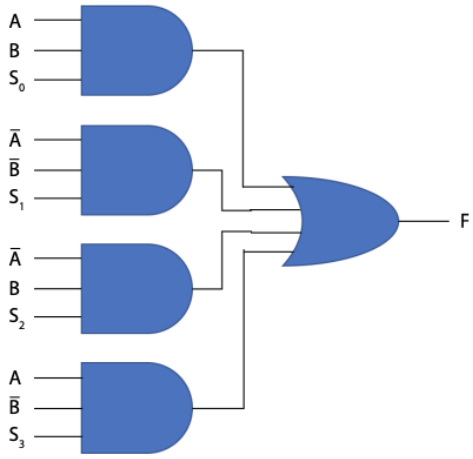
EECS 151/251A Homework 3

Due Monday, September 27th, 2021

Problem 1: Logic [4 pts]

Consider the circuit below. All inputs (A, B, S0, S1, S2, and S3) must be tied to 0 or 1.



What must S0 thru S3 be such that F = XNOR(A, B)?

1) S0 (multiple choice – correct is 1):
   a. 1
   b. 0
2) S1 (multiple choice – correct is 1):
   a. 1
   b. 0
3) S2 (multiple choice – correct is 0):
   a. 1
   b. 0
4) S3 (multiple choice – correct is 0):
   a. 1
   b. 0
5) Can all 2 input logic functions of signals A and B be realized using the above architecture? (multiple choice – correct is YES)
   a. YES
   b. NO, not all

Problem 2: Logic Simplification [4 pts]

Refer to this truth table containing 4 inputs (A, B, C, D) and 1 output (Q):

A	B	C	D	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

1. Write a sum-of-products directly from the truth table (multiple choice – correct is b):

    a (A'+B'+C'+D')(A'+B'+C'+D)(A'+B'+C+D')(A'+B+C'+D')(A'+B+C'+D)(A'+B+C+D')(A+B'+C'+D')(A+B'+C'+D)(A+B+C+D')

    b A'B'C'D' + A'B'C'D + A'B'CD' + A'BC'D' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D + ABCD'

    c (A'+B'+C+D)(A'+B+C+D)(A+B'+C+D')(A+B'+C+D)(A+B+C'+D')(A+B+C'+D)(A+B+C+D)

    d A'B'C'D' + A'B'C'D' + A'B'C'D + A'BC'D + ABC'D' + A'BCD' + A'BC'D' + AB'C'D + ABCD'

2. Use a Karnaugh Map to simplify the logic and write the simplified sum-of-products representation (multiple choice – correct choice c):

    a A'C + C'D' + ABC + A'B'

    b A'B' + C'D' + ABC' + BCD

    c A'C'+ B'C' + A'D' + BCD'

    d A'D + A'C' + AB' + BCD

3. Use a Karnaugh map to simplify the logic and write the simplified product-of-sums representations (multiple choice – correct choice a):

    a (A'+B+C)(C'+D')(A'+B+C')

    b (A + B' + C')(C+D)(A+B'+C)

    c (A + B + C)(C'+D)(A'+B'+C')

    d (A + B)(C+D)(A+B'+C)

4. Using the sum-of-products representation, draw the circuit that implements this function, you may use only 2-input gates. How many AND and OR gates are there (enter it in the format of 1,2 if, for example, there are 1 AND gate and 2 OR gate).

    a Answer: 5,3

5. Transform this circuit such that it is made up only of inverters and NAND gates, how many inverters. And NAND gates are there (enter it in the format of 1,2 if, for example, there are 1 inverter and 2 NAND gates)

    a Answer: 7,19

Problem 3: FSMs - Pattern Detection [6 pts]

In this problem, you are asked to design a pattern detector circuit that aims to extract the pattern "00110" from an input serial bitstream. The circuit receives a new bit every clock cycle from its input "*in*" and has an output "*out*" used to indicate a pattern has been detected. *out* is pulled high for 1 clock cycle to indicate a pattern is detected, and then pulled low to prepare for the next match.

1. Draw the state diagram of this circuit, marking the transition conditions and output values. The detected patterns should not overlap, i.e. it should take at least 5 more cycles after a pattern is detected to detect another pattern. Use a Moore state machine for your implementation. Assume an IDLE state, how many total states (including the IDLE state) are needed to implement this circuit?

    a Answer: 6

2. Write the Verilog, using a case statement to represent the different states, that corresponds to your circuit. Simulate your circuit and show the waveform with two pattern detection events. Submit your code and a screenshot of the waveform.

```
module pattern_detector (
  input wire in,
  input wire clk,
  output wire out);
  reg [2:0] state = 0;
  localparam M0 = 0;
  localparam M1 = 1;
  localparam M2 = 2;
  localparam M3 = 3;
  localparam M4 = 4;
  localparam M5 = 5;
  assign out = (state == M5) ? 1'b1 : 1'b0;
  always @ (posedge clk)
  begin
  case(state)
```

```
M0:
begin
        if(in == 0) state <= M1;
    else if(in == 1) state <= M0;
end
M1:
begin
        if(in == 0) state <= M1;
    else if(in == 1) state <= M2;
end
M2:
begin
        if(in == 1) state <= M1;
    else if(in == 0) state <= M3;
end
M3:
begin
        if(in == 1) state <= M4;
    else if(in == 0) state <= M0;
end
M4:
begin
        if(in == 0) state <= M1;
    else if(in == 1) state <= M5;
end
```

```
        M5:
        begin
                if(in == 0) state <= M1;
            else if(in == 1) state <= M0;
        end
        endcase
        end
    endmodule
```

3. Draw an updated state diagram of your circuit if we now want overlapping patterns to be detected as two patterns, i.e. "001100110" should have two pattern detection events in it. Use a Moore state machine for your implementation. Assume an IDLE state, how many total states (including the IDLE state) are needed to implement this circuit.
    a. Answer: 6
4. Modify your answer to the previous part to implement a Mealy state machine with as few states as possible. How many total states (including the IDLE) are needed to implement this circuit?
    a. Answer: 5
5. Make the necessary modifications to your Verilog and show the updated simulation results showing an overlapping detection event (2 detections separated by less than 5 cycles). Submit your code and a screenshot of the waveform.

```
module pattern_detector (
    input wire in,
    input wire clk,
    output wire out);
    reg [2:0] state = 0;
    localparam M0 = 0;
    localparam M1 = 1;
    localparam M2 = 2;
    localparam M3 = 3;
```

```
localparam M4 = 4;
localparam M5 = 5;
always @ (posedge clk)
begin
case(state)
M0:
begin
        out <= 1'b0;
        if(in == 0) state <= M1;
        else if(in == 1) state <= M0;
end
M1:
begin
        out <= 1'b0;
        if(in == 0) state <= M1;
        else if(in == 1) state <= M2;
end
M2:
begin
        out <= 1'b0;
        if(in == 1) state <= M1;
        else if(in == 0) state <= M3;
end
M3:
begin
        out <= 1'b0;
```

```
                if(in == 1) state <= M4;
                else if(in == 0) state <= M0;
        end
        M4:
        begin
                if(in == 1) state <= M1;
                else if(in == 0)
                begin
                        state <= M2;
                        out <= 1'b1;
                end
        end
    endcase
    end
endmodule
```