Softcore System on a Chip Lab 1: Rotating Square Circuit

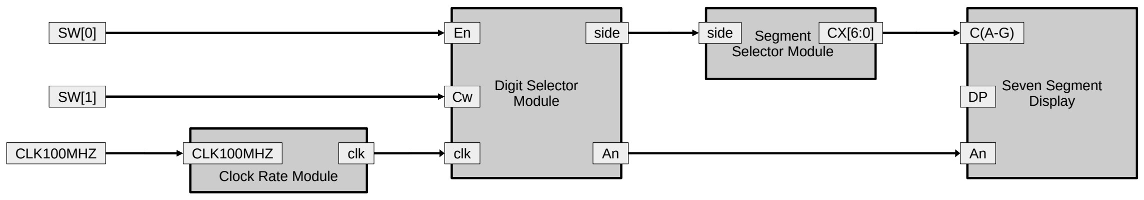
David Edeni and Zach Martin

September 11th, 2025

Lab Description:

This project implements a rotating square circuit on a four-digit seven-segment LED display. A square pattern is generated using segments (a, b, f, g) or (c, d, e, g). The circuit circulates the square around the four digits, with two control inputs: en (enable/pause) and cw (direction, cw = 1 means clockwise and cw = 0 means counterclockwise). The en input signal enables or pauses the circulation (en = 1 means the circuit is enabled/active and en = 0 means inactive) and the cw input signal specifies the direction of the circulation.

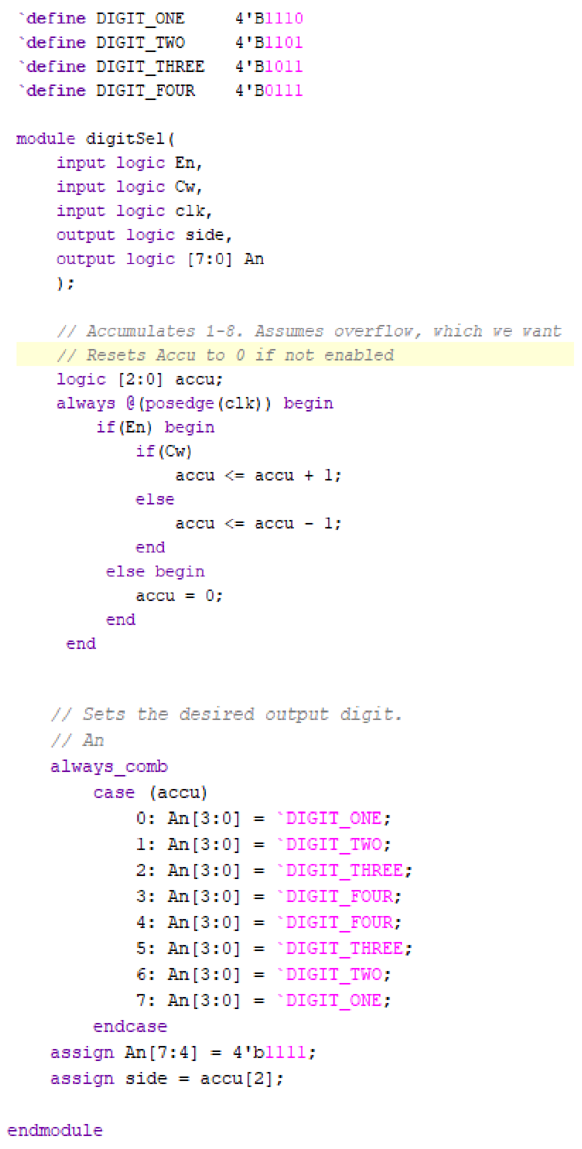
To implement our four-digit seven-segment LED rotating square circuit, we designed the following Block Diagram that produces a square pattern on the LED display at a rate visible to the human eye.



To implement this Block Diagram, we utilized four System Verilog Modules: digitSel.sv, segSel.sv, clockRate.sv, and top.sv.

The digitSel System Verilog module selects which digit of a four-digit seven-segment LED display should be active at a given time, and it rotates the active digit around the display in either a clockwise or counterclockwise direction based on the cw control signal. This module drives the digit enable (An) signal to control which of the four digits on the LED display lights up, while also generating a side signal that indicates which ”half” of the rotation you are in.

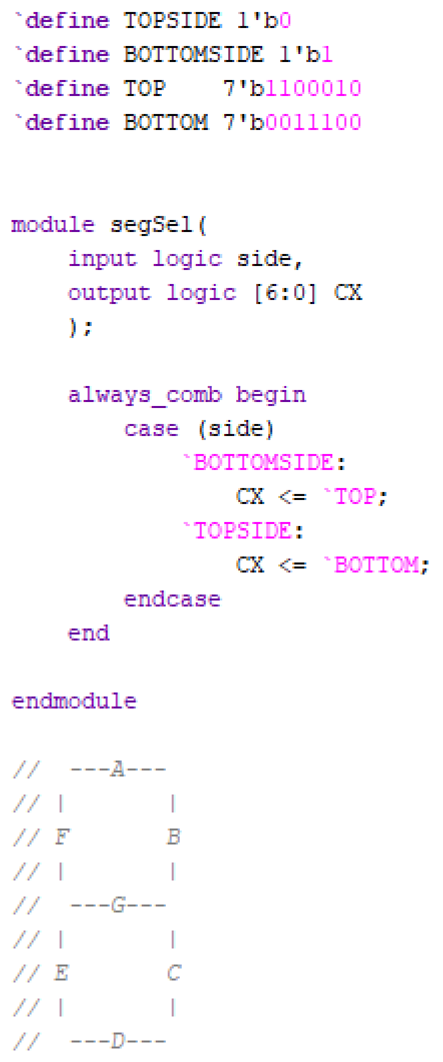
Figure 1: The digitSel (digit selector) module:



```
`define DIGIT_ONE     4'B1110
`define DIGIT_TWO     4'B1101
`define DIGIT_THREE   4'B1011
`define DIGIT_FOUR    4'B0111

module digitSel(
    input logic En,
    input logic Cw,
    input logic clk,
    output logic side,
    output logic [7:0] An
    );

    // Accumulates 1-8. Assumes overflow, which we want
    // Resets Accu to 0 if not enabled
    logic [2:0] accu;
    always @(posedge(clk)) begin
        if(En) begin
            if(Cw)
                accu <= accu + 1;
            else
                accu <= accu - 1;
            end
         else begin
            accu = 0;
         end
     end


    // Sets the desired output digit.
    // An
    always_comb
        case (accu)
            0: An[3:0] = `DIGIT_ONE;
            1: An[3:0] = `DIGIT_TWO;
            2: An[3:0] = `DIGIT_THREE;
            3: An[3:0] = `DIGIT_FOUR;
            4: An[3:0] = `DIGIT_FOUR;
            5: An[3:0] = `DIGIT_THREE;
            6: An[3:0] = `DIGIT_TWO;
            7: An[3:0] = `DIGIT_ONE;
        endcase
    assign An[7:4] = 4'b1111;
    assign side = accu[2];

endmodule
```

The segSel System Verilog module determines which segments (LED bars) of a seven-segment digit display should be lit up, depending on whether the ”rotating square” is on the top side or bottom side of the display. This module selects whether the board’s active digit shows the square’s top or bottom half.

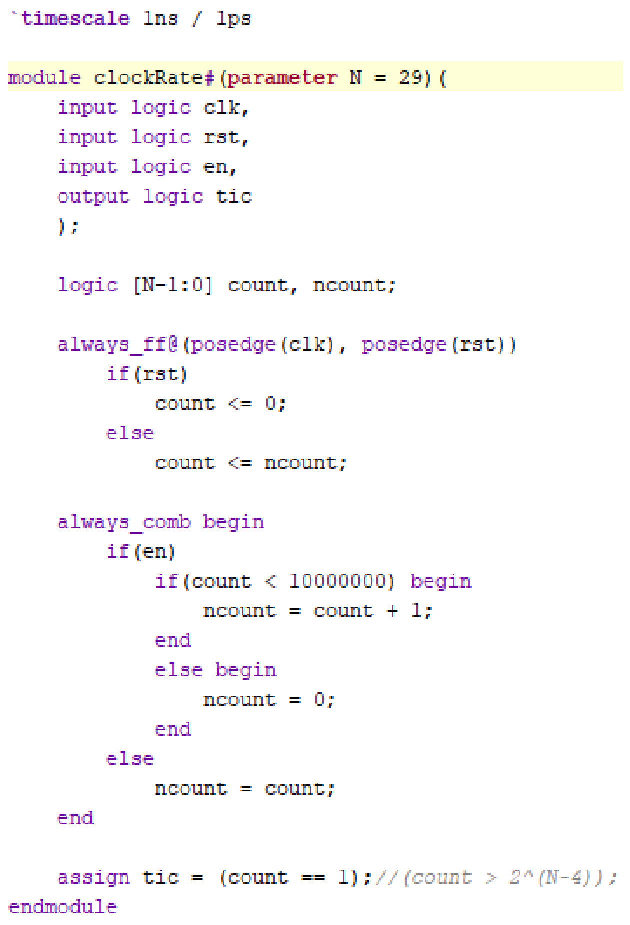
Figure 2: The segSel (segment selector) module:



```
`define TOPSIDE 1'b0
`define BOTTOMSIDE 1'b1
`define TOP    7'b1100010
`define BOTTOM 7'b0011100


module segSel(
    input logic side,
    output logic [6:0] CX
    );

    always_comb begin
        case (side)
            `BOTTOMSIDE:
                CX <= `TOP;
            `TOPSIDE:
                CX <= `BOTTOM;
        endcase
    end

endmodule

//  ---A---
// |       |
// F       B
// |       |
//  ---G---
// |       |
// E       C
// |       |
//  ---D---
```

The clockRate System Verilog module is a clock divider / rate generator. The FPGA's main clock (100 MHz) is way too fast for human eyes, so to properly animaye the rotating square, you need a slower pulse (”tic”). This module divides the high-speed clock down to generate a visible slower timing signal.

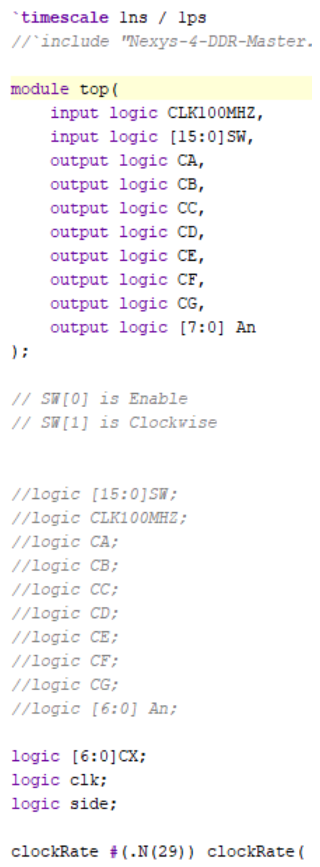
Figure 3: The clockRate (segment selector) module:



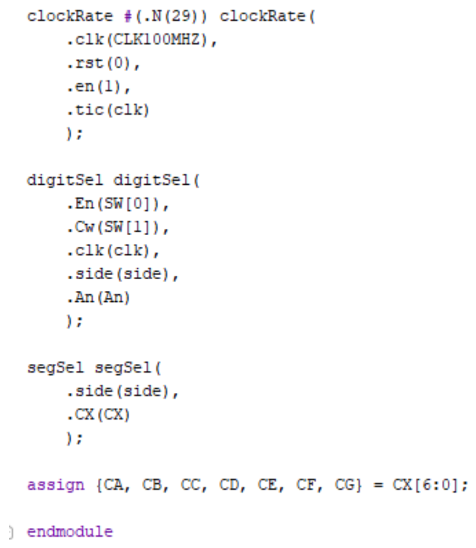
```
`timescale 1ns / 1ps

module clockRate#(parameter N = 29)(
    input logic clk,
    input logic rst,
    input logic en,
    output logic tic
    );

    logic [N-1:0] count, ncount;

    always_ff@(posedge(clk), posedge(rst))
        if(rst)
            count <= 0;
        else
            count <= ncount;

    always_comb begin
        if(en)
            if(count < 10000000) begin
                ncount = count + 1;
            end
            else begin
                ncount = 0;
            end
        else
            ncount = count;
    end

    assign tic = (count == 1);//(count > 2^(N-4));
endmodule
```

Finally, the top System Verilog module is the glue that ties all our submodules (digitSel, segSel, clockRate) together to drive the rotating square circuit on the seven-segment display. This module connects the FPGA's hardware inputs (clock, switches) and outputs (seven-segment LEDs). It instantiates the three functional modules: digitSel, segSel, and clockRate, and wires them together so the square rotates across the 4-digit seven-segment display.

Figure 4: The top (square circuit implementation) module:



```
`timescale 1ns / 1ps
//`include "Nexys-4-DDR-Master.

module top(
    input logic CLK100MHZ,
    input logic [15:0]SW,
    output logic CA,
    output logic CB,
    output logic CC,
    output logic CD,
    output logic CE,
    output logic CF,
    output logic CG,
    output logic [7:0] An
);

// SW[0] is Enable
// SW[1] is Clockwise


//logic [15:0]SW;
//logic CLK100MHZ;
//logic CA;
//logic CB;
//logic CC;
//logic CD;
//logic CE;
//logic CF;
//logic CG;
//logic [6:0] An;

logic [6:0]CX;
logic clk;
logic side;

clockRate #(.N(29)) clockRate(
```



```
clockRate #(.N(29)) clockRate(
    .clk(CLK100MHZ),
    .rst(0),
    .en(1),
    .tic(clk)
    );

digitSel digitSel(
    .En(SW[0]),
    .Cw(SW[1]),
    .clk(clk),
    .side(side),
    .An(An)
    );

segSel segSel(
    .side(side),
    .CX(CX)
    );

assign {CA, CB, CC, CD, CE, CF, CG} = CX[6:0];

endmodule
```