HW3a:

Question: a) Produce the minimum Sum of Products equation of the following K-Map.

```
  F
 /   \\/
A   X 1 0 0
   \ / 1
   X / 0
   | 0
   \ 1
B X 0 1 1
   X 0
   | 0
   \ 0
```

b) Draw the circuit diagram for your minimum Sum of Products equation above. Notice that you can ONLY USE NAND, NOR, and Inverter gates.
Find the error in the following state machine when the inputs are A, B, and C. The output is C, and there are 4 states.
You are designing a system to control the temperature of a batch of beer as it brews. The system contains a heating element to heat water, and a temperature sensor that tells you if the temperature is too low (LOW_T = 1), or too high (HIGH_T = 1).

If the temperature is too low, the heating element should be turned on and remain on until the temperature gets too high. If the temperature is too high, the heating element should be turned off and remain off until the temperature gets too low. Note that LOW_T and HIGH_T will never both be true at the same time.

Design a circuit that will control the state of the heating element. Create a finite state machine for the circuit, then convert the FSM into a circuit with only NAND gates, using as few as possible.
Design a 4-input gate that outputs TRUE when at least two inputs are true. Using a K-Map, produce the minimum Sum of Products form for this circuit. Then draw the circuit diagram.
\[
((A + \overline{BA})c + \overline{cD}) \cdot B + \overline{DB}
\]

a) Simplify the boolean equation completely.

b) Draw the simplified circuit diagram using only NOT, NAND and NOR gates.
Show the output waveform for the transition \((A=0, B=1, C=0)\) to \((A=1, B=0, C=1)\). All the gates have a 1 ns delay.
Write a complete testbench for the Odd Parity Checker.

module OddParityChecker (clk, reset, w, out);
    input logic clk, reset, w;
    output logic out;

    enum { Odd, Even } ps, ns;

    always_comb begin
        case (ps)
            Odd: if (w) ns = Even;
                else ns = Odd;
            Even: if (w) ns = Odd;
                else ns = Even;
        endcase
    end

    always_ff @(posedge clk) begin
        if (reset)
            ps <= Even;
        else
            ps <= ns;
    end
endmodule
Use a state table and K-Maps to give Boolean expressions for the outputs of the following finite state machine. Then implement the circuit using basic gates and D flip-flops.

(inputs: A, B)
(outputs: C, D, E)
Convert the following circuit to use only NAND gates.