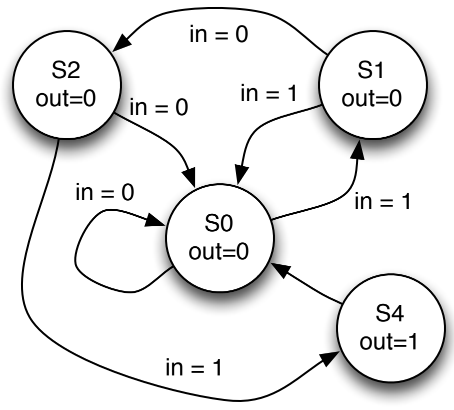
EECS 151/251A Homework 4

Due Wednesday, February 14th, 2018

Problem 1: More Verilog

1. In the space below write out the Verilog code for a module that implements a finite state machine with the behavior of the following state transition diagram:



2. Will this Verilog snippet result in any sequential elements? Why?

```
module m(input [2:0] b, output a);
  always @(a or b) begin
    case (b)
      2'b01: a = 1'b1;
      2'b10: a = 1'b1;
    endcase
  end
endmodule
```

3. What difference is there, if any, between how simulators and synthesizers targeting hardware handle the special value `1'bx`?

4. **EECS 251A Only.** Using flip-flops, simple gates, and multiplexors, draw a diagram for the circuit resulting from synthesizing the following Verilog code. Use individual flip-flops (not N-bit wide registers) and draw out every individual signal path.

```
module foo(ld, X, CNTL, clk, y);
  parameter N = 4;
  input ld, clk;
  input [2:0] CNTL;
  input [N-1:0] X;
  output y;
```

```
    parameter LOAD = 00, SHIFT = 01, COMP = 10;
    reg [N-1:0] S;
    wire [N-1:0] W;
    assign W = S ^ {0, W[N-1:1]};
    assign y = | W;
    always @ (posedge clk)
      case (CNTL)
        LOAD    : S <= X;
        SHIFT   : S <= X >> 1;
        COMP    : S <= W;
        default : S <= X;
      endcase
  endmodule
```

Problem 2: Karnaugh Maps

Take this truth table consisting of 6 input variables and 1 output:

A	B	C	D	E	F	Out	A	B	C	D	E	F	Out
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	1	0	0	0	0	1	0
0	0	0	0	1	0	0	1	0	0	0	1	0	1
0	0	0	0	1	1	0	1	0	0	0	1	1	1
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	1	0	1	0	0	1	0	1	0
0	0	0	1	1	0	0	1	0	0	1	1	0	0
0	0	0	1	1	1	x	1	0	0	1	1	1	1
0	0	1	0	0	0	0	1	0	1	0	0	0	0
0	0	1	0	0	1	x	1	0	1	0	0	1	0
0	0	1	0	1	0	1	1	0	1	0	1	0	1
0	0	1	0	1	1	1	1	0	1	0	1	1	1
0	0	1	1	0	0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	1	1	0	1	1	0	1	0
0	0	1	1	1	0	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	0	1	1	1	1	1
0	1	0	0	0	0	0	1	1	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0	0	0	1	0
0	1	0	0	1	0	1	1	1	0	0	1	0	1
0	1	0	0	1	1	1	1	1	0	0	1	1	1
0	1	0	1	0	0	0	1	1	0	1	0	0	0
0	1	0	1	0	1	0	1	1	0	1	0	1	0
0	1	0	1	1	0	0	1	1	0	1	1	0	0
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	0	0	0	1	1	1	0	0	0	1
0	1	1	0	0	1	x	1	1	1	0	0	1	1
0	1	1	0	1	0	1	1	1	1	0	1	0	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	0	0	0	1	1	1	1	0	0	1
0	1	1	1	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1

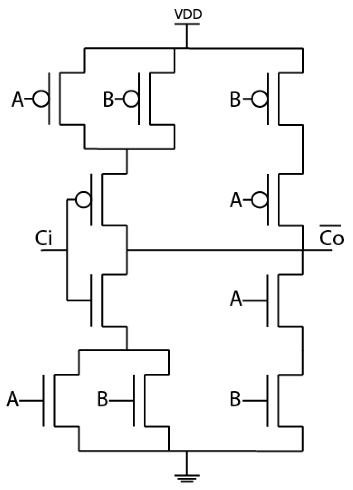
where x means "don't care".

1. Use a 6-variable Karnaugh Map (2D) to derive a simplified boolean function from this truth table using the Sum of Products method.
2. Use a 6-variable Karnaugh Map (2D) to derive a simplified boolean function from this truth table using the Product of Sums method.
3. Are your two answers to this question the same? Why or why not?

Problem 3: Static Complementary CMOS Logic

1. Implement the logic function $\overline{C_o} = \overline{AB + AC_i + BC_i}$ using a complementary pull-up and pull-down network.

2. A friend proposes the following implementation of the function in subquestion (1). Does this perform the same function as the gate from (1)? If so, what advantage does it have over your implementation in (1)? If not, what is the function it implements?



3. Is the gate shown in (2) a static CMOS gate? Explain your answer.

4. **EECS 251A Only.** Design a complex CMOS logic gate that implements the function below. $f(A, B, C, D) = \overline{A \cdot (B \cdot (C + D) + C \cdot D)}$