ECE 581 Project 3 Report

Chuck Faber (cfaber@pdx.edu)

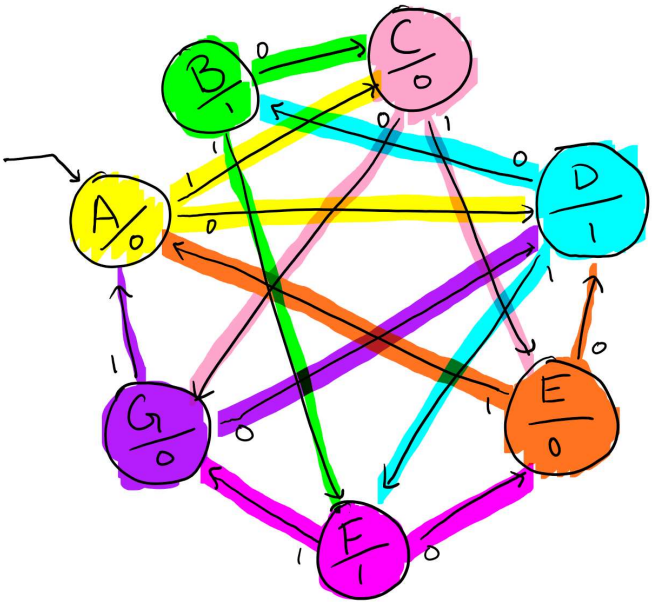
Chris Mersman (cmersman@pdx.edu)

Problem 1

N/A - Completion of Tutorial

Problem 2

FSM Design



Model A Code

```
// Model A - Regular Case Statements and Binary State Encoding
module FSM_A (
    input in, rst, clk,
    output logic out
);
    typedef enum logic[2:0] {SA, SB, SC, SD, SE, SF, SG} states_t;
    states_t cs, ns;

    always_ff @(posedge clk or posedge rst) begin: seq_logic
        if (rst) cs <= SA;
        else cs <= ns;
    end: seq_logic

    always_comb begin: ns_logic
        unique case (cs)
            SA: ns = in ? SC : SD;
            SB: ns = in ? SF : SC;
            SC: ns = in ? SE : SG;
```

```
            SD: ns = in ? SF : SB;
            SE: ns = in ? SA : SD;
            SF: ns = in ? SG : SE;
            SG: ns = in ? SA : SD;
            default: ns = cs;
        endcase
    end: ns_logic

    always_comb begin: out_logic
        unique case (cs)
            SA: out = 1'b0;
            SB: out = 1'b1;
            SC: out = 1'b0;
            SD: out = 1'b1;
            SE: out = 1'b0;
            SF: out = 1'b1;
            SG: out = 1'b0;
            default: out = 1'bx;
        endcase
    end: out_logic

endmodule
```

Model B Code

```
// Model B - Regular Case Statements and One Hot Encoding
module FSM_B (
    input in, rst, clk,
    output logic out
);
    typedef enum logic[6:0] {
        SA = 7'b0000001,
        SB = 7'b0000010,
        SC = 7'b0000100,
        SD = 7'b0001000,
        SE = 7'b0010000,
        SF = 7'b0100000,
        SG = 7'b1000000
        } states_t;
    states_t cs, ns;

    always_ff @(posedge clk or posedge rst) begin: seq_logic
        if (rst) cs <= SA;
        else cs <= ns;
    end: seq_logic

    always_comb begin: ns_logic
        unique case (cs)
            SA: ns = in ? SC : SD;
            SB: ns = in ? SF : SC;
            SC: ns = in ? SE : SG;
            SD: ns = in ? SF : SB;
            SE: ns = in ? SA : SD;
            SF: ns = in ? SG : SE;
            SG: ns = in ? SA : SD;
            default: ns = cs;
```

```
        endcase
    end: ns_logic

    always_comb begin: out_logic
        unique case (cs)
            SA: out = 1'b0;
            SB: out = 1'b1;
            SC: out = 1'b0;
            SD: out = 1'b1;
            SE: out = 1'b0;
            SF: out = 1'b1;
            SG: out = 1'b0;
            default: out = 1'bx;
        endcase
    end: out_logic

endmodule
```

Model C Code

```
// Model C - Reversed Case Statement with One Hot Encoding
module FSM_C (
    input in, rst, clk,
    output logic out
);
    enum {A=0, B=1, C=2, D=3, E=4, F=5, G=6} states_idx;
    typedef enum logic [6:0] {
        SA = 7'b0000001 << A,
        SB = 7'b0000001 << B,
        SC = 7'b0000001 << C,
        SD = 7'b0000001 << D,
        SE = 7'b0000001 << E,
        SF = 7'b0000001 << F,
        SG = 7'b0000001 << G
    } states_t;
    states_t cs, ns;

    always_ff @(posedge clk or posedge rst) begin: seq_logic
        if (rst) cs <= SA;
        else cs <= ns;
    end: seq_logic

    always_comb begin: ns_logic
        unique case (1'b1)
            cs[A]: ns = in ? SC : SD;
            cs[B]: ns = in ? SF : SC;
            cs[C]: ns = in ? SE : SG;
            cs[D]: ns = in ? SF : SB;
            cs[E]: ns = in ? SA : SD;
            cs[F]: ns = in ? SG : SE;
            cs[G]: ns = in ? SA : SD;
            default: ns = cs;
        endcase
    end: ns_logic

    always_comb begin: out_logic
```

```
        unique case (1'b1)
            cs[A]: out = 1'b0;
            cs[B]: out = 1'b1;
            cs[C]: out = 1'b0;
            cs[D]: out = 1'b1;
            cs[E]: out = 1'b0;
            cs[F]: out = 1'b1;
            cs[G]: out = 1'b0;
            default: out = 1'bx;
        endcase
    end: out_logic

endmodule
```

Testbench Code

```
module top();

    parameter NTESTS = 50;
    logic in, rst, clk, outA, outB, outC;
    logic [6:0] trace;

    FSM_A modelA (
        .out(outA),
        .*);

    FSM_B modelB (
        .out(outB),
        .*);

    FSM_C modelC (
        .out(outC),
        .*);

    initial begin: clock
        clk = 1'b1;
        forever #50 clk = ~clk;
    end: clock


    task resetFSM();
        // in = 1'bx;
        @(negedge clk) rst = 1'b1;
        @(negedge clk) rst = 1'b0;
        // $display("\nResetting FSM.");
    endtask

    initial begin: testbench
        for (int i = 0; i < NTESTS; i++) begin
            resetFSM();
            trace = $random() & 7'h7F;
            in = trace;
            repeat (1) @(negedge clk);
            $display($time, " Trace: %07b\tInput: %01b\tOutA: %01b\tOutB: %01b\tOutC: %01b", trace, in,
outA, outB, outC);
        end
```

```
        $finish();
    end: testbench

endmodule
```

SV Model Testbench Transcript

```
VSIM 1> run -all
#                  250 Trace: 0100100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                  550 Trace: 0000001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                  850 Trace: 0001001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 1150 Trace: 1100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 1450 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 1750 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 2050 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 2350 Trace: 0010010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 2650 Trace: 0000001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 2950 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 3250 Trace: 1110110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 3550 Trace: 0111101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 3850 Trace: 1101101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 4150 Trace: 0001100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 4450 Trace: 1111001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 4750 Trace: 1000110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 5050 Trace: 1000101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 5350 Trace: 0101010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 5650 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 5950 Trace: 1110111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 6250 Trace: 0010010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 6550 Trace: 0001111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 6850 Trace: 1110010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 7150 Trace: 1001110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 7450 Trace: 1101000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 7750 Trace: 1000101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 8050 Trace: 1011100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 8350 Trace: 0111101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 8650 Trace: 0101101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 8950 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 9250 Trace: 1100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 9550 Trace: 0001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 9850 Trace: 0000000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                10150 Trace: 0100000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                10450 Trace: 0101010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                10750 Trace: 0011101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                11050 Trace: 0010110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                11350 Trace: 0010011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                11650 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                11950 Trace: 1010011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                12250 Trace: 1101011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                12550 Trace: 1010101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                12850 Trace: 0000010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                13150 Trace: 0101110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                13450 Trace: 0011101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                13750 Trace: 1001111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                14050 Trace: 0100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                14350 Trace: 0001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
```

```
#               14650 Trace: 1001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               14950 Trace: 0111100   Input: 0        OutA: 1 OutB: 1 OutC: 1
# ** Note: $finish    : prob2_tb.sv(52)
#    Time: 14950 ns  Iteration: 1  Instance: /top
```

Netlist Testbench Transcript

```
# ** Warning: (vsim-3448) osu05_stdcells.v(361): Setting negative specify check constraint (-190 ps) to
zero.
#    Time: 0 ps  Iteration: 0  Instance: /top/modelA/\cs_reg[0]
VSIM 1> run -all
#                 250 Trace: 0100100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                 550 Trace: 0000001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                 850 Trace: 0001001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                1150 Trace: 1100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                1450 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                1750 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                2050 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                2350 Trace: 0010010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                2650 Trace: 0000001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                2950 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                3250 Trace: 1110110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                3550 Trace: 0111101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                3850 Trace: 1101101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                4150 Trace: 0001100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                4450 Trace: 1111001   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                4750 Trace: 1000110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                5050 Trace: 1000101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                5350 Trace: 0101010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                5650 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                5950 Trace: 1110111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                6250 Trace: 0010010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                6550 Trace: 0001111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                6850 Trace: 1110010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                7150 Trace: 1001110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                7450 Trace: 1101000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                7750 Trace: 1000101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                8050 Trace: 1011100   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                8350 Trace: 0111101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                8650 Trace: 0101101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                8950 Trace: 1100101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                9250 Trace: 1100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                9550 Trace: 0001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                9850 Trace: 0000000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               10150 Trace: 0100000   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               10450 Trace: 0101010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               10750 Trace: 0011101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               11050 Trace: 0010110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               11350 Trace: 0010011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               11650 Trace: 0001101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               11950 Trace: 1010011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               12250 Trace: 1101011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               12550 Trace: 1010101   Input: 1        OutA: 0 OutB: 0 OutC: 0
#               12850 Trace: 0000010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               13150 Trace: 0101110   Input: 0        OutA: 1 OutB: 1 OutC: 1
#               13450 Trace: 0011101   Input: 1        OutA: 0 OutB: 0 OutC: 0
```

```
#                13750 Trace: 1001111   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                14050 Trace: 0100011   Input: 1        OutA: 0 OutB: 0 OutC: 0
#                14350 Trace: 0001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                14650 Trace: 1001010   Input: 0        OutA: 1 OutB: 1 OutC: 1
#                14950 Trace: 0111100   Input: 0        OutA: 1 OutB: 1 OutC: 1
# ** Note: $finish    : prob2_tb.sv(52)
#    Time: 14950 ns  Iteration: 1  Instance: /top
# End time: 21:47:51 on Nov 18,2021, Elapsed time: 0:00:04
```

Synthesis Table

	Model A	Model B	Model C
# of Total Cells	22	63	30
Total Cell Area	8550	23130	14904
Power Consumption	1.6759 mW	3.1703 mW	2.5122 mW

Problem 3

Gray Adder Design Code

```
// 4 bit binary gray code adder
// Component 1: 2 4-bit Gray 2 Binary Code Converters
// Component 2: 1 4-bit binary adder
// Component 3: 1 5-bit binary 2 gray code converter

// The G2B will convert both operands to binary
// It will feed them to the 4-bit binary adder
// The B2G converter will take the result and the CO and convert to gray code

module gray2bin #(parameter N = 4) (
    output logic [N-1:0] bin,
    input [N-1:0] gray
);

    always_comb begin
        bin[N-1] = gray[N-1];
        for (int i=N-2; i >= 0; i--) begin
            bin[i] = bin[i+1] ^ gray[i];
        end
    end
endmodule

module bin2gray #(parameter N = 4) (
    output logic [N-1:0] gray,
    input [N-1:0] bin
);

    genvar i;
    assign gray[N-1] = bin[N-1];
    generate
        for (i=N-2; i >= 0; i--) begin
```

```
            assign gray[i] = bin[i+1]^bin[i];
        end
    endgenerate

endmodule

module CLA_adder #(parameter nBITS = 4) (
    output logic [nBITS-1:0] sum,
    output logic co,
    input [nBITS-1:0] ain, bin,
    input cin
);
    logic [nBITS-1:0] P, G;
    logic [nBITS:0] C;

    always_comb begin
        P = ain ^ bin;
        G = ain & bin;
        C[0] = cin;
        for (int i=0; i < nBITS; i++) begin
            C[i+1] = G[i] | (P[i] & C[i]);
        end
        sum = P ^ C[nBITS-1:0];
        co = C[nBITS];
    end

endmodule

module gray_adder #(parameter N=4) (
    output logic [N:0] result,
    input [N-1:0] in_A, in_B
    );

    wire [N-1:0] A_bin;
    wire [N-1:0] B_bin;
    wire [N-1:0] sum;
    wire co;

    gray2bin #(N) g2b_A (
        .gray(in_A),
        .bin(A_bin)
    );

    gray2bin #(N) g2b_B (
        .gray(in_B),
        .bin(B_bin)
    );

    CLA_adder #(N) adder (
        .ain(A_bin),
        .bin(B_bin),
        .cin(1'b0),
        .sum(sum),
        .co(co)
    );

    bin2gray #(N+1) b2g (
        .bin({co, sum}),
```

```
        .gray(result)
    );

endmodule
```

Gray Adder Testbench

```
`timescale 1ns/10ps
module top();

    logic [3:0] in_A, in_B;
    logic [4:0] result;

    gray_adder ga(.*);


    initial begin
        #5 in_A = 4'b0000; in_B = 4'b0000; //Covers 16 combinations of adding 2 graycode numbers
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0001; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0010; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0011; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0000; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0000; in_B = 4'b0001;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0000; in_B = 4'b0010;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0000; in_B = 4'b0011;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0000; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0001; in_B = 4'b0100;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0010; in_B = 4'b1000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0011; in_B = 4'b1100;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b0100; in_B = 4'b0000;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b1000; in_B = 4'b0001;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b1100; in_B = 4'b0010;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #5 in_A = 4'b1000; in_B = 4'b0011;
        $display(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B);
        #10 $finish;
    end

    // initial begin
    //     $monitor(,$time," result=%b, in_A=%b, in_B=%b",result,in_A,in_B); //This monitor statement
prints anytime these values change
    //     #1 in_A = 4'b0000; in_B = 4'b0000; //Covers 16 combinations of adding 2 graycode numbers
```

```
    //      #1 in_A = 4'b0001; in_B = 4'b0000;
    //      #1 in_A = 4'b0010; in_B = 4'b0000;
    //      #1 in_A = 4'b0011; in_B = 4'b0000;
    //      #1 in_A = 4'b0000; in_B = 4'b0000;
    //      #1 in_A = 4'b0000; in_B = 4'b0001;
    //      #1 in_A = 4'b0000; in_B = 4'b0010;
    //      #1 in_A = 4'b0000; in_B = 4'b0011;
    //      #1 in_A = 4'b0000; in_B = 4'b0000;
    //      #1 in_A = 4'b0001; in_B = 4'b0100;
    //      #1 in_A = 4'b0010; in_B = 4'b1000;
    //      #1 in_A = 4'b0011; in_B = 4'b1100;
    //      #1 in_A = 4'b0100; in_B = 4'b0000;
    //      #1 in_A = 4'b1000; in_B = 4'b0001;
    //      #1 in_A = 4'b1100; in_B = 4'b0010;
    //      #1 in_A = 4'b1000; in_B = 4'b0011;
    //      #10 $finish;
    // end

endmodule


// module top();
//     parameter N = 4;

//     logic [N-1:0] in_A, in_B;
//     logic [N:0] result;

//     gray_adder #(N) ga ();

// endmodule
```

RTL Simulation Testbench Results

```
VSIM 1> run -all
#                     5 result=xxxxx, in_A=0000, in_B=0000
#                    10 result=00000, in_A=0001, in_B=0000
#                    15 result=00001, in_A=0010, in_B=0000
#                    20 result=00010, in_A=0011, in_B=0000
#                    25 result=00011, in_A=0000, in_B=0000
#                    30 result=00000, in_A=0000, in_B=0001
#                    35 result=00001, in_A=0000, in_B=0010
#                    40 result=00010, in_A=0000, in_B=0011
#                    45 result=00011, in_A=0000, in_B=0000
#                    50 result=00000, in_A=0001, in_B=0100
#                    55 result=01100, in_A=0010, in_B=1000
#                    60 result=11011, in_A=0011, in_B=1100
#                    65 result=01111, in_A=0100, in_B=0000
#                    70 result=00100, in_A=1000, in_B=0001
#                    75 result=11000, in_A=1100, in_B=0010
#                    80 result=01110, in_A=1000, in_B=0011
# ** Note: $finish    : prob3_tb.sv(44)
#    Time: 90 ns  Iteration: 0  Instance: /top
# End time: 19:36:22 on Nov 28,2021, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
```

Synthesized Netlist Testbench Results

```
VSIM 1> run -all
#                     5 result=xxxxx, in_A=0000, in_B=0000
#                    10 result=00000, in_A=0001, in_B=0000
#                    15 result=00001, in_A=0010, in_B=0000
#                    20 result=00010, in_A=0011, in_B=0000
#                    25 result=00011, in_A=0000, in_B=0000
#                    30 result=00000, in_A=0000, in_B=0001
#                    35 result=00001, in_A=0000, in_B=0010
#                    40 result=00010, in_A=0000, in_B=0011
#                    45 result=00011, in_A=0000, in_B=0000
#                    50 result=00000, in_A=0001, in_B=0100
#                    55 result=01100, in_A=0010, in_B=1000
#                    60 result=11011, in_A=0011, in_B=1100
#                    65 result=01111, in_A=0100, in_B=0000
#                    70 result=00100, in_A=1000, in_B=0001
#                    75 result=11000, in_A=1100, in_B=0010
#                    80 result=01110, in_A=1000, in_B=0011
# ** Note: $finish    : prob3_tb.sv(44)
#    Time: 90 ns  Iteration: 0  Instance: /top
# End time: 19:35:41 on Nov 28,2021, Elapsed time: 0:00:01
# Errors: 0, Warnings: 1
```

Synthesis Details

Number of Cells	31 (26 combinational, 4 inv/buffers)
Total Cell Area	11088.000000
Power Consumption	16.8567 mW

Problem 4

FSM Code

```
// Proj 2 Prob 1 FSM Boolean Expression Design
module S1_FSM (
    input clk, rst,
    output logic out
);

    logic [3:0] cs, ns;

    always_ff @(posedge clk or posedge rst) begin: seq_logic
        if (rst) begin
            cs <= 4'b0000;
        end else cs <= ns;
    end: seq_logic

    always_comb begin: next_state_logic
        ns[3] = (!cs[3]&!cs[2]) | (!cs[3]&cs[2]&!(cs[1]^cs[0])) | (cs[3]&cs[2]&(cs[1]^cs[0]));
        ns[2] = cs[3];
        ns[1] = cs[2];
        ns[0] = cs[1];
```

```
    end: next_state_logic

    always_comb begin: output_logic
        out = !cs[2] | cs[2] & !(cs[1]^cs[0]);
    end: output_logic

endmodule
```

Synthesis Details

Number of Cells	8 Cells (4 combinational, 4 sequential)
Total Area	7704.0
Power Consumption	1.0420 mW

Testbench Code

```
// Testbench
module top ();
    `timescale 1ns/10ps

    logic clk, rst, out_0;

    initial begin
        clk = 1'b1;
        forever #50 clk = ~clk;
    end

    S1_FSM s1_0 (
        .out(out_0),
        .*);

    initial begin
        rst = 1'b1;
        repeat (1) @(negedge clk);
        rst = 1'b0;
        $display("Bool FSM Q: %04b  Out: %01b", s1_0.cs, out_0);
        repeat (1) @(negedge clk);
        repeat (50) @(negedge clk) begin
            $display("Bool FSM Q: %04b  Out: %01b", s1_0.cs, out_0);
        end
        $finish();

    end

endmodule
```

SV Model Testbench Output

```
VSIM 1> run -all
# Bool FSM Q: 0000  Out: 1
# Bool FSM Q: 0100  Out: 1
```

```
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# ** Note: $finish    : prob4_tb.sv(27)
#    Time: 5150 ns  Iteration: 1  Instance: /top
# End time: 21:58:58 on Nov 18,2021, Elapsed time: 0:00:02
# Errors: 0, Warnings: 0
```

Synthesized Netlist Testbench Output

```
# ** Warning: (vsim-3448) osu05_stdcells.v(361): Setting negative specify check constraint (-190 ps) to zero.
#    Time: 0 ps  Iteration: 0  Instance: /top/s1_0/\cs_reg[0]
VSIM 1> run -all
# Bool FSM Q: 0000  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# Bool FSM Q: 0100  Out: 1
# Bool FSM Q: 1010  Out: 1
# Bool FSM Q: 0101  Out: 0
# Bool FSM Q: 0010  Out: 1
# Bool FSM Q: 1001  Out: 1
# ** Note: $finish    : prob4_tb.sv(27)
```

```
#    Time: 5150 ns  Iteration: 1  Instance: /top
# End time: 22:02:32 on Nov 18,2021, Elapsed time: 0:00:05
# Errors: 0, Warnings: 2
```