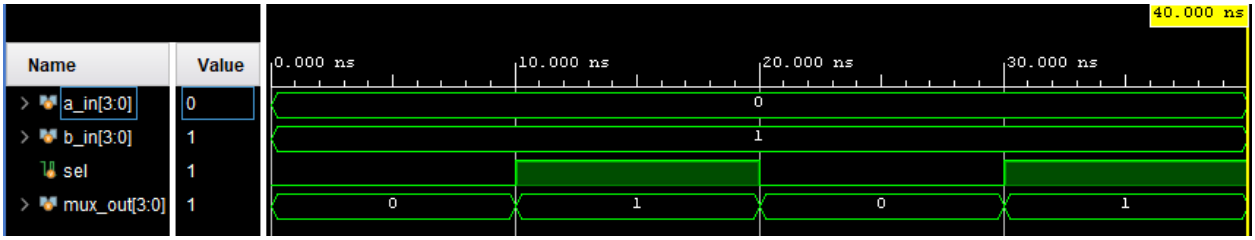

Figure 2: Simulation Waveform and of mux.sv

Figure 2 shows the results a simple testbench that tests to see if the mux will switch from a_in to b_in when the control signal (sel) is changed. When sel is high, the output of the mux should read 1 and when sel is low, the output of the mux should read 0.

Figure 3 shows the results of the testbench that tests the rotation of the square in a clockwise direction. The an signal shows which digit the square will light up on and the sseg signal shows if the square will be on the top half or bottom half of the digit. This testbench also tests to see if the rotation will pause when the enable (en) signal is low.

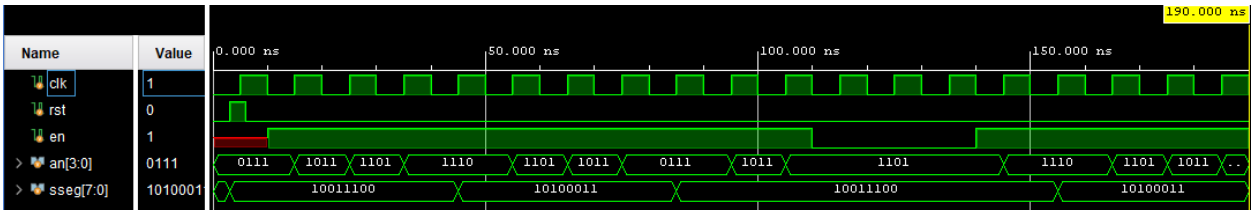

Figure 3: Simulation Waveform and of clockwise.sv

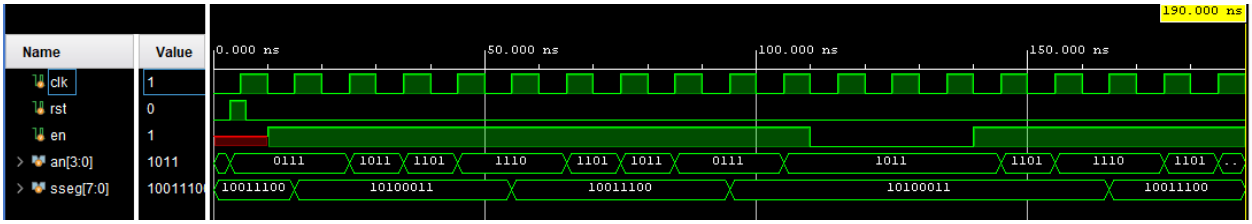

Figure 4: Simulation Waveform and of counter_clockwise.sv

Figure 4 shows the results of the testbench that tests the rotation of the square in a counter-clockwise direction. This testbench is set up identically to the clockwise testbench. The difference is waveforms is shown in the sseg signal. Both waveforms in Figure 3 and Figure 4 have the same an signal, but their sseg signals are opposite of each other. This is to be expected. That is how the direction of the square is adjusted from clockwise to counter-clockwise.

Code

```
‘timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/04/2023 05:04:32 PM
// Design Name:
// Module Name: clockwise
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////
```

```
module clockwise #(parameter N=28) (
    input logic clk, rst, en,
    output logic [3:0] an,
    output logic [7:0] sseg
);

    logic [N-1:0] q_reg;
    logic [N-1:0] q_next;

    always_ff @(posedge clk, posedge rst)
        if (rst)
            q_reg <= 0;
        else if (en)
            q_reg <= q_next;

    assign q_next = q_reg + 1;

    always_comb
        case (q_reg[N-1:N-3])
            3'b000:
                begin
                    an = 4'b0111;
                    sseg = 8'b10011100;
                end
            3'b001:
                begin
                    an = 4'b1011;
                    sseg = 8'b10011100;
                end
            3'b010:
                begin
                    an = 4'b1101;
                    sseg = 8'b10011100;
                end
            3'b011:
                begin
                    an = 4'b1110;
                    sseg = 8'b10011100;
                end
            3'b100:
                begin
                    an = 4'b1110;
                    sseg = 8'b10100011;
                end
            3'b101:
                begin
                    an = 4'b1101;
                    sseg = 8'b10100011;
                end
            3'b110:
                begin
                    an = 4'b1011;
                    sseg = 8'b10100011;
```

```
                end
            default:
                begin
                    an = 4'b0111;
                    sseg = 8'b10100011;
                end
        endcase

endmodule
```

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/04/2023 05:27:04 PM
// Design Name:
// Module Name: counter_clockwise
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/04/2023 05:04:32 PM
// Design Name:
// Module Name: clockwise
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module counter_clockwise #(parameter N=28) (
    input logic clk, rst, en,
    output logic [3:0] an,
    output logic [7:0] sseg
);

    logic [N-1:0] q_reg;
    logic [N-1:0] q_next;

    always_ff @(posedge clk, posedge rst)
        if (rst)
            q_reg <= 0;
        else if (en)
            q_reg <= q_next;

    assign q_next = q_reg + 1;

    always_comb
        case (q_reg[N-1:N-3])
            3'b000:
                begin
                    an = 4'b0111;
                    sseg = 8'b10011100;
                end
            3'b001:
                begin
                    an = 4'b0111;
                    sseg = 8'b10100011;
                end
            3'b010:
                begin
                    an = 4'b1011;
                    sseg = 8'b10100011;
                end
            3'b011:
                begin
                    an = 4'b1101;
                    sseg = 8'b10100011;
                end
            3'b100:
                begin
                    an = 4'b1110;
                    sseg = 8'b10100011;
                end
            3'b101:
                begin
```

```
                    an = 4'b1110;
                    sseg = 8'b10011100;
                end
            3'b110:
                begin
                    an = 4'b1101;
                    sseg = 8'b10011100;
                end
            default:
                begin
                    an = 4'b1011;
                    sseg = 8'b10011100;
                end
        endcase

endmodule
```

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/04/2023 05:33:57 PM
// Design Name:
// Module Name: mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module mux# (parameter N=8)(
    input logic [N-1:0] a_in,
    input logic [N-1:0] b_in,
    input logic sel,
    output logic [N-1:0] mux_out
);

   assign mux_out = sel ? b_in : a_in;

endmodule
```

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/04/2023 06:03:18 PM
// Design Name:
// Module Name: wrapper
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module wrapper(
    input logic clk, rst, en, sel,
    output logic [7:0] an,
    output logic [7:0] sseg
    );

    logic [3:0] cw_an;
    logic [3:0] ccw_an;
    logic [7:0] cw_sseg;
    logic [7:0] ccw_sseg;

    clockwise myClockwise (
        .clk(clk),
        .rst(rst),
        .en(en),
        .an(cw_an),
        .sseg(cw_sseg)
    );

    counter_clockwise myCounterClockwise (
        .clk(clk),
        .rst(rst),
        .en(en),
        .an(ccw_an),
        .sseg(ccw_sseg)
    );
```

```
    mux #(4) mux_an (
      .a_in(cw_an),
      .b_in(ccw_an),
      .sel(sel),
      .mux_out(an)
   );

   mux #(8) mux_sseg (
      .a_in(cw_sseg),
      .b_in(ccw_sseg),
      .sel(sel),
      .mux_out(sseg)
   );

   assign an[7:4] = 4'b1111;

endmodule
```