18.11.2019

CS 223-2

LAB-4

PRELIMINARY REPORT

ZÜBEYİR BODUR

21702382

Zübeyir Bodur

Trainer No: 21

2. SystemVerilog Code for the Driver

a) Modules

```
// Simple two bit register


module TwoBitRegister( input logic d[1:0], clk,
                                output logic q[1:0]
    );
    always_ff @(posedge clk) begin
        q <= d;
    end
endmodule


// Next State Logic for the driver


module sNext( input logic e, c, s[1:0],
                    output logic sNext[1:0]
    );
    logic s0_XNOR_c, s0_XOR_c, s1_e_NOT;
    logic product_1, product_2;


    xnor xnor1( s0_XNOR_c, s[0], c);
    and and1 ( product_1, ~s[1], e, s0_XNOR_c);


    xor xor1( s0_XOR_c, s[0], c);
    and and2( product_2, s[1], s0_XOR_c);


    and and3( s1_e_NOT, s[1], ~e);


    or or2( sNext[1], product_1, product_2, s1_e_NOT );
```

```
        xor xor2( sNext[0], s[0], e);
    endmodule

    /* SystemVerilog code for the driver
     * where y[3:0] = {A, B, Ab, Bb}
     */
    module driverFSM( input logic e, c, clk,
                                output logic y[3:0]
        );
        logic sNext[1:0], s[1:0] = {0, 0}; // initial value for current state
        // CL for s1 next
        sNext nextStateLogic(e, c, s[1:0], sNext[1:0]);

        // Register between states
        TwoBitRegister register( sNext[1:0], clk, s[1:0]);



        // Output logic
        xnor xnor1( y[3], s[1], s[0]);
        not inv1( y[2], s[1]);
        xor xor3( y[1], s[1], s[0]);
        buf buffer( y[0], s[1]);
    endmodule
```

b) Testbench

```
    // Testbench for the driver
```

```
module driverFSMtest();
    logic e, c, clk, y[3:0];
    driverFSM FSM(e, c, clk, y[3:0]);
    initial begin
        clk = 0; e = 1; c = 1; #20;
        repeat(2) begin
            repeat(2) begin
                repeat(8) begin
                    clk = ~clk; #20;
                end
                c = ~c; #20;
            end
            e = ~e; #20;
        end
    $stop;
    end
endmodule
```

2. SystemVerilog Code for the Car

a) Modules

```
/* SystemVerilog code to divide the given clk input so that
* frequency of clk prime is less than 40Hz.
* f_clk prime can also be modified by i bus.
*/
module clkDivider( input logic [1:0]i,
                   input logic clk,
                   input logic rst,
                   output logic clk_prime
    );
    logic [31:0]divider;
```

```
always_comb begin
```

```
            case(i)
                2'b10:
                    divider = 32'b10011000100101101000000;
    // Slowest, which is 75 rev/min, freq is 10^7 Hz

                2'b01:
                    divider = 32'b01001100010010110100000;
    // Medium, which is 150 rev/min, freq is 5*10^6 Hz

                2'b01:
                    divider = 32'b00100110001001011010000;
    // Fastest, which is 300 rev/min, freq is 25*10^5 Hz
                2'b11:
                    divider = 32'd0; // Don't divide the clk
    //signal if we tell car to stop.
            endcase
    end

        logic [31:0]count = 32'b0;
        always_ff @( posedge clk, posedge rst) begin
            if (rst == 1)
                count <= 32'b0;
            else if (count >= divider - 1)
                count <= 32'b0;
            else
                count <= count + 1;
        end

        always_ff @(posedge clk, posedge rst) begin
            if (rst == 1)
```

```
                clk_prime <= 32'b0;
        else if (!rst && divider != 0 && count == divider - 1)
                clk_prime <= ~clk_prime;
        else
                clk_prime <= clk_prime;
    end
endmodule
```

```
/* SystemVerilog code for the car
 * where i == {00} -- fastest
 *       i == {01} -- medium
 *       i == {10} -- slowest
 *       i == {11} -- car stops
 * and   y == {A, B, Ab, Bb}
 */
module car( input logic e, c, i[1:0], clk,
                output logic y[3:0]
    );
    logic clk_prime, i1_NAND_i0, work;
    clkDivider divide( i[1:0], clk, clk_prime);
```

```
    // Car works if enable is 1 and i != {11}
     nand i_bus_NOT_11( i1_NAND_i0, i[1], i[0]);
     and workingCase( work, e, i1_NAND_i0);

     driverFSM driver( work, c, clk_prime, y);
endmodule
```

b) Testbenches

```
// Testbench for the car
module carTest();
     logic e, c, i[1:0], clk, y[3:0];
     car carUUT(e, c, i, clk, y[3:0]);
     initial begin
          clk = 0; e = 1; c = 1; i[1] = 0; i[0] = 0; #20;
          repeat(2) begin
               repeat(2) begin
                    repeat(2) begin
                         repeat(2) begin
                              repeat(8) begin
                                   clk = ~clk; #20;
                              end
                              c = ~c; #20;
                         end
                         i[0] = ~i[0]; #20;
                    end
                    i[1] = ~i[1]; #20;
               end
               e = ~e; #20;
          end
```

```
        $stop;
        end
    endmodule
```