INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (End Semester)	**SEMESTER (Spring)**

Roll Number									**Section**		**Name**	

Subject Number	C	S	2	1	0	0	2	**Subject Name**	*Switching Circuits and Logic Design*

Department / Center of the Student		**Additional sheets**	

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

CS21002 Switching Circuits and Logic Design

End-Semester Test

27–April–2016 2:00–5:00pm Maximum marks: 60

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. A three-input *minority gate* $\mu(a,b,c)$ produces output 1 if and only if at most one of the three inputs is 1. Also, let $M(a,b,c) = \mu(a,b,c)'$, that is, $M(a,b,c)$ is the three-input *majority gate* which outputs 1 if and only if at least two of the three inputs are 1. It is easy to derive that $\mu(a,b,c) = a'b' + b'c' + c'a'$, and $M(a,b,c) = ab + bc + ca$.

Recall that a set of gates is called *functionally complete* if any Boolean function can be realized using those gates and the constants $0,1$. We have seen that $\{$OR,AND,NOT$\}$ and $\{$NAND$\}$ are two examples. Prove or disprove the following two statements. No credits will be given without formal justifications.

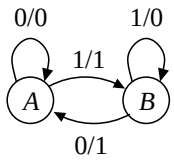
(a) Minority gates (along with the constants 0 and 1) are functionally complete. **(4)**

Solution *True.* We have $\mu(a,b,1) = a'b' = (a+b)'$, that is, a minority gate can be used to build a NOR gate.

(b) Majority gates (along with the constants 0 and 1) are functionally complete. **(6)**

Solution *False.* We have $M(a,b,0) = ab$ and $M(a,b,1) = a+b$. But apparently, we cannot design the complement function using majority gates. In order to *prove* this, we proceed by contradiction. Let N be a network of majority gates, which takes $a,0,1$ as inputs, and outputs a'. We may assume that N is a complement-producing network which uses the minimum possible number of majority gates. The output of N is the output of a majority gate (since the output cannot be directly taken from the input lines $a,0,1$). Call this gate G. When $a = 0$, the output of G is 1, that is, at least two of the three inputs of G are 1. On the other hand, if $a = 1$, the output of G is 0, that is, at most one of the three inputs of G is 1. This proves that there is (at least) one input to G which is 1 when $a = 0$, and 0 when $a = 1$. We can throw away G, and take that input of G as a realization of a'. This contradicts the minimality of the network N.

2. Consider the following Mealy machine M that takes a one-bit input stream and produces a one-bit output stream. Assume that the initial state of M is A.



(a) What is the output of M if the input is 0010110111? **(2)**

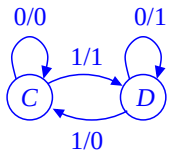
Solution 0011101100

(b) What is the input of M if the output is 0010110111? **(2)**

Solution 0011011010

(c) Design a Mealy machine N that, given the output of M, produces the corresponding input for M. Specify the start state of N. (Treat M as a message-encoding circuit. Then, N is the corresponding message-decoding circuit.) **(4)**

Solution Notice that M changes state and outputs 1 if and only if it sees a change in the input bit. Since A is the start state of M, it is assumed that the (-1)-st bit is 0. At the decoding end, a zero bit in the input indicates no change in the output bit, a one bit in the input indicates a bit flip in the output. Therefore N can be designed as follows. Its start state is C.



(d) Now, suppose that N does not know the start state of M. It is designed assuming that A is the start state. N receives a stream of hundred bits output by M. If the start state of M is B, how many bits will be incorrectly decoded by N? **(2)**

Solution All of the hundred bits.

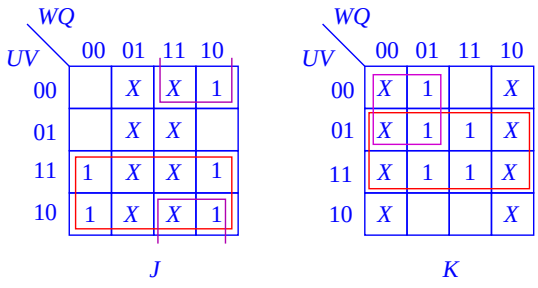
3. Think of a three-input memory device called a UVW flip-flop. If $U = V = 0$, then the bit W is stored in the state (that is, $Q(t+1) = W$). For other combinations of U and V values, the input W does not play any role. If $U = 1$ and $V = 0$, then the state of the flip-flop is set ($Q(t+1) = 1$). If $U = 0$ and $V = 1$, then the state of the flip-flop is reset ($Q(t+1) = 0$). Finally, if $U = V = 1$, then the state of the flip-flop is complemented ($Q(t+1) = Q'(t)$).

(a) Design a UVW flip-flop using only a JK flip-flop and logic gates. **(5)**

Solution We have the following excitation equations for the JK flip-flop.

U	V	W	$Q(t)$	$Q(t+1)$	J	K
0	0	0	0	0	0	X
0	0	0	1	0	X	1
0	0	1	0	1	1	X
0	0	1	1	1	X	0
0	1	0	0	0	0	X
0	1	0	1	0	X	1
0	1	1	0	0	0	X
0	1	1	1	0	X	1
1	0	0	0	1	1	X
1	0	0	1	1	X	0
1	0	1	0	1	1	X
1	0	1	1	1	X	0
1	1	0	0	1	1	X
1	1	0	1	0	X	1
1	1	1	0	1	1	X
1	1	1	1	0	X	1

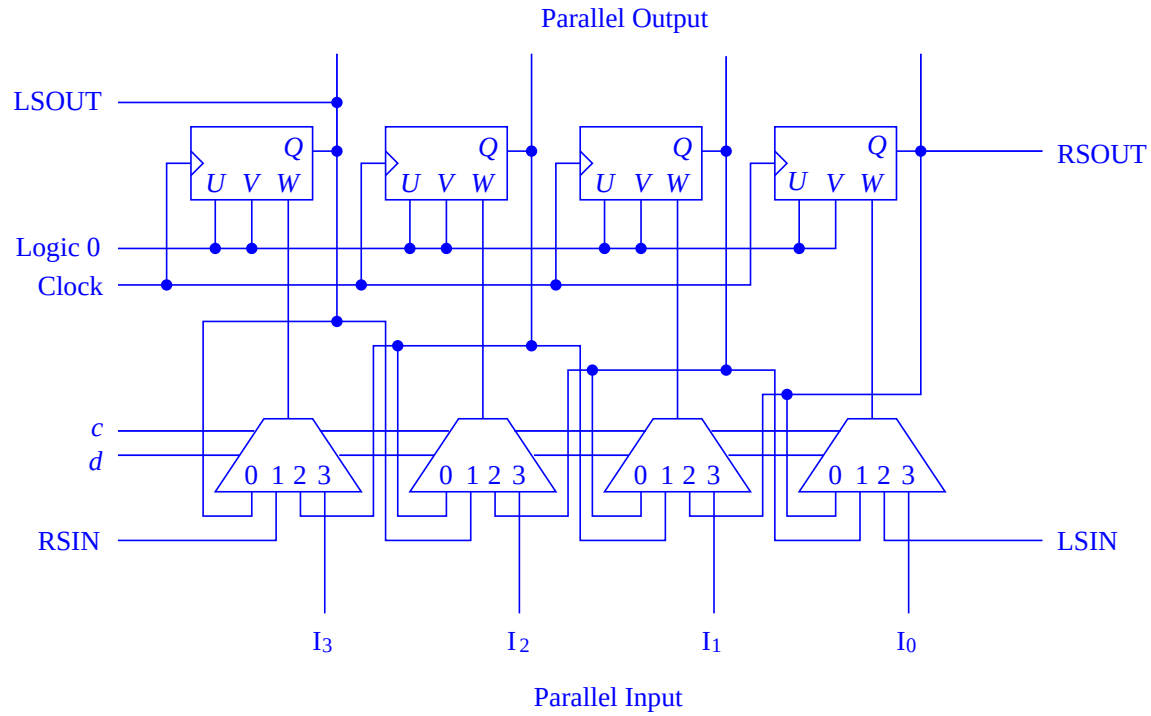
The Karnaugh maps for J and K are as follows:



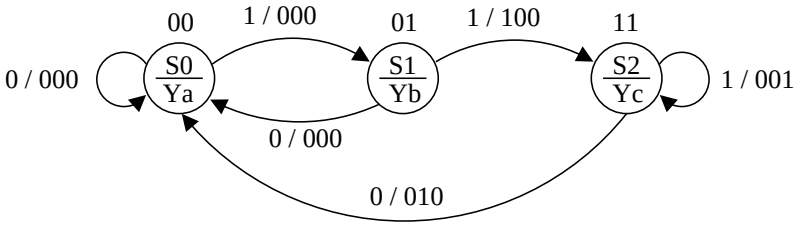
From the map, it follows that $J = U + V'W$ and $K = V + U'W'$.

(b) Design a 4-bit universal shift register using UVW flip-flops. Recall that a universal shift register has two control inputs c and d. If $cd = 00$, the register retains its old value. If $cd = 01$, there is a right shift of the values. If $cd = 10$, there is a left shift of the values. Finally, if $cd = 11$, then the register encounters a parallel load. **(5)**

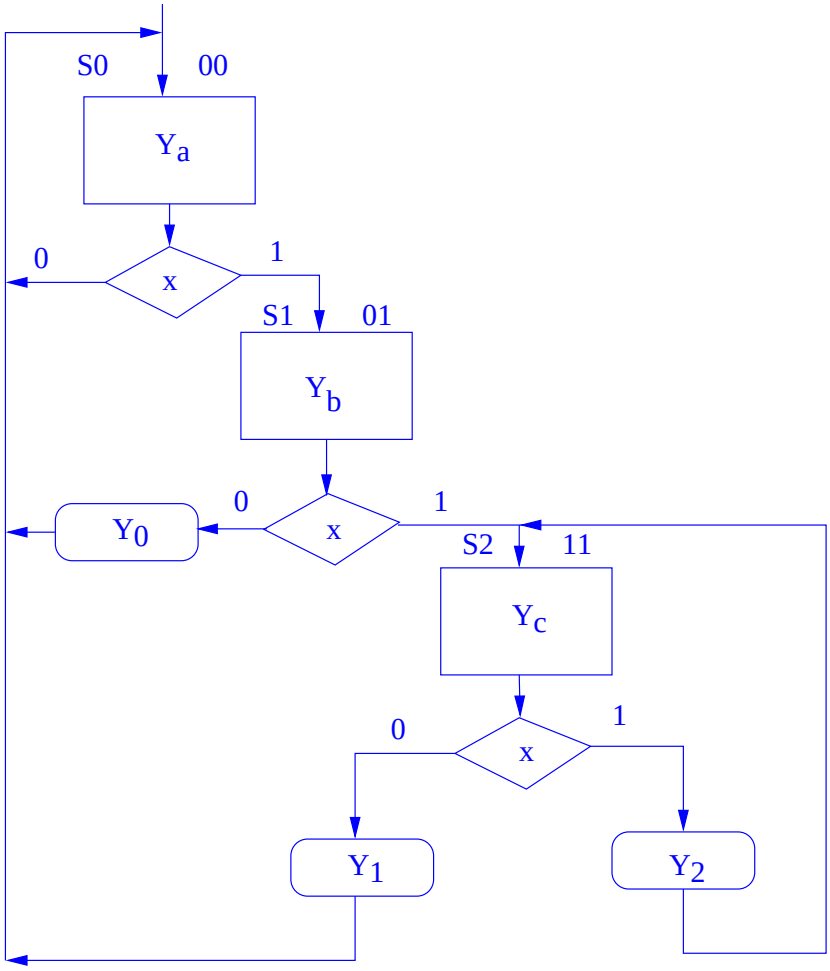
Solution An easy option is to take $U = V = 0$, and W from the output of a 4:1 MUX.



4. (a) Consider the Finite State Machine (FSM) shown in the figure below. The state machine has three states, which are encoded by two bits A, B. The encoding as shown in the figure is 00 for $S0$, 01 for $S1$, and 11 for $S2$. The machine has three Moore outputs Y_a, Y_b, Y_c and three Mealy outputs Y_0, Y_1, Y_2. For each transition, the Mealy outputs are specified in the sequence $Y_0Y_1Y_2$. Draw the Algorithmic State Machine (ASM) chart for the FSM. Clearly label your ASM chart. **(5)**



Solution



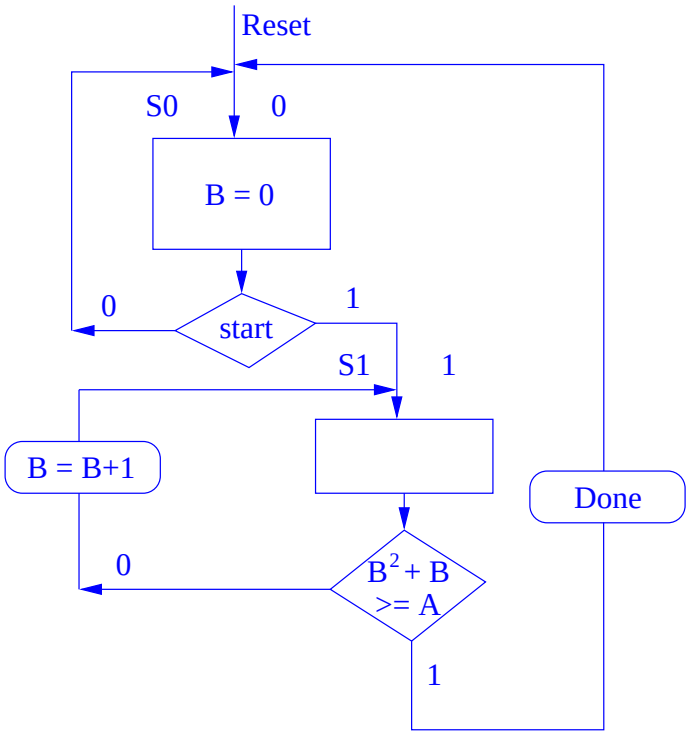
(b) A sequential circuit receives an n-bit input A, and produces an n-bit output B, where both A and B are unsigned numbers. The purpose of the circuit is to calculate the square root of A, to the closest integer, and to store the result in B. For example, if $13 \leqslant A \leqslant 20$, then $B = 4$.

Assume the availability of a squaring circuit $B * B$. Using it, develop an algorithm for computing the result, by incrementing B until $B * B$ gives the correct answer. Subsequently, draw the ASM for your algorithm. In your ASM, the circuit has a reset input, a start input, and a done output to indicate that the result is ready.

Credit will be given for a minimal circuit. (**Hint:** Try to use the identity $(B+1)^2 - B^2 = 2B+1$.) **(5)**

Solution The design is based on the following algorithm:

while $(B^2 + B < A)$ $B = B+1$;



5. (a) Consider the Boolean equation $F = [(a+bc)d + (b' + ac')d']a'b$. Analyze algebraically the circuit for static single input change (SIC) hazards. (**Hint:** If you target a variable for a transition, assign suitable values on the other variables to expose the hazards.) **(5)**

Solution Consider, $F = [(a+bc)d + (b' + ac')d']a'b$. All variables have uncomplemented and complemented literals, hence we need to consider all.

Case 1: a is the target variable which undergoes transition.

Note that $b = 1$, as otherwise $F = 0$. Also note that $c = 0$, as otherwise F does not depend on a, which is the target variable. Thus, $F = [ad + ad']a'$. Now, putting $d = 0$, we get $F = aa'$, which exposes a static-0 hazard. Similarly, $d = 1$ also exposes a static-0 hazard.

Case 2: b is the target variable which undergoes transition.

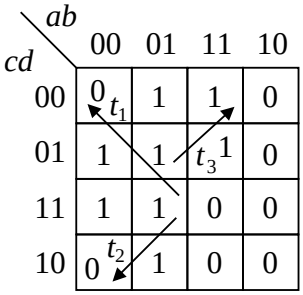
$a' = 1$, as otherwise $F = 0$. Thus, $F = [bcd + b'd']b$. In order to ensure both b and b' exist in the function, $d = 0$. Thus, $F = b'b$ and this gives a static-0 hazard independent of c.

Case 3: c or d is the target variable which undergoes transition at a time (there are two cases clubbed into one).

$a = 0$, and $b = 1 \Rightarrow F = cd$, in both cases and hence there is no hazard.

(b) The K-map in the adjacent figure shows three transitions with start and end points as:

t_1 : $(a'bcd \rightarrow a'b'c'd')$
t_2 : $(a'bcd \rightarrow a'b'cd')$
t_3 : $(a'bc'd \rightarrow abc'd')$



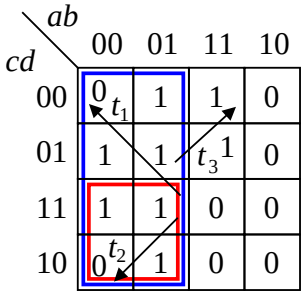
Answer the following questions in the context of hazards in the Sum-of-Products (SOP) form of the Boolean covers for the K-map.

(i) Classify the three transitions into static and dynamic multiple input change (MIC) hazards. **(1)**

Solution Transitions t_1 and t_2 are dynamic hazards, whereas t_3 is a static hazard.

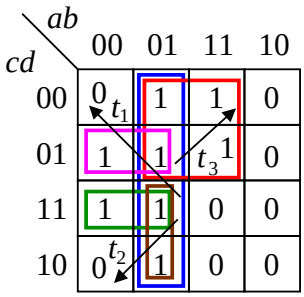
(ii) Identify the privileged cubes for the transitions, whenever applicable. **(1)**

Solution



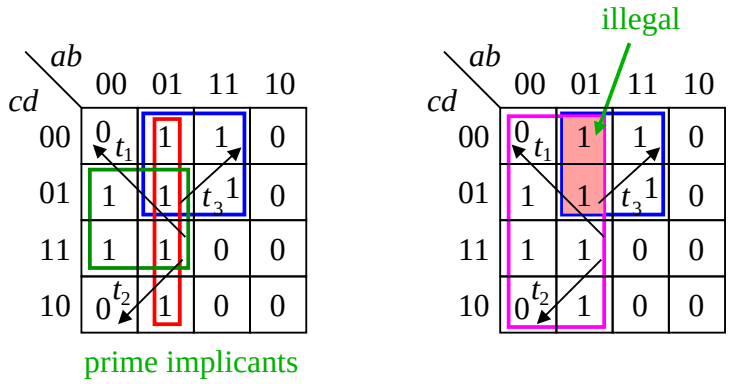
(iii) Identify the required cubes for the transitions, whenever applicable. **(1)**

Solution

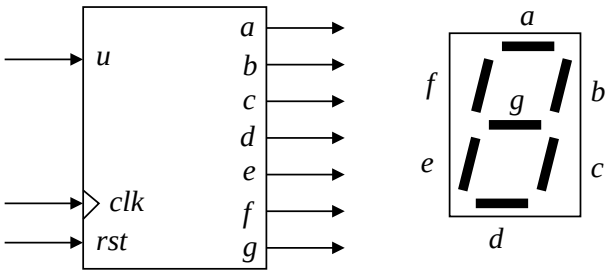


(iv) Prove or disprove that the K-map has a Boolean cover which is dynamic hazard free for the specified three MIC transitions. **(2)**

Solution False, as the following figure demonstrates.



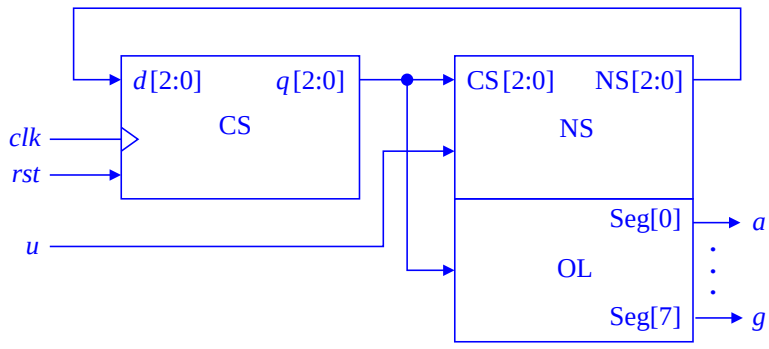
6. Consider a sequential **modulo 6** up-down counter with a seven-segment LED (light emitting diode) connected as shown in the following figure. The state of the counter changes on the positive edge of the clock *clk*, and the design has an asynchronous reset *rst*. If the input *u* is 1, the counter value will change in the sequence $0, 1, 2, 3, 4, 5, 0, 1, 2, \ldots$. If *u* is 0, the counter value will change in the sequence $0, 5, 4, 3, 2, 1, 0, 5, 4, \ldots$. There are seven output signals, *a*–*g*, with each connecting to a segment on LED. A segment of the LED will be on if its control signal is 0 (active low).



In the context of this design, answer the following parts.

(a) Draw the block diagram of the counter with a seven-segment LED, clearly showing the three sub-blocks, namely Current State (CS), Output Logic (OL), Next State Logic (NS), as discussed in the class. As a reminder, the CS block assigns the next state as the current state at the positive edge of the clock, and resets the state whenever the reset goes high. The other blocks are self explanatory from their names. The block diagram should also clearly show the inputs, outputs, internal ports, bus widths which are used in the connections. **(2)**

Solution



(b) Write synthesizable Verilog codes with separate modules for the blocks CS, NS, and OL. **(6)**

```
module CS(d, clk, rst, q);
     input [2:0] d;
     input clk, rst;
     output [2:0] q;
     reg [2:0] q;
     always @(posedge clk or posedge rst) begin
       if(rst) q <= 3'b0;
       else q <= d;
     end
endmodule


module NS(cs, ns, u);
     input [2:0] cs;
     input u;
     output [2:0] ns;

     always@(cs or u) begin
        if(u)
           if(u!=3'b5)
              ns=cs+1;
           else
              ns=3'b0;
        else
           if(u!=3'b0)
              ns=cs-1;
           else
              ns=3'b5;
     end
 endmodule


 module OL(cs,seg7);
      input [2:0] cs;
      output [6:0] seg7;
      reg [6:0] seg7;

      always@(cs) begin
      casex(cs)
         3'b0 : seg7=7'b0000001;
         3'b1 : seg7=7'b1001111;
         3'b2 : seg7=7'b0010010;
         3'b3 : seg7=7'b0000110;
         3'b4 : seg7=7'b1001100;
         3'b5 : seg7=7'b0100100;
         default: seg7=7'bx;
      endcase
      end
   endmodule
```

(c) Write the Verilog code of the top-level circuit, properly instantiating the above three modules. **(2)**

```
module updownctr(clk, rst seg7);
     input clk, rst;
     output [6:0] seg7;
     wire [2:0] w1, w2;

     CS Blk1(.d(w1),.q(w2),.clk(clk),.rst(rst));
     NS Blk2(.cs(w2),.ns(w1),.u(u));
     OL Blk3(.cs(w2),.seg7(seg7));
endmodule
```