Rotating Square Circuit

Andrew Clinkenbeard

September 21, 2021

1 Introduction

In this assignment, a reaction timer was created. When started up the display will read the welcome message: HI. When the user presses the start button, the display goes dark, and after a random interval from 2 to 15 seconds the stimulus LED will light up. At this point the user must press the react button as the time elapsed is shown on the seven segment display. If the user press the button too early 9999 will display. If the user presses the button too late or 1 second has passed the display will show 1000. At any point if the user hits the reset button, it will take them back to the welcome message displayed.

2 Method

2.1 Overview of Method

The first step in completing this assignment was to create a top level module that contained the state machine and how to move about it. This module allowded for the display of things on the seven segment display by conecting the seven segment driver with a patterns module. The patterns were gained by converting the milliseconds passed in a binary form, passing them through a binary to decimal convertor, which then allowded the appropriate numbers to be displayed on the seven segment display. The random time was gained by inserting bits from a random number generator that allowded a counter that to count down at random intervals between runs.

```
module state(
    input logic clk,
    input logic reset_n,
    input logic [0:4] btn,
    output logic [7:0] sseg,
    output logic [7:0] an,
    output logic LED0
    );

    // display parameters
    parameter ZERO = 4'b0000;
    parameter ONE = 4'b0001;
```

```
parameter TWO = 4'b0010;
parameter THREE = 4'b0011;
parameter FOUR = 4'b0100;
parameter FIVE = 4'b0101;
parameter SIX= 4'b0110;
parameter SEVEN= 4'b0111;
parameter EIGHT= 4'b1000;
parameter NINE= 4'b1001;
parameter H = 4'b1010;
parameter I = 4'b1011;
parameter BLANK  = 4'b1100;


typedef enum logic [2:0] {HI, WAIT_REACT, START_REACTION, EARLY,
    TEST, LATE, DONE} State;
State current_state;
State next_state;
logic start_btn;
logic react_btn;
logic rest_btn;

logic [15:0] display_num;
logic [7:0] pattern_2_sseg3;
logic [7:0] pattern_2_sseg2;
logic [7:0] pattern_2_sseg1;
logic [7:0] pattern_2_sseg0;


logic time_rest;
logic [15:0] ms_ticks;
logic [15:0] bcd_2_pattern;
logic [15:0] random = 16'b0000010111011100;


logic [15:0] score;
logic [7:0] my_random;


assign rst = rest_btn;
always_ff @(posedge clk, posedge rst)
    if (rst)
        current_state <= HI;
    else
        current_state <= next_state;


always_comb @(posedge clk)
```

```
case(current_state)
HI:
    begin
    display_num = {H,I, BLANK , BLANK };
    LED0 = 0;
    time_rest = 1'b1;
    if (start_btn)
        begin
        next_state = WAIT_REACT;
        time_rest = 1'b1;
        random = {3'b000, my_random, 5'b11000};
        end
    else
        next_state = HI;
    end
WAIT_REACT:
    begin
    time_rest = 1'b0;
    display_num = { BLANK , BLANK , BLANK , BLANK};
    if (ms_ticks >= random)
        begin
        next_state = START_REACTION;
        LED0 = 1;
        //time_rest = 1'b1;
        end
    else if (react_btn)
        next_state = EARLY;
    else
        next_state = WAIT_REACT;
    end
START_REACTION:
    begin
    LED0 = 1;
    display_num = bcd_2_pattern;
    time_rest = 1'b1;
    next_state = TEST;

    end

EARLY:
    begin
        LED0 = 0;
        display_num = { NINE,NINE,NINE,NINE};
    end

TEST:
    begin
```

```
        display_num = bcd_2_pattern;
        time_rest = 0;
        if (react_btn)
            begin
            score = bcd_2_pattern;
            next_state = DONE;
            end

            if (ms_ticks > 1000)
                next_state = LATE;
        end

    LATE:
        begin
            LED0 = 0;
            display_num = { ONE,ZERO,ZERO,ZERO};
        end

    DONE:
        begin
            LED0 = 0;
            display_num = score;
        end

    endcase

    count_up ms_counter(
    .clk(clk),
    .rst(time_rest),
    .time_elapsed(ms_ticks)
    );



    bin_2_bcd my_bin2bcd   (
    .bin(ms_ticks),
    .bcd0(bcd_2_pattern[3:0]),
    .bcd1(bcd_2_pattern[7:4]),
    .bcd2(bcd_2_pattern[11:8]),
    .bcd3(bcd_2_pattern[15:12])
    );

    ssegdriver myssegdriver(
    .clk(clk),
    .rst(!reset_n),
    .ss0(pattern_2_sseg0),
    .ss1(pattern_2_sseg1),
```

```
.ss2(pattern_2_sseg2),
.ss3(pattern_2_sseg3),
.an(an),
.sseg(sseg)
);


Patterns myPattern(
.number(display_num),
.sseg_pattern3(pattern_2_sseg3),
.sseg_pattern2(pattern_2_sseg2),
.sseg_pattern1(pattern_2_sseg1),
.sseg_pattern0(pattern_2_sseg0)
);



// debounce buttons
// start button
debounce_imm# (.N(4)) debounce0(
.clk(clk),
.rst(!reset_n),
.in(btn[3]),
.out(start_btn)
);

// react button
debounce_imm# (.N(4)) debounce1( // start button
.clk(clk),
.rst(!reset_n),
.in(btn[4]),
.out(react_btn)
);

// reset button
debounce_imm# (.N(4)) debounce2( // stop button
.clk(clk),
.rst(!reset_n),
.in(btn[1]),
.out(rest_btn)
);


prbs#  (.N(25), .M(8))my_rdm_number(
.clk(clk),
.rst(!reset_n) ,
.rdm_num(my_random)
```

```
    );

endmodule
```

2.2 Counter

Below is the counter which was used to count the milliseconds passed since the start button was pressed.

```
module count_up(
    input logic clk,
    input logic rst,
    output logic [15:0] time_elapsed
    );

    parameter time_up  = 20'b00011000011010100000;
    logic [31:0] counting;
    logic [31:0] ncount;
    logic [15:0] next_time_elapsed;

    always_ff @(posedge clk, posedge rst)
        if (rst)
            begin
            time_elapsed <= 0;
            counting <= 0;
            end
        else
            begin
            counting <= ncount;
            time_elapsed <= next_time_elapsed;
            end

    always_comb
        if (counting == time_up)
            begin
                ncount = 0;
                next_time_elapsed = time_elapsed + 1;
            end
        else
            begin
                ncount = counting +1;
                next_time_elapsed = time_elapsed;
            end
```

```
endmodule
```

2.3 Binary to Decimal

Below is the binary to decimal convertor which allowded the time to be displayed on the seven segment.

```
module bin_2_bcd(
    input logic [15:0] bin,
    output logic [3:0] bcd0,
    output logic [3:0] bcd1,
    output logic [3:0] bcd2,
    output logic [3:0] bcd3

    );

integer i;

    always_comb
    begin
        bcd3 = 4'd0;
        bcd2 = 4'd0;
        bcd1 = 4'd0;
        bcd0 = 4'd0;


        for (i=15; i>=0; i--)
        begin
            if(bcd3 >= 5)
                bcd3 += 3;
            if(bcd2 >= 5)
                bcd2 += 3;
            if(bcd1 >= 5)
                bcd1 += 3;
            if(bcd0 >= 5)
                bcd0 += 3;

            bcd3 = bcd3 << 1;
            bcd3[0]=bcd2[3];
            bcd2 = bcd2 << 1;
            bcd2[0]=bcd1[3];
            bcd1 = bcd1 << 1;
            bcd1[0]=bcd0[3];
            bcd0 = bcd0 << 1;
```

```
            bcd0[0]=bin[i];
        end
    end
endmodule
```

2.4 Random Number

Finally, below is the random number generator which was used to get a radnom time between 2 and 15 seconds.

3 Testing

To test this, first several runs were done where no reaction time was pressed. Instead the time it took from the start button to the stimulus LED coming on was measured in order to ensure the randomness as well as the 1000 would show when the user was too slow. Next several runs were done were the start button and the react button were hit closer together in order to show that it displayed 9999 when the user was early. Finally several normal runs were done in order to show that the reaction time worked as inteded.

4 Results

The reaction timer worked as wanted. All states were able to be met, and the reset button brought the user back to the HI display at anypoint.

5 Conclusion

In conclusion, the reaction timer gave a good way to understand both statemachines as well as timing. It also gave the ability to see how random numbers are actually generated. The video of the working device can be seen at this link.