Rotating Square Circuit

Andrew Clinkenbeard

September 9, 2021

1 Introduction

In this assignment, a rotating square pattern was displayed on a four digit seven segment display. Displayed in the figure below.

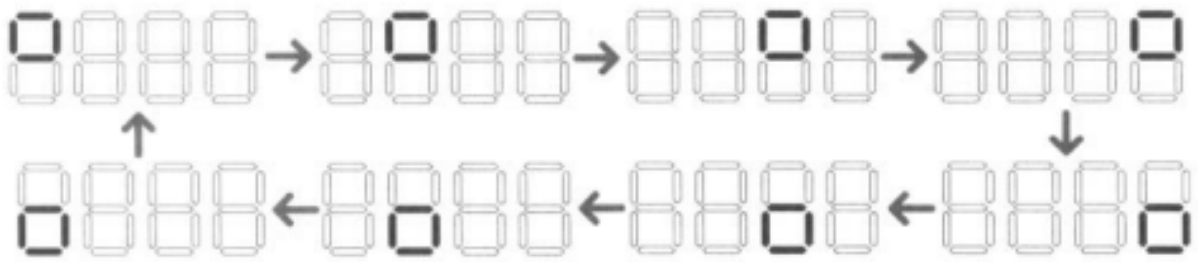

Figure 1: Desired Pattern

2 Method

2.1 Counter

The task was completed by creating a LED time-multiplexing circuit. The code below shows the counter that was used to acomplish the time-mulitplexing.

```
module counter# (parameter N=18, parameter M=2)(
    input logic clk,
    input logic rst,
    output logic [M-1:0] count
    );

    logic [N-1:0] state, nstate;

    always_ff @(posedge(clk), posedge(rst))
        if (rst)
            state<=0;
```

```
        else
            state<= nstate;

    assign nstate=state+1;

    assign count = state[N-1:N-M];


endmodule
```

2.2 Seven Segment Driver

The counter was then instantiated in a driver for the seven segment displays. This driver used the counter to determine which anode to turn on. The codes is shown below.

```
module ssegdriver(
    input logic clk,
    input logic rst,
    input logic [7:0] ss0,
    input logic [7:0] ss1,
    input logic [7:0] ss2,
    input logic [7:0] ss3,
    output logic [7:0] an,
    output logic [7:0] sseg
    );
    logic [1:0] count;

    counter# (.N(18), .M(2)) mycounter(
    .clk(clk),
    .rst(rst),
    .count(count)
    );

    always_comb
    case(count)
    0: begin
        sseg = ss0;
        an=4'b1110;
        end
    1: begin
        sseg = ss1;
        an=4'b1101;
        end
    2: begin
        sseg = ss2;
        an=4'b1011;
        end
```

```
    default: begin
        sseg = ss3;
        an=4'b0111;
        end
    endcase

    // set unused anodes to not be on
    assign an[7:4] = 4'b1111;
endmodule
```

2.3 Pattern

The counter was also instantiated in a module that chooses which step in the pattern the device is currently in. The code is shown below.

```
module Patterns(
    input logic clk,
    input logic rst,
    output logic [7:0] ss0,
    output logic [7:0] ss1,
    output logic [7:0] ss2,
    output logic [7:0] ss3
    );

    logic [2:0] count;
    parameter TOP_SQUARE = 8'b10011100;
    parameter BOTM_SQUARE = 8'b10100011;
    parameter BLANK = 8'b11111111;

    counter# (.N(28), .M(3)) mycounter(
    .clk(clk),
    .rst(rst),
    .count(count)
    );

    always_comb
    case(count)
    0:
        begin
        ss0=BLANK;
        ss1=BLANK;
        ss2=BLANK;
        ss3=TOP_SQUARE;
        end

    1:
        begin
```

```
        ss0=BLANK;
        ss1=BLANK;
        ss2=TOP_SQUARE;
        ss3=BLANK;
        end

    2:
        begin
        ss0=BLANK;
        ss1=TOP_SQUARE;
        ss2=BLANK;
        ss3=BLANK;
        end

    3:
        begin
        ss0=TOP_SQUARE;
        ss1=BLANK;
        ss2=BLANK;
        ss3=BLANK;
        end

    4:
        begin
        ss0=BOTM_SQUARE;
        ss1=BLANK;
        ss2=BLANK;
        ss3=BLANK;
        end

    5:
        begin
        ss0=BLANK;
        ss1=BOTM_SQUARE;
        ss2=BLANK;
        ss3=BLANK;
        end

    6:
        begin
        ss0=BLANK;
        ss1=BLANK;
        ss2=BOTM_SQUARE;
        ss3=BLANK;
        end

    7:
```

```
        begin
        ss0=BLANK;
        ss1=BLANK;
        ss2=BLANK;
        ss3=BOTM_SQUARE;
        end

    endcase

endmodule
```

2.4 Main

Finally the above modules were instantiated in a top level module which connected the modules together to enable the pattern to show up on the physical hardware. The code is shown below.

```
module ssegmain(
    input clk,
    input reset_n,
    output [7:0] sseg,
    output [7:0] an
    );


    // wires to connect ss# across different modules
    logic [7:0] ss0, ss1, ss2, ss3;

    // seven segment driver decleration
    ssegdriver disp_unit(
    .clk(clk),
    .rst(rest_n),
    .ss0(ss0),
    .ss1(ss1),
    .ss2(ss2),
    .ss3(ss3),
    .an(an),
    .sseg(sseg)
    );


    // patternts module decleration
    Patterns(
    .clk(clk),
    .rst(rest_n),
    .ss0(ss0),
    .ss1(ss1),
```

```
    .ss2(ss2),
    .ss3(ss3)
    );

endmodule
```

3 Testing

An expected error was the pattern moving counter-clockwise instead clock-wise. The fix for this was to reverse the order in the Patterns file of which patter appeared when. Another expected error was to have inccorect timing. This would have caused the LEDs to not turn on and off fast enough to be seen as continuous by humans.

4 Results

The pattern matched the intdeded result. The square was able to rotate around the display.

5 Conclusion

In conclusion, the pattern was displayed as wanted. The seven segment displays were used as expected. The resulting pattern can be seen at this link.