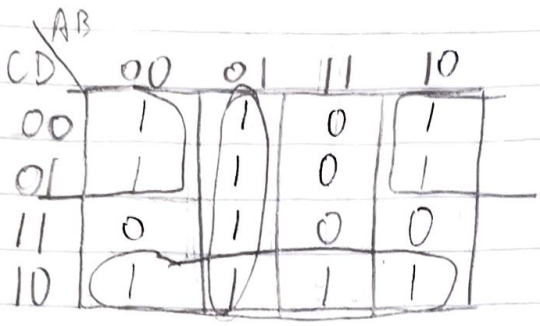
EECS 151 hw 2 Scott Shao 3031883050

Problem 1:

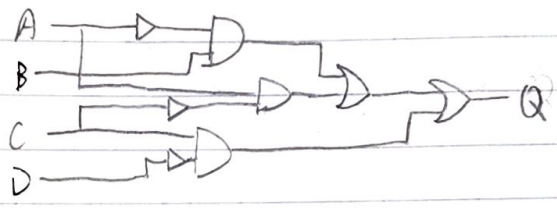
1. $Q = A'B'C'D' + A'B'C'D + A'B'CD' + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D + AB'CD' + ABCD'$

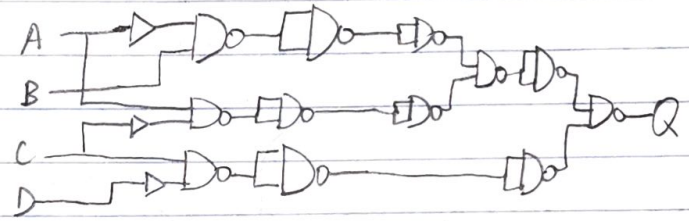
2.



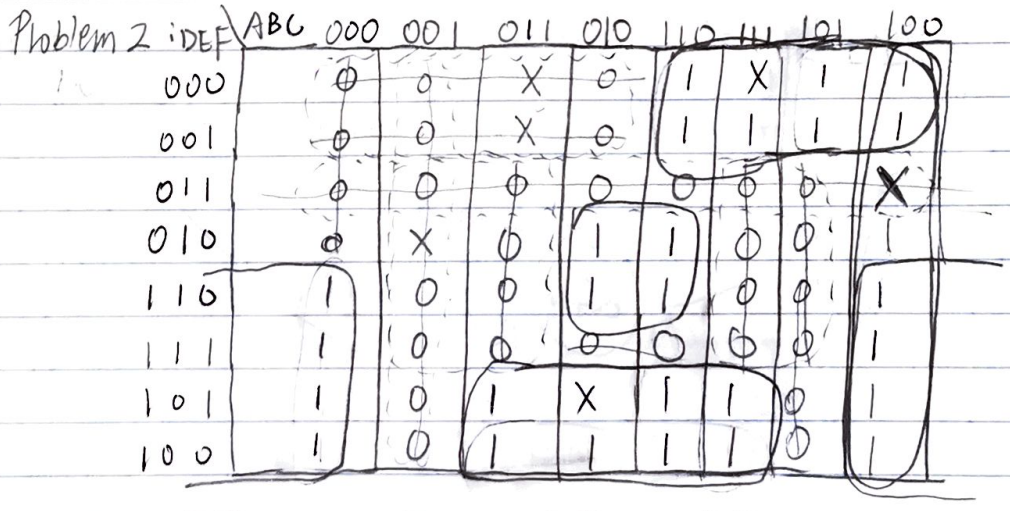
$Q = \bar{A}B + C\bar{D} + A\bar{C}$ or $Q = (\bar{A}+\bar{B}+C)(\bar{A}+\bar{C}+\bar{D})(B+\bar{C}+\bar{D})$

3,





Problem 2:



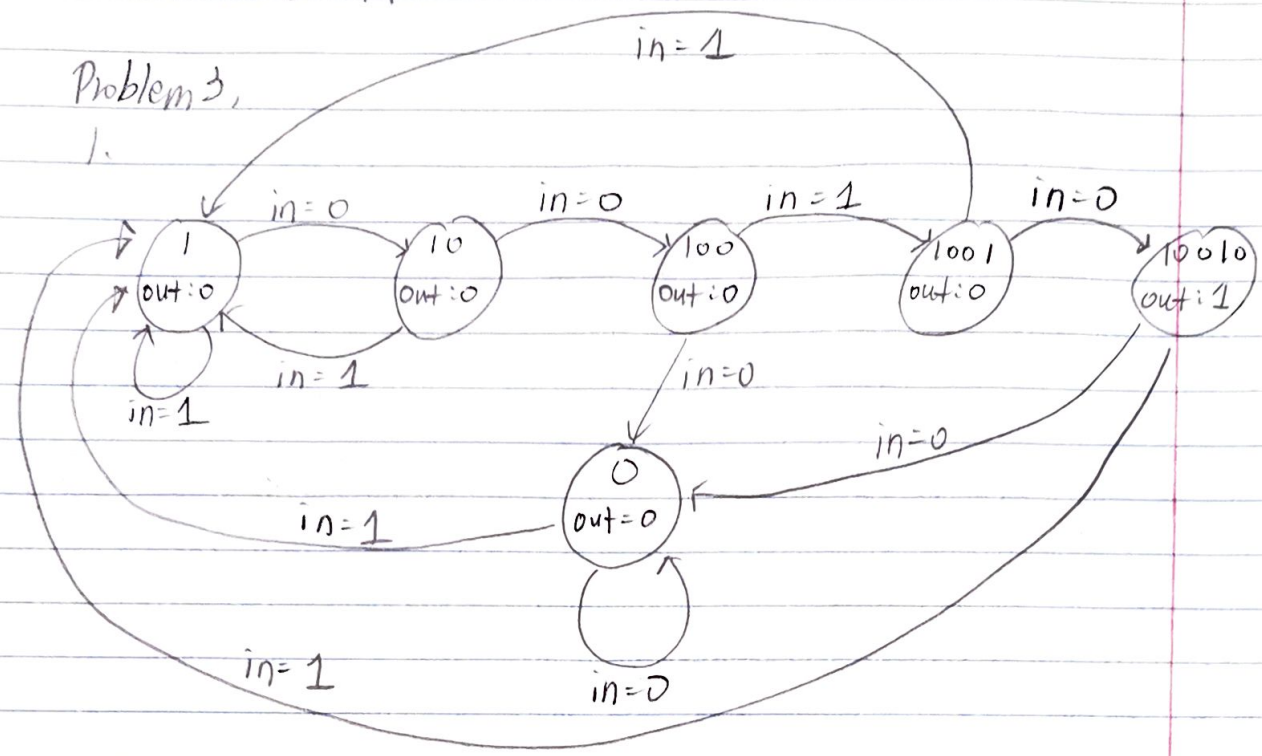
1. $Q = \bar{B}\bar{C}D + BD\bar{E} + A\bar{D}\bar{E} + A\bar{B}\bar{C} + B\bar{C}E\bar{F}$

2. $Q = (\bar{C}+\bar{E})(B+\bar{C}+\bar{D})(\bar{B}+\bar{E}+\bar{F})(A+B+D)(A+\bar{D}+\bar{E})$

3, The two functions are not equivalent.
It is due to to don't care (X) terms. This makes the grouping for SOP and POS not the compliment of each other.

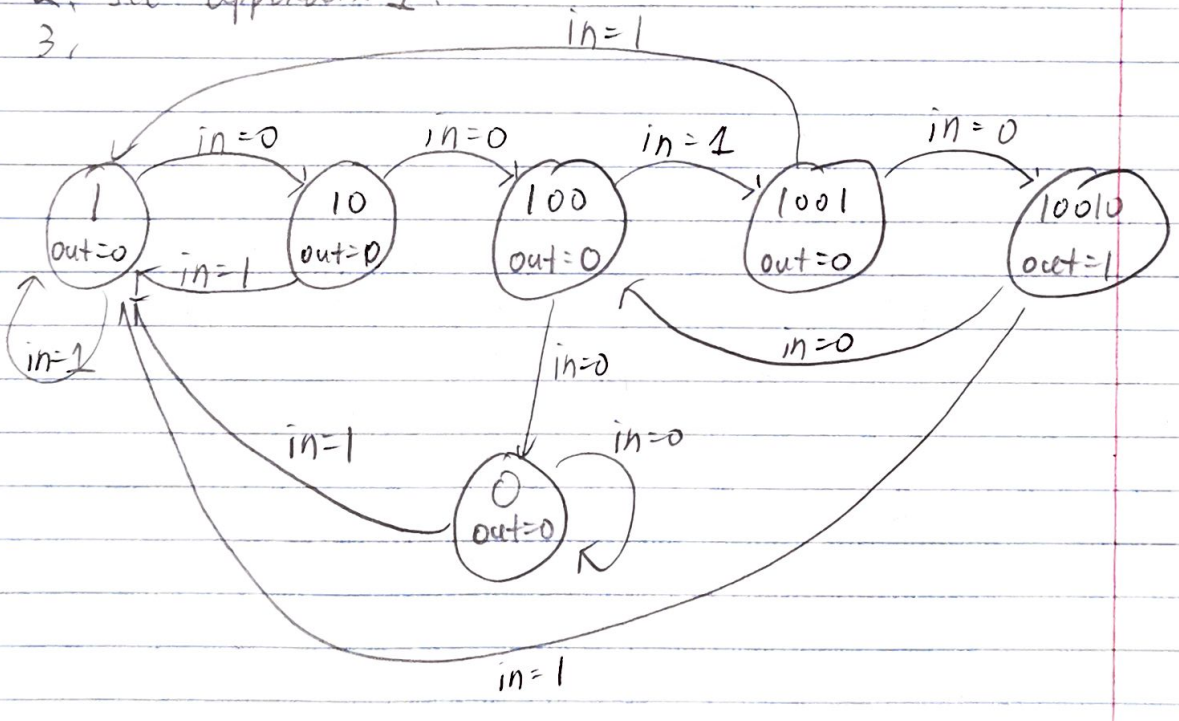
Problem 3,

1.

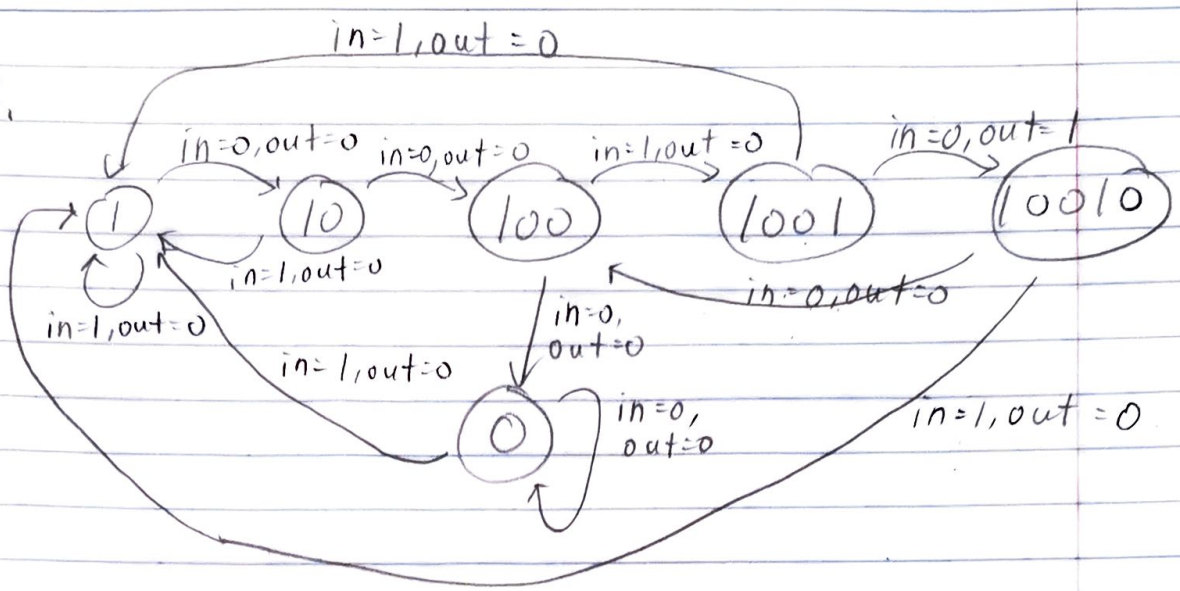


2, See appendix 1.

3,



4.


in=1, out=0
in=0, out=0
in=0, out=0
in=1, out=0
in=0, out=1
1
10
100
1001
10010
in=1, out=0
in=1, out=0
in=0, out=0
in=0, out=0
in=0, out=0
0
in=0, out=0
in=1, out=0

```
module pattern (input clk, input rst, input sig, output out);

localparam  S0 = 3'd0;
localparam  S1 = 3'd1;
localparam  S2 = 3'd2;
localparam  S4 = 3'd3;
localparam  S9 = 3'd4;
localparam  S18 = 3'd5;

reg [2:0] CurrentState;
reg [2:0] NextState;

assign out = (CurrentState == S18);

always @(posedge clk) begin
    if (rst) begin
        CurrentState <= S0;
    end else begin
        CurrentState <= NextState;
    end
end

always @(*) begin
    NextState = CurrentState;
    case (CurrentState)
        S0: begin
            if (sig == 1'b1) NextState = S1;
        end
        S1: begin
            if (sig == 1'b0) NextState = S2;
        end
        S2: begin
            if (sig == 1'b0) NextState = S4;
            else NextState = S1;
        end
        S4: begin
            if (sig == 1'b1) NextState = S9;
            else NextState = S0;
        end
        S9: begin
            if (sig == 1'b0) NextState = S18;
            else NextState = S1;
        end
        S18: begin
            if (sig == 1'b1) NextState = S1;
            else NextState = S0;
        end
    endcase
end
endmodule
```

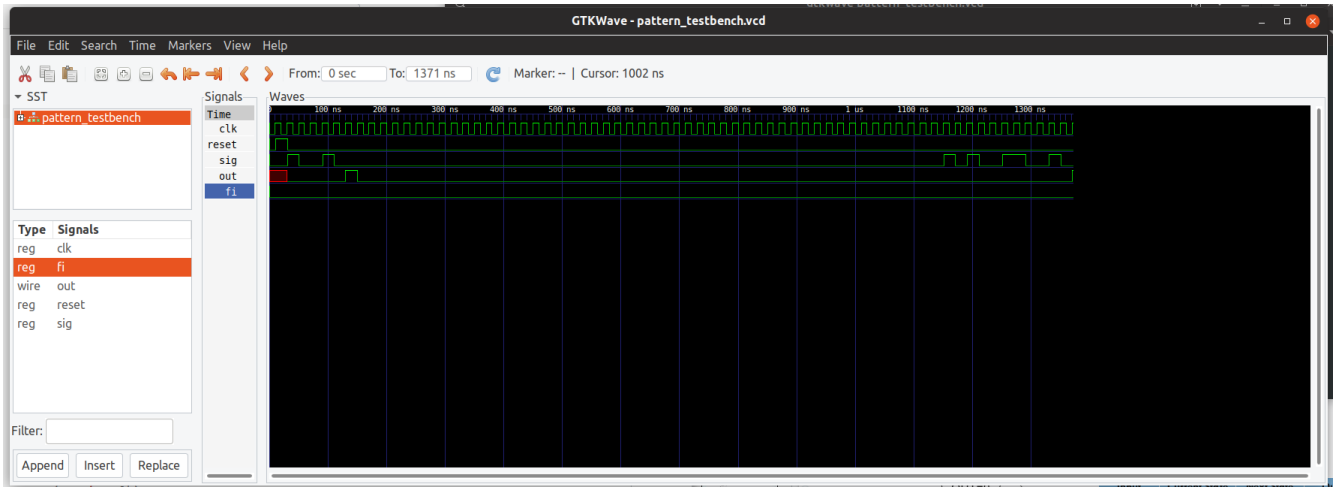
```
`timescale 1ns/1ns

module pattern_testbench();
    reg clk;
    reg reset;
    reg sig;
    wire out;
    reg fi = 1'b0;
    initial clk = 0;
    initial reset = 1'b0;
    initial sig = 1'b0;

    always #(10) clk <= ~clk;

    pattern dut(
        .clk(clk),
        .rst(reset),
        .sig(sig),
        .out(out)
    );

    initial begin
        $dumpfile("pattern_testbench.vcd");
        $dumpvars(0, pattern_testbench);

        @(posedge clk); #1;
        reset = 1;
        @(posedge clk); #1;
        reset = 0;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        repeat (2) @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        if (out == 1'b0) fi = 1'b1;
        repeat (50) @(posedge clk); #1;
        if (out == 1'b1) fi = 1'b1;
        @(posedge clk) #1;
        sig = 1;
        @(posedge clk) #1;
        sig = 0;
        @(posedge clk) #1;
        sig = 1;
        @(posedge clk) #1;
        sig = 0;
        @(posedge clk) #1;
        @(posedge clk) #1;
        sig = 1;
        @(posedge clk) #1;
        @(posedge clk) #1;
        sig = 0;
        @(posedge clk) #1;
        @(posedge clk) #1;
        sig = 1;
        @(posedge clk) #1;
        sig = 0;
        @(posedge clk) #1;
        if (out == 1'b0) fi = 1'b1;
        $display("Out: %0d", out);
```

```
        if (fi) $display("Fail");
        else $display("Pass");
        $finish();
    end
endmodule
```

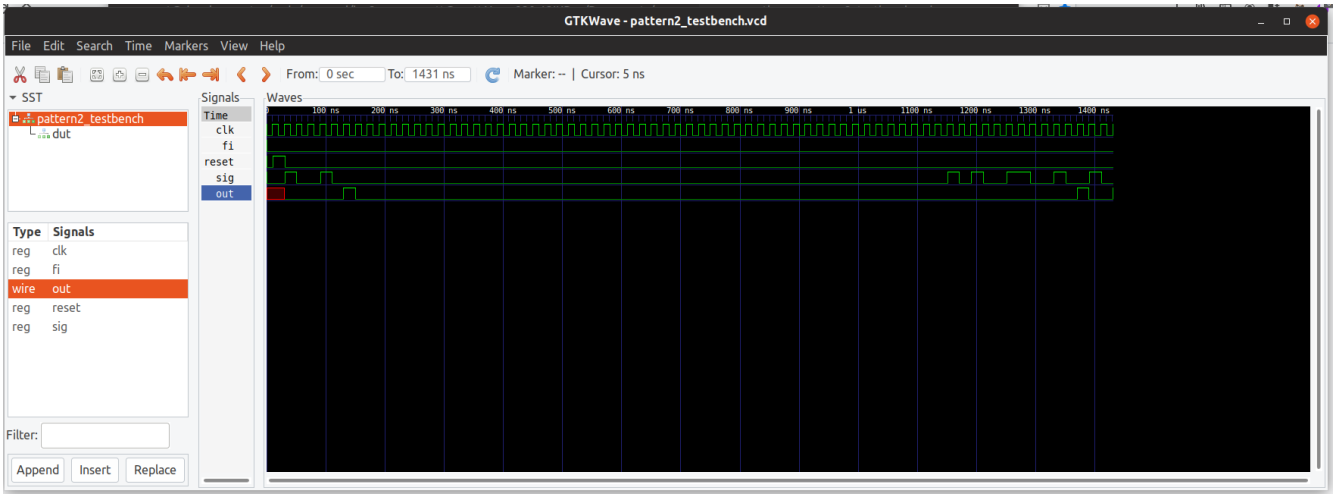
```
module pattern (input clk, input rst, input sig, output out);

localparam  S0 = 3'd0;
localparam  S1 = 3'd1;
localparam  S2 = 3'd2;
localparam  S4 = 3'd3;
localparam  S9 = 3'd4;
localparam  S18 = 3'd5;

reg [2:0] CurrentState;
reg [2:0] NextState;

assign out = (CurrentState == S18);

always @(posedge clk) begin
    if (rst) begin
        CurrentState <= S0;
    end else begin
        CurrentState <= NextState;
    end
end

always @(*) begin
    NextState = CurrentState;
    case (CurrentState)
        S0: begin
            if (sig == 1'b1) NextState = S1;
        end
        S1: begin
            if (sig == 1'b0) NextState = S2;
        end
        S2: begin
            if (sig == 1'b0) NextState = S4;
            else NextState = S1;
        end
        S4: begin
            if (sig == 1'b1) NextState = S9;
            else NextState = S0;
        end
        S9: begin
            if (sig == 1'b0) NextState = S18;
            else NextState = S1;
        end
        S18: begin
            if (sig == 1'b1) NextState = S1;
            else NextState = S4;
        end
    endcase
end
endmodule
```

```
`timescale 1ns/1ns

module pattern2_testbench();
    reg clk;
    reg reset;
    reg sig;
    wire out;
    reg fi = 1'b0;
    initial clk = 0;
    initial reset = 1'b0;
    initial sig = 1'b0;

    always #(10) clk <= ~clk;

    pattern dut(
        .clk(clk),
        .rst(reset),
        .sig(sig),
        .out(out)
    );

    initial begin
        $dumpfile("pattern2_testbench.vcd");
        $dumpvars(0, pattern2_testbench);

        @(posedge clk); #1;
        reset = 1;
        @(posedge clk); #1;
        reset = 0;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        repeat (2) @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        if (out == 1'b0) fi = 1'b1;
        repeat (50) @(posedge clk); #1;
        if (out == 1'b1) fi = 1'b1;
        @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        @(posedge clk); #1;
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        if (out == 1'b0) fi = 1'b1;
        @(posedge clk); #1;
```

```
        sig = 1;
        @(posedge clk); #1;
        sig = 0;
        @(posedge clk); #1;
        if (out == 1'b0) fi = 1'b1;
        $display("Out: %0d", out);
        if (fi) $display("Fail");
        else $display("Pass");
        $finish();
    end
endmodule
```



Above: Problem 3 Question 2 gtk waveform.



Above: Problem 3 Question 5 gtk waveform.