EECS150
Midterm1
10/5/06

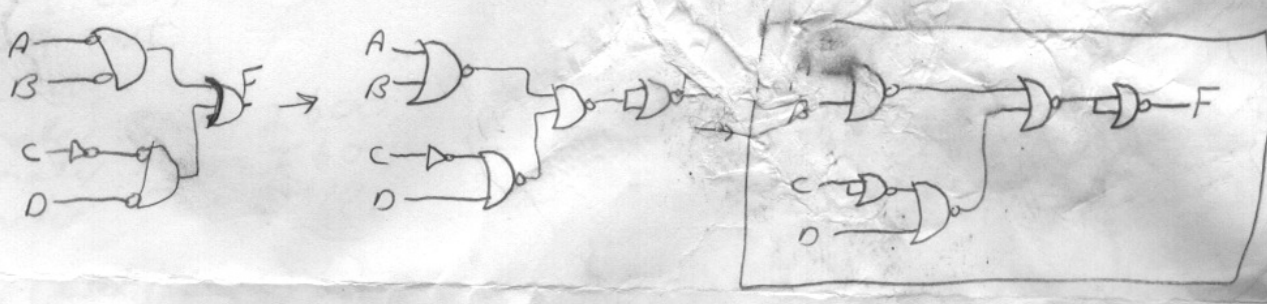
1&2	10	/10
3	20	/20
4 & 5	20	/20
6	30	/30
7	20	/20
Total	100	/100

Note: problems 1 and 2 are a little trickier than last semester! Read carefully!

1) Given **F=A'B'+CD'**
   a. Write **F'** in product of sums notation

$\overline{F} = \overline{\overline{A}\overline{B}+CD} = (\overline{\overline{A}\overline{B}})(\overline{CD}) = (A+B)(\overline{C}+D)$

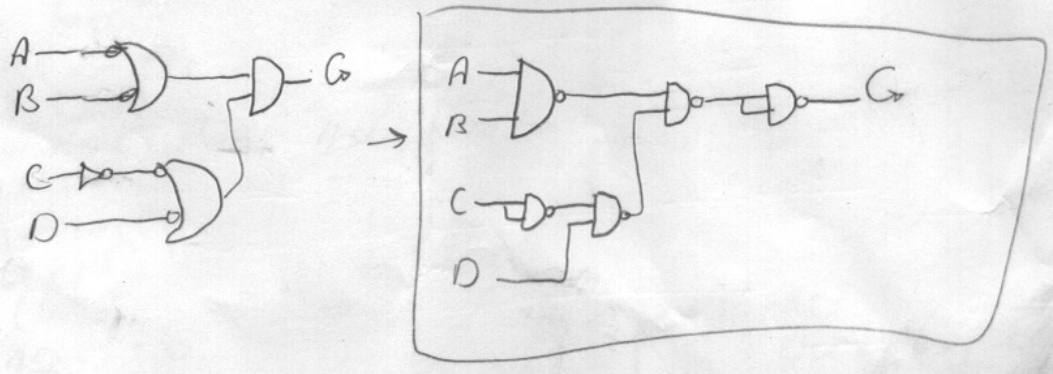
   b. Implement F using as few 2 input NOR gates as possible. Assume that only the true literals (A,B,C,D) are available, not their complements (A', B', C', D').



2) Given **G = (A'+B')(C+D')**
   a. Write **G'** in sum of products notation.

$\overline{G} = \overline{(\overline{A}+\overline{B})(C+\overline{D})} = (\overline{\overline{A}+\overline{B}}) + \overline{(C+\overline{D})} = AB+\overline{C}D$

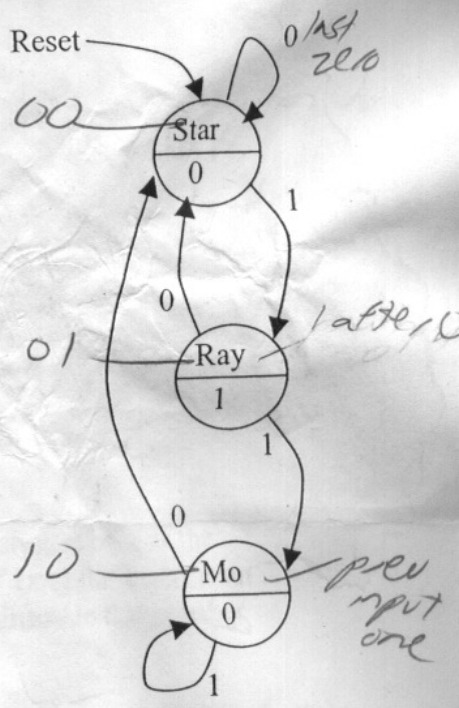
   b. Implement G using as few 2 input NAND gates as possible. Assume that only the true literals are available, not their complements.



3) Answer the following questions for the FSM below:

a. Briefly describe the function of this sequence detector. When is the output 1? The seq. det. outputs a 1 when it receives a 1 as input that was not immediately preceeded by a 1 and a 0 otherwise.

b. Write a Verilog module which would implement this FSM for input variable "In" and output variable "Out." Use the same standard format as was presented in the Lab 3 lecture and used in Lab 3. (Define your states; use one always block for next state and output; use one always block for state transition)

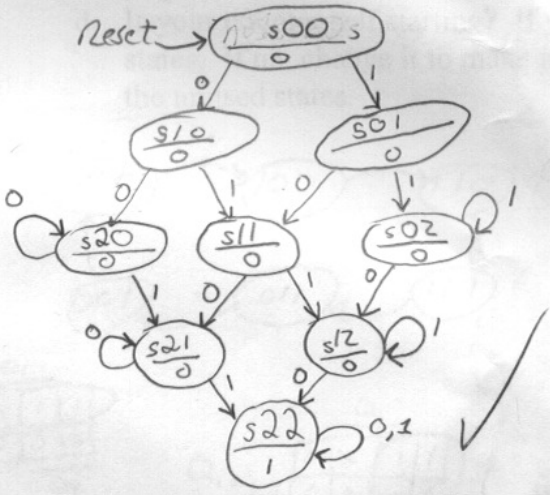



```
module FSM(Reset, Clock, In, Out);
  input Reset, Clock, In;
  output Out;
  reg [1:0] curr;
  reg [1:0] next;
  reg Out;
  parameter Star = 2'b00,
  parameter Ray = 2'b01;
  parameter Mo = 2'b10,

always @(posedge Clock) begin
    if (Reset)
      curr <= Star;
    else
      curr <= next;
end //always

always @(curr or In) begin
    next = curr; //default → stay here
    case (curr)
      Star: begin
        Out = 1'b0;
        if (In)
          next = Ray; // stay here for In
      end //Star
      Ray: begin
        Out = 1'bZ;
        if (In)
          next = Mo;
        else
          next = star;
      end //Ray
      Mo: begin
        Out = 1'b0;
        if (!In)
          next = star; // stay here for In = 1
      end //Mo
      default:
        Out = 1'bx; // stay in null state until reset, output x
    endcase
end //always
endmodule //FSM
```

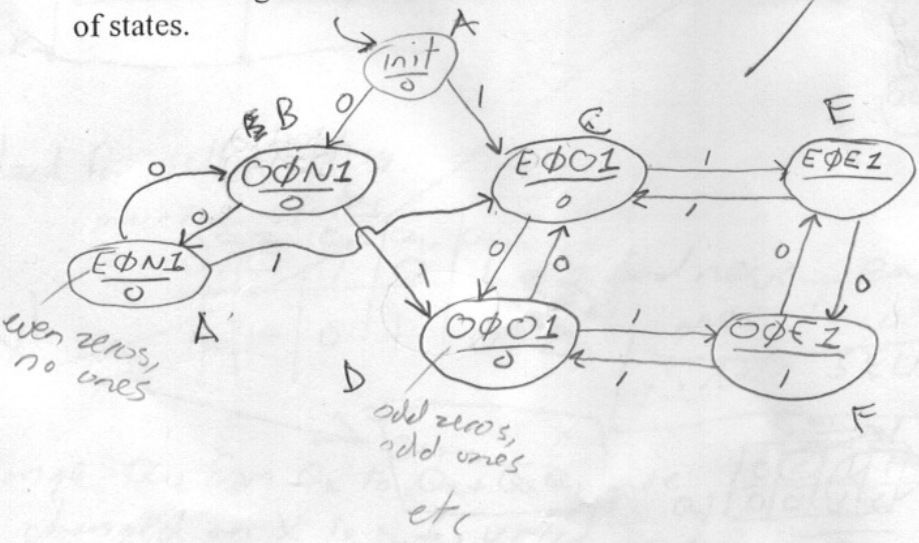
4) A finite state machine has one input and one output. The output becomes 1 and remains 1 thereafter when at least two 0s and at least two 1s have occurred as inputs, in any order after reset. Draw a state diagram of this FSM as a Moore machine. Try to minimize the number of states.



State "sxy" means
x zeroes and
y ones have been seen
(saturating at 2 in each case)

Note: all states transition to s00 upon Reset, overriding in-diagram transitions.

5) A Moore machine has one input and one output. The output should be 1 if the total number of 0s at the input is odd, and the total number of 1s at the input is an even number greater than 0. Draw a state diagram. Try to minimize the number of states.



Assumed no Reset signal and initial state = init. If Reset signal required, Reset causes Transition to init from any state as in previous FSM.

6) Design a 3 FlipFlop counter which transitions through states $Q_2Q_1Q_0$ = 000, 100, 110, 111, 011, 001 and then repeats.

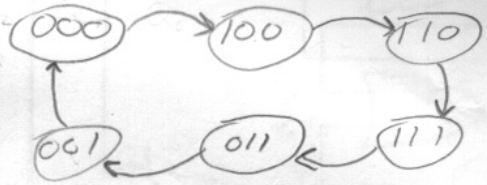
a. Draw the state diagram and state transition table

b. Draw the Karnaugh maps, clearly indicating the implicants that you use in your covers of the next-state functions.

c. Implement the counter using D flip flops and whatever gates you like.

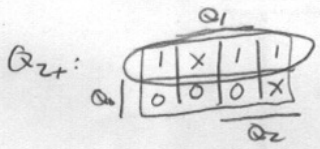
d. Is your counter self starting? If yes, show the transitions of the unused states. If no, change it to make it self starting, and show the transitions of the unused states.

a)

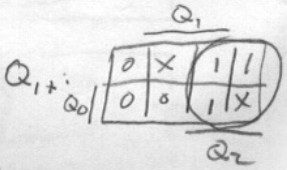


Q_2	Q_1	Q_0	Q_{2+}	Q_{1+}	Q_{0+}
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	X	X	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	X	X	X
1	1	0	1	1	1
1	1	1	0	1	1

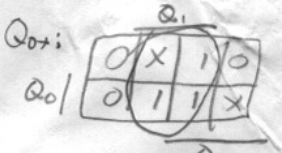
b)



$Q_{2+} = \overline{Q_0}$

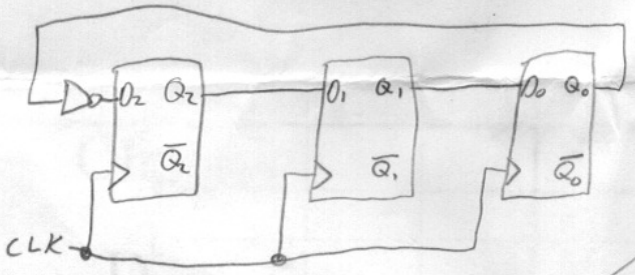


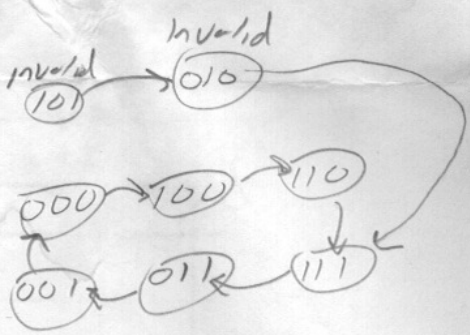
$Q_{1+} = Q_2$



$Q_{0+} = Q_1$

c)





d) check for self starting:

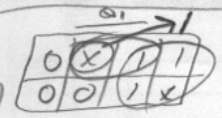
invalid states

Q_2	Q_1	Q_0	Q_{2+}	Q_{1+}	Q_{0+}
0	1	0	1	0	1
1	0	1	0	1	0

bad news — each invalid state goes to other invalid state ∴ NOT SELF STARTING!

the fix!

- To fix, change Q_{1+} from Q_2 to $Q_{1+} = Q_{2+} + \overline{Q_0}Q_1$, ie



- Since changed an X to a 1, valid states unaffected. Retry state transitions for invalid states

Q_2	Q_1	Q_0	Q_{2+}	Q_{1+}	Q_{0+}
0	1	0	1	1	1
1	0	1	0	1	0

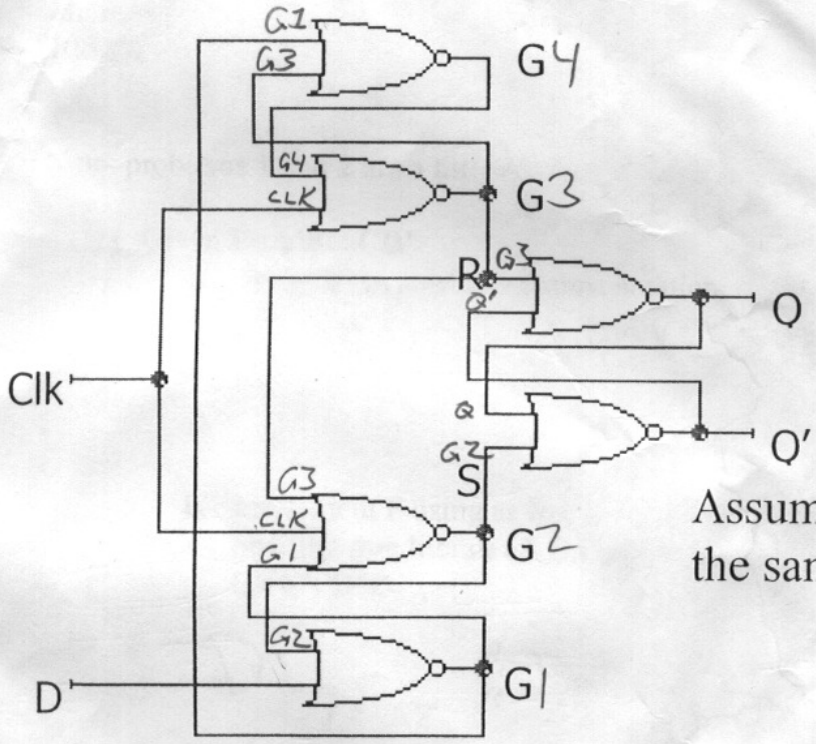
NOW it's self starting —

010 → 111 ✓good
invalid valid

101 → 010 → 111 ✓good
invalid invalid valid

} PROVED TO BE SELF STARTING

7) 20



Assume all gates have exactly the same delay.

1 gate delay