ELC 5396 02 Class Report 1

Elizabeth Kooiman

September 5, 2023

Summary

The purpose of the project was to design and implement a system on the Nexys4 DDR board that displayed a pattern of rotating squares on the seven segment display. The implementation required an option to reset the display, pause the display, and switch between the clockwise and counterclockwise directions. The design implemented used a mux and a counter to implement the desired results.

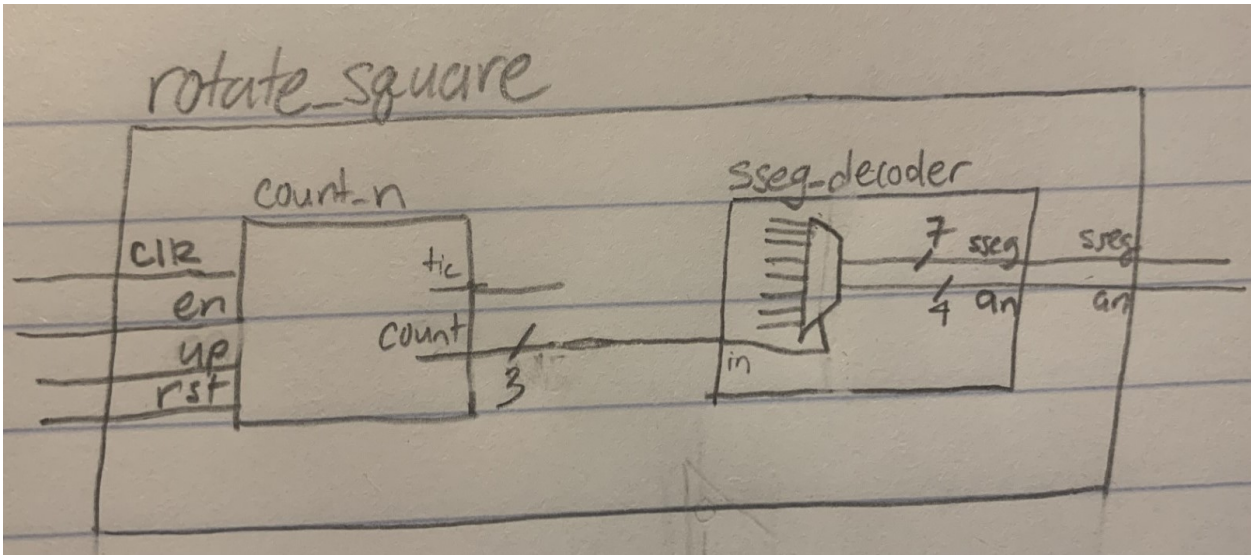

Figure 1: Design Block Diagram

A block diagram of the system designed is shown in Figure 1. Two sub modules were created: a counter and a seven segment (sseg) display decoder. The counter module includes inputs of the clock (clk), a count enable (en), a direction signal (up), and a reset (rst). The code for this module is shown in Listing 1. This module takes the inputs and increments a piece of memory that is used to keep track of the current count according to the desired behavior. The output of this module is a tick signal which is unused in subsequent modules and the current count. The sseg decoder module is shown in Listing 2. The most significant 3 bits of the count output from the module is used as the input to a sseg decoder module. This allows for the differentiation between 2^3, or 8 different states. There are eight different squares that can be illuminated in the pattern, and each state corresponds to lighting a specific square within the pattern. The module uses a multiplexer with the count input as the select signal to determine which predefined value of the

sseg and an signal to output to illuminate the correct square on the sseg LED. The sseg and an outputs of this module are connected to the board to impliment the design. The rotate square module shown in Listing 3 is used to create instances of both a count module and a sseg decoder module. It connects these together and produces the desired output. After this is completed, a wrapper module shown in Listing 5 was used to connect the final design code to the physical implimentation on the board using a constraints file. After the wrapper module was complete, the design could be programmed onto the board. In the final implementation, sw[0] on the board was used as the clockwise/counterclockwise signal, sw[1] was used as the enable, and sw[2] was used as the reset.

Results

Before programming the Nexys4 DDR board, the rotate square module was simulated to determine the desired behavior was being performed. The simulation test bench is shown in Listing 4. The test bench sets the value of clk, rst, en, and up in various orders to ensure the proper functioning of the system in different circumstances. An expected results table is shown in Tables 1, 2, 3, and 4. The resulting waveform is shown in Figure 2. The simultation matched the expected results for the operation of the system. The an signal correctly set which of the four sseg instances were illuminated by alternating between the hex values e, d, b, and 7 to turn on the correct square to illuminate. The sseg signal correctly alternated between the hex numbers 23 and 1c as it selected whether the top or bottom square was supposed to be illuminated as time passed. These correct outputs indicated the system correctly contolled the sseg to achieve the desired pattern.

Table 1: Expected Results Table 1 rotate square

Time(ns):	0	10	20	30	40	50	60	70	80	90
rst:	1	0	0	0	0	0	0	0	0	0
up:	X	1	1	1	1	1	1	1	1	1
en:	X	1	1	1	1	1	1	1	1	1
an:	e	e	e	d	d	b	b	7	7	7
sseg:	23	23	23	23	23	23	23	23	23	23

Table 2: Expected Results Table 2 rotate square

Time(ns):	100	110	120	130	140	150	160	170	180	190
rst:	0	0	0	0	0	0	0	0	0	0
up:	1	1	1	1	1	1	1	1	1	1
en:	1	1	0	0	0	0	0	0	0	0
an:	7	7	b	b	b	b	b	b	b	b
sseg:	1c	1c	1c	1c	1c	1c	1c	1c	1c	1c

Table 3: Expected Results Table 3 rotate square

Time(ns):	200	210	220	230	240	250	260	270	280	290
rst:	0	0	0	0	0	0	0	0	0	0
up:	1	1	1	1	1	1	1	1	1	1
en:	0	0	1	1	1	1	1	1	1	1
an:	b	b	b	b	d	d	e	e	e	e
sseg:	1c	1c	1c	1c	1c	1c	1c	1c	23	23

Table 4: Expected Results Table 4 rotate square

Time(ns):	300	310	320	330	340	350	360	370	380	390
rst:	0	0	0	0	0	0	0	0	0	0
up:	1	1	0	0	0	0	0	0	0	0
en:	1	1	1	1	1	1	1	1	1	1
an:	d	d	b	d	d	e	e	e	e	d
sseg:	23	23	23	23	23	23	23	1c	1c	1c

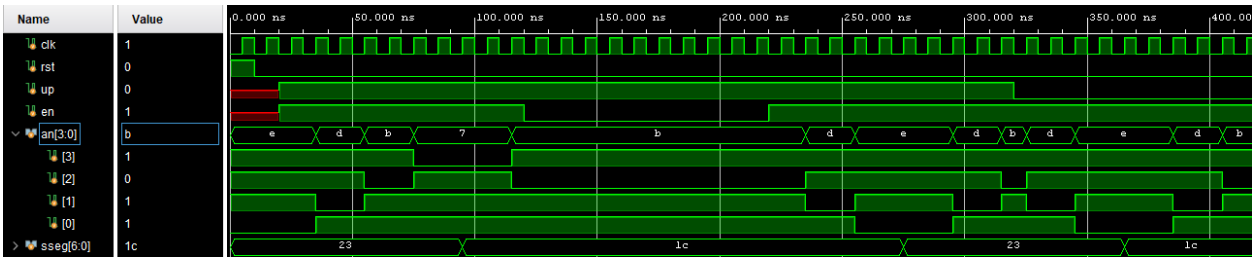

Figure 2: Simulation Waveform Results

Code

Listing 1: count module

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////
// Company:
// Engineer: Elizabeth Kooiman
// Create Date: 09/05/2023 03:02:20 PM
// Module Name: count_n
// Project Name: Class Report 1
//////////////////////////////////////////////


module count_n#(parameter N=20)(
    input logic clk,
    input logic rst,
    input logic up,
    input logic en,
    output logic tic,
    output logic [N-1:0] count
    );
    parameter ZERO={N,{1'b0}};
    logic [N-1:0] count_curr;
    logic [N-1:0] count_next;

    always_ff @(posedge(clk),posedge(rst))
    begin
        if (rst) begin
        count_curr <= ZERO;
        end
        //else if(en) begin
        else begin
        count_curr <= count_next;
        end
    end

    always_comb
        if(en) begin
            if(up) begin
                count_next = count_curr +1;
                end
            else begin
                count_next = count_curr -1;
                end
        end
        else begin
            count_next = count_curr;
        end


    assign count = count_curr;
    assign tic = count_curr == 1;
```

```
endmodule
```

Listing 2: sseg decoder module

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////
// Engineer: Elizabeth Kooiman
// Create Date: 08/31/2023 11:40:16 AM
// Module Name: sseg_decoder
// Project Name: Class Report 1
//////////////////////////////////////////////////


module sseg_decoder(
    input logic [2:0] in,
    output logic [6:0] sseg,
    output logic [3:0] an
    );

    //0123
    //4567
always_comb begin
    case(in)
    3'b000:
        begin
        an = 4'b0111;
        sseg = 7'b0011100;
        end
    3'b001:
        begin
        an = 4'b1011;
        sseg = 7'b0011100;
        end
    3'b010:
        begin
        an = 4'b1101;
        sseg = 7'b0011100;
        end
    3'b011:
        begin
        an = 4'b1110;
        sseg = 7'b0011100;
        end
    3'b111:
        begin
        an = 4'b0111;
        sseg = 7'b0100011;
        end
    3'b110:
        begin
        an = 4'b1011;
        sseg = 7'b0100011;
        end
```

```
    3'b101:
        begin
        an = 4'b1101;
        sseg = 7'b0100011;
        end
    3'b100:
        begin
        an = 4'b1110;
        sseg = 7'b0100011;
        end
    default:
        begin
        an = 4'b0111;
        sseg = 7'b1111111;
        end

    endcase
end




endmodule
```

Listing 3: rotate square module

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

// Engineer: Elizabeth Kooiman
// Create Date: 09/05/2023 03:41:58 PM
// Module Name: rotate_square
// Project Name: Class Report 1
//
    //////////////////////////////////////////////////////////////////////////////////


module rotate_square#(parameter N = 29)(
    input logic clk,
    input logic rst,
    input logic up,
    input logic en,
    output logic [3:0] an,
    output logic [6:0] sseg
    );

    logic [N-1:0] count;
    logic tic;


    count_n#(.N(N)) counter(
    .clk(clk),
    .rst(rst),
```

```
    .up(up),
    .en(en),
    .tic(tic),
    .count(count)
    );

    sseg_decoder sseg1(
    .in(count[N-1:N-3]),
    .sseg(sseg),
    .an(an)
    );


endmodule
```

Listing 4: rotate square test module

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

// Engineer: Elizabeth Kooiman
// Create Date: 09/05/2023 05:56:28 PM
// Module Name: rotate_square_t
// Project Name: Class Report 1
//
   //////////////////////////////////////////////////////////////////////////////////


module rotate_square_t();

logic clk;
logic rst;
logic up;
logic en;
logic [3:0] an;
logic [6:0] sseg;
parameter N = 4;
rotate_square #(.N(N)) uut(
    .clk(clk),
    .rst(rst),
    .up(up),
    .en(en),
    .an(an),
    .sseg(sseg)
    );

    initial begin
    clk = 0;
    forever #5 clk = ~clk;
    end

    initial begin
    rst = 1;
```

```
    #10;
    rst = 0;
    #10;

    //TEST CLOCKWISE
    en = 1;
    up = 1;
    #100;
    en = 0;
    #100;
    en = 1;
    #100;
    up = 0;
    #100;
    $finish;
    end


endmodule
```

Listing 5: top wrapper module

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

// Engineer: Elizabeth Kooiman
//
// Create Date: 09/05/2023 03:57:46 PM
// Module Name: top_class_report_1
// Project Name: Class Report 1
//
    //////////////////////////////////////////////////////////////////////////////////



module top_class_report_1(
    input logic clk,
    input logic en,
    input logic cw,
    input logic rst,
    output logic [7:0] AN,
    output logic [6:0] seg
    );

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

    rotate_square rotate(
    .clk(clk),
    .rst(rst),
    .up(cw),
    .en(en),
    .an(AN[3:0]),
    .sseg(seg)
```

```
    );
endmodule
```