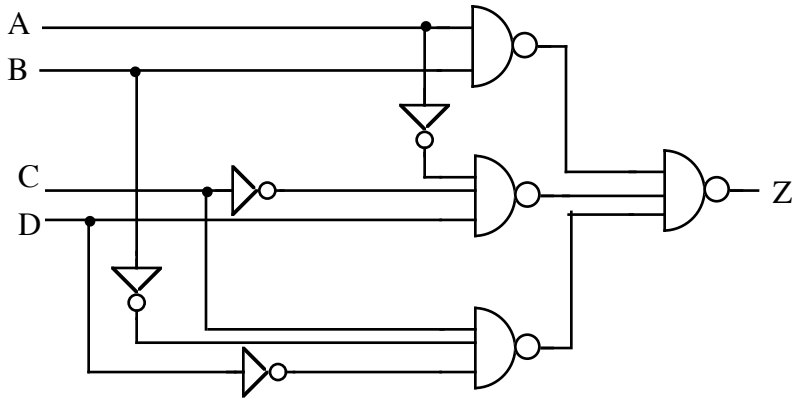
Computer Science W4825

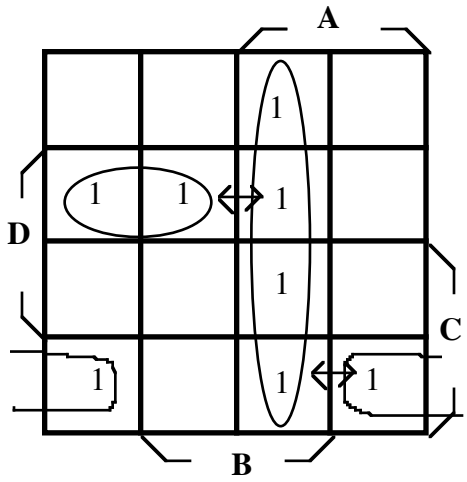
MIDTERM EXAMINATION Solutions March 4, 2004

1.

(a) A combinational logic circuit is shown below. Specify 2 different input variables such that changes in those variables can cause transient false output signals. Indicate the initial state for each change and describe the malfunction.



$Z=AB+\bar{A}\bar{C}D+\bar{B}C\bar{D}$



Mapping the expression on a K-Map, with the product terms (p-terms) corresponding to the circuit ovalled, we see that, in two places, marked on the map with double arrows, changing one variable causes one p-term to go on and another to go off. If the one that goes off changes first, then there will be an interval during which both are off and Z glitches to 0. These are where the static combinational 1-hazards are. They occur between states 0101-1101 and between states 1110-1010. The glitches could occur for changes in either direction. So we could say hazards exist for 0101 or 1101 with A changing, or 1110 or 1010 with B changing.

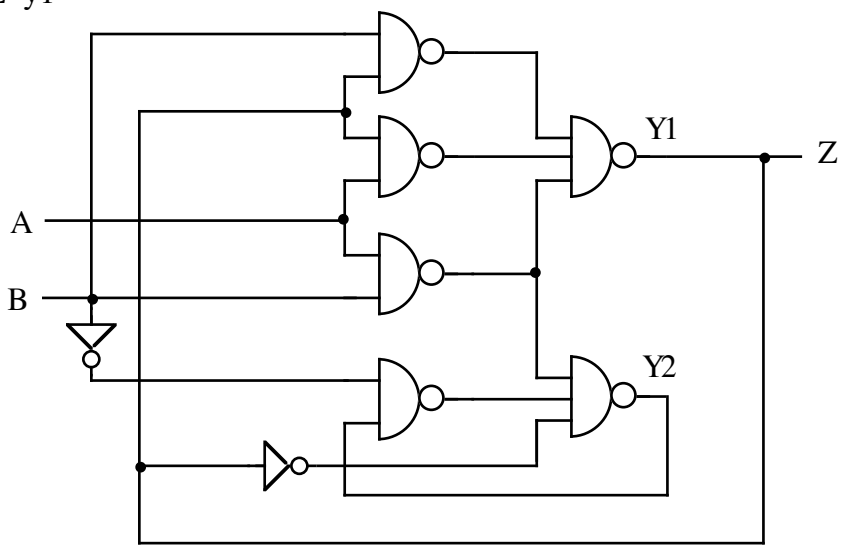
(b) A flow matrix is shown, along with a circuit realizing the function described. Assume SIC (single-input-change) operation. Specify an input-change (give the initial total state and the input variable that changes) that might the system go to the wrong state? Explain. Show where in the circuit a sufficiently large delay would cause this to happen. Where in the circuit would a sufficiently large delay prevent this malfunction?

	AB 00	01	11	10	y1	y2
1	1,0	1,0	3,0	1,0	0	0
2	2,0	1,0	3,0	2,0	0	1
3	2,1	3,1	3,1	3,1	1	1

Y1=AB+Ay1+By1

Y2=AB+$\bar{B}$y2+y1

Z=y1



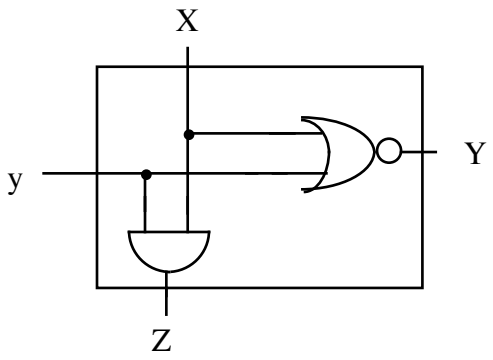
Inspecting the flow table we see that, starting in state 3-01, if we change B once, the specified destination state is 2. If, from the same starting point we changed B 3 times, the specified destination state is 1. So this constitutes an essential hazard. What is supposed to happen for this input change is that y1 goes from 1 to 0 (changing the internal state from 3 to 2.) The hazard will be manifested if the news of the y1-change reaches Y2 BEFORE it sees the B-change. So Y2 sees the system in 2-01, where it is supposed to change to 0. If it does change then the system may end up in state 1-00 instead of in 2-00. We can force this to happen if we put a sufficiently large delay in the branch with the inverter fed by B. We could ensure that the hazard is NOT manifested by putting a sufficiently large delay in the branch containing the Y1-output. Another good place would be in the branch with the lower inverter, which is on the path from Y1 and Y2.

2. Find a minimal-row cover of the table shown below:

	X = 0	X = 1
1	3,-	-,-
2	4,-	1,-
3	2,0	4,-
4	2,1	4,-

		X = 0	X = 1
(14)	1	2,1	1/3,-
(23)	2	3,0	1,-
(24)	3	3,1	1,-

3. NASA's Martian Rover discovered a mysterious device and transmitted back a logic circuit diagram that described it. The device appears to be a linear iterative circuit, in which information flows from left to right. The diagram specifying the logic for the basic module is shown below. Using the symbols on the diagram in the usual manner, reverse engineer this device by deriving the flow table for the function implemented. In one sentence, describe the set of input sequences that produce 1-outputs. (Assume that y=0 at the left end.)



$Y=\overline{\overline{X}+y}=\overline{\overline{x}}\,\overline{y}$, $Z=xy$
The Y-matrix is thus

	X = 0	X = 1	y
1	1	0	0
2	0	0	1

This generates the flow matrix:

	X = 0	X = 1	y
1	2,0	1,0	0
2	1,0	1,1	1

We can now see that, starting in state-1, if we apply an arbitrarily long string of 1's, the output is a corresponding string of 0's, and the system remains in state-1. Applying a 0 moves the state to 2, with the output remaining 0. Repeated 0's bounce the system back and forth between states 1 and 2, with 0's being generated. If a 1 occurs after an ODD number of 0's since the last 1, then, and only then, does a 1-output occur. So we can say that this circuit responds with a 1 when a 1-input terminates an odd string of 0-inputs.

4. Specify a flow table for the iterative function that does subtraction. That is, if inputs A and B represent n-bit binary numbers considered to be unsigned integers, with A≥B, the output S is to be the binary representation of A-B. (The flow table for the binary subtracter is no more complex than that for the binary adder.)

Look at a subtraction problem:

10111000011

01110001010

01000111001

Assume we are looking for A-B. If Ai=Bi, and there is no borrow in, then there will be no borrow out and the difference bit will be 0. If Ai=Bi and there IS a borrow in, then the difference bit will be 1 and there will be a borrow out. If Ai > Bi (Ai=1 and Bi=0) then there will never be a borrow out, even if there is a borrow in, and the difference bit will be the complement of the borrow in. If Ai<Bi, then there will always be a borrow out, and the difference bit will be the complement of the borrow in. This is all reflected in the flow table below, where state 1 represents no borrow in and state 2 represents borrow in.

		AB 00	01	11	10
no borrow	1	1,0	2,1	1,0	1,1
borrow	2	2,1	2,0	2,1	1,0

5.
(a) For the flow table below, find the set of all state mappings and generate the multiplication table for them. Label the mappings for the 0 and 1 columns respectively, M0 and M1.

Start by multiplying M0 by M0. In this case we get M2. The do M0xM1, getting M1, etc., until no new mappings are generated. Build the multiplicaton table as you do this.

	X 0	1
1	2,0	2,0
2	1,1	2,0
3	3,0	2,1

M0	M1	M2	M3
2	2	1	1
1	2	2	1
3	2	3	1

	right M0	M1	M2	M3
M0	M2	M1	M0	M3
left M1	M3	M1	M1	M3
M2	M0	M1	M2	M3

(b) Suppose we have the following complete set of mappings, coding for them, and multiplication table for them. Generate SOP expressions for the interior cell logic that specifies the mapping for the input sequence that it spans.

M0	M1	M2
1	1	2
2	1	2
3	1	2

mapping	w1	w2
M0	0	0
M1	1	0
M2	0	1

	right M0	M1	M2
M0	M0	M1	M2
left M1	M1	M1	M2
M2	M2	M1	M2

In the multiplication table, replace each Mi entry with its w-code to obtain the W-matrix below.

		wR1wR2		
		00	10	01
	00	00	10	01
wL1wL2	10	10	10	01
	01	01	10	01

W1W2

Generate the expressions for W1 and W2 from this table (exploiting don't cares.) This yields:

W1=wR1+ wL1$\bar{w}$R2, w2=wR2+wL2$\bar{w}$R1