EE434

ASIC and Digital Systems

Final Exam

May 4, 2017. (8am – 10am)

Instructor: Dae Hyun Kim (daehyun@eecs.wsu.edu)

Name:

WSU ID:

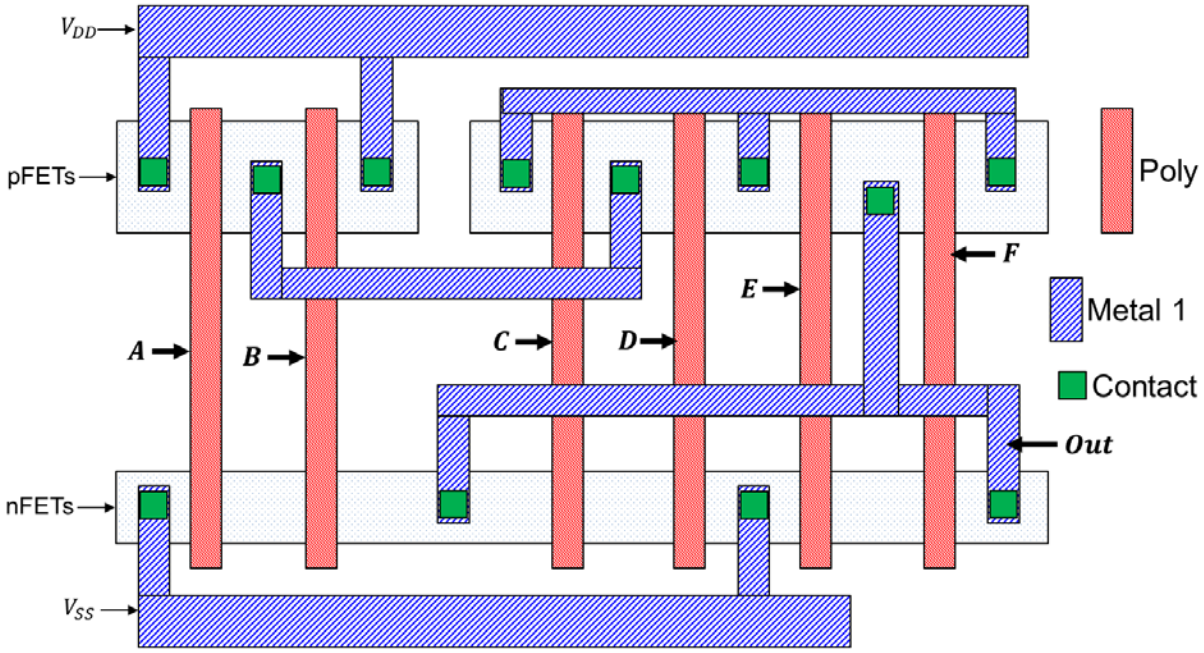
Problem	Points	
1	10	
2	10	
3	10	
4	10	
5	30	
6	10	
7	10	
8	10	
Total	80	

* Allowed: Textbooks, cheat sheets, class notes, notebooks, calculators, watches

* Not allowed: Electronic devices (smart phones, tablet PCs, laptops, etc.) except calculators and watches

Problem #1 (Layout Analysis, 10 points)

The following combinational logic has six primary inputs (A, B, C, D, E, F) and a primary output (Out). Find all input vectors that can detect a stuck-at-1 fault at input E.



$$Z = \overline{A \cdot B + C \cdot D + E \cdot F}$$

$$Z_f = \overline{A \cdot B + C \cdot D + F}$$

$$Z \oplus Z_f = 1 \rightarrow E = 0 \rightarrow F = 1 \rightarrow \; A \cdot B + C \cdot D = 0$$

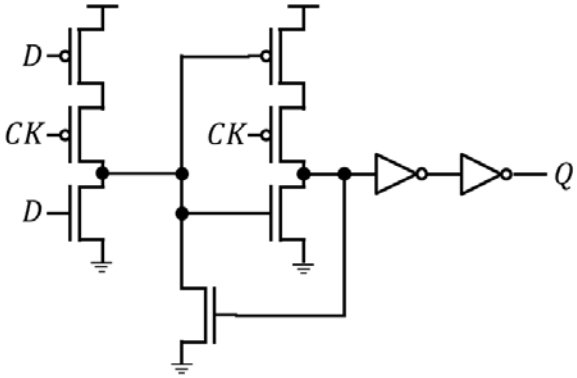
$\therefore (A\ B\ C\ D)$
$= (0\ 0\ 0\ 0), (0\ 0\ 0\ 1), (0\ 0\ 1\ 0), (0\ 1\ 0\ 0), (0\ 1\ 0\ 1), (0\ 1\ 1\ 0), (1\ 0\ 0\ 0), (1\ 0\ 0\ 1), (1\ 0\ 1\ 0)$

And

$$(E\ F) = (0\ 1)$$

Problem #2 (Static CMOS gates, 10 points)

Describe the function of the following circuit in as much detail as possible (D: data input, CK: clock).



(Partovi, ISSCC'96)

Suppose X is the node driving the pFET and the nFET in the second stage and Y is the node driving the first inverter.

- When CK=0.
  - If D=0, X is 1 due to the pFETs in the first stage. => Y=0 => Q=0.
  - If D=1, X is 0 due to the nFET in the first stage. => Y=1 => Q=1.
  - Thus, Q=D if CK=0.
- When CK=1.
  - Y is floating, i.e., it holds the previous value regardless of D.

Thus, this is an active-low D-latch (i.e., a D-latch in which Q=D when CK=0).

Problem #3 (Timing Analysis, 10 points)

Answer the following questions.

- WNS can be less than TNS (i.e., “WNS<TNS” can happen). (True/**False**)

$TNS = \sum Negative\ slack(NS) = WNS + \sum_{NS \neq WNS} NS$. $\sum_{NS \neq WNS} NS \leq 0$, so $TNS \leq WNS$.

- TNS can be less than WNS (i.e., “TNS<WNS” can happen). (**True**/False)

- WNS can be equal to TNS (i.e., “WNS=TNS” can happen). (**True**/False)

- A design has only two violating paths. In this case, the following can happen. (True/**False**)
    - “WNS of the design is -2ns and TNS of the design is -4.5ns.

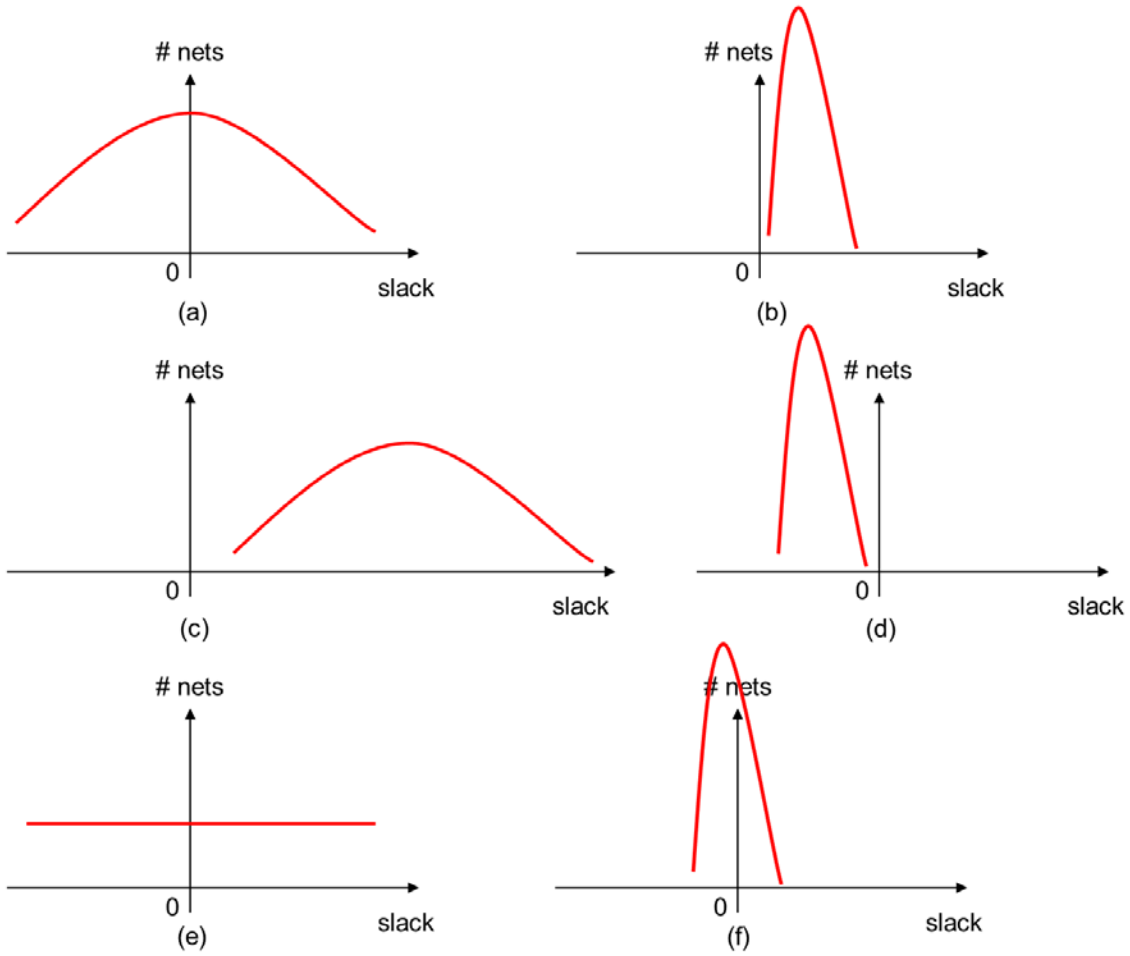
In this case, $TNS = WNS1 + WNS2$ where $WNS1$ is the negative slack of the first critical path and $WNS2$ is the negative slack of the second critical path ($WNS1 \leq WNS2$). $TNS = -4.5ns = WNS1 + WNS2 = -2ns + WNS2$, so $WNS2$ is $-2.5ns$, but this is a contradiction because $WNS1$ should be less than or equal to $WNS2$. Thus, this cannot happen.

- A design has only four violating paths. In this case, the following can happen. (True/**False**)
    - “WNS of the design is -2ns and TNS of the design is -8.2ns.

$WNS1 = -2ns \leq WNS2 \leq WNS3 \leq WNS4$. Thus, $TNS = \sum WNS \geq -8ns$, so TNS cannot be -8.2ns.

Problem #4 (Timing Analysis, 10 points)

You are given six designs, (a), (b), …, (f). Their timing analysis results are shown below. It is also known that the power consumption and the total layout area of a design are proportional to the total positive slack. You are supposed to choose a design and send it to a foundry for fabrication without any further optimization. Choose one among the six designs and explain why you decided to choose the design for fabrication.

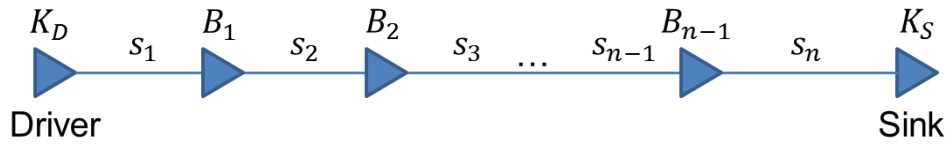


I would choose (b) for the following reasons.

	Timing	Power	Area
(a)	Violated		
(b)		Low	Small
(c)		High	Large
(d)	Violated		
(e)	Violated		
(f)	Violated		

Problem #5 (Interconnect Optimization, 30 points)

The following figure shows a net optimized by buffer insertion. The driver and the sink are denoted by K_D and K_S, respectively, and the inserted buffers are denoted by B_i ($1 \le i \le n-1$). $n \ge 2$, i.e., there is at least one buffer between the driver and the sink.



- Output resistance of K_D: R_D
- Output resistance of B_i ($1 \le i \le n-1$): R_i (e.g., $R_1, R_2, \ldots$)
- Input capacitance of K_S: C_S
- Input capacitance of B_i ($1 \le i \le n-1$): C_i
- Delay of B_i ($1 \le i \le n-1$): D_i
- Length of the i-th net ($1 \le i \le n$): s_i (um)
- $\sum_{i=1}^{n} s_i = L$ (um)
- Wire unit resistance: r (Ω/um)
- Wire unit capacitance: c (fF/um)

We assume that the net is optimized to minimize the delay from the driver to the sink.

(Hint: Derive s_1 and s_2 as functions of the above parameters when $n = 2$. You can somehow use the result for the following questions).

Suppose $R_D = R_0$ and $C_S = C_n$. Then, the delay of segment s_k is

$$\tau_k = R_{k-1}(c \cdot s_k + C_k) + r \cdot C_k \cdot s_k + \frac{1}{2} r c {s_k}^2$$

Then, the total delay is

$$\tau = \sum_{k=1}^{n} \tau_k + \sum_{k=1}^{n-1} D_k = c \sum_{k=1}^{n} R_{k-1} s_k + \sum_{k=1}^{n} R_{k-1} C_k + r \sum_{k=1}^{n} C_k s_k + \frac{1}{2} rc \sum_{k=1}^{n} {s_k}^2 + \sum_{k=1}^{n-1} D_k$$

$$\frac{\partial \tau}{\partial s_k} = c(R_{k-1} - R_{n-1}) + r(C_k - C_n) + rc(s_k - s_n) = 0$$

$$\therefore s_k = s_n + \frac{R_{n-1} - R_{k-1}}{r} + \frac{C_n - C_k}{c}$$

From $\sum_{k=1}^{n} s_k = L$, $\sum_{k=1}^{n} s_k = n \cdot s_n + \frac{1}{r}\sum_{k=1}^{n}(R_{n-1} - R_{k-1}) + \frac{1}{c}\sum_{k=1}^{n}(C_n - C_k) = n \cdot s_n + \frac{n \cdot R_{n-1} - R_T}{r} + \frac{n \cdot C_n - C_T}{c} = L$ where $R_T = \sum_{k=0}^{n-1} R_k$ and $C_T = \sum_{k=1}^{n} C_k$.

Thus, we obtain the following:

$$s_k = \frac{L}{n} + \frac{1}{r} \cdot \left(\frac{1}{n} \cdot R_T - R_{k-1}\right) + \frac{1}{c} \cdot \left(\frac{1}{n} \cdot C_T - C_k\right)$$

Answer the following questions for $n = 10$ (i.e., we insert 9 buffers optimally):

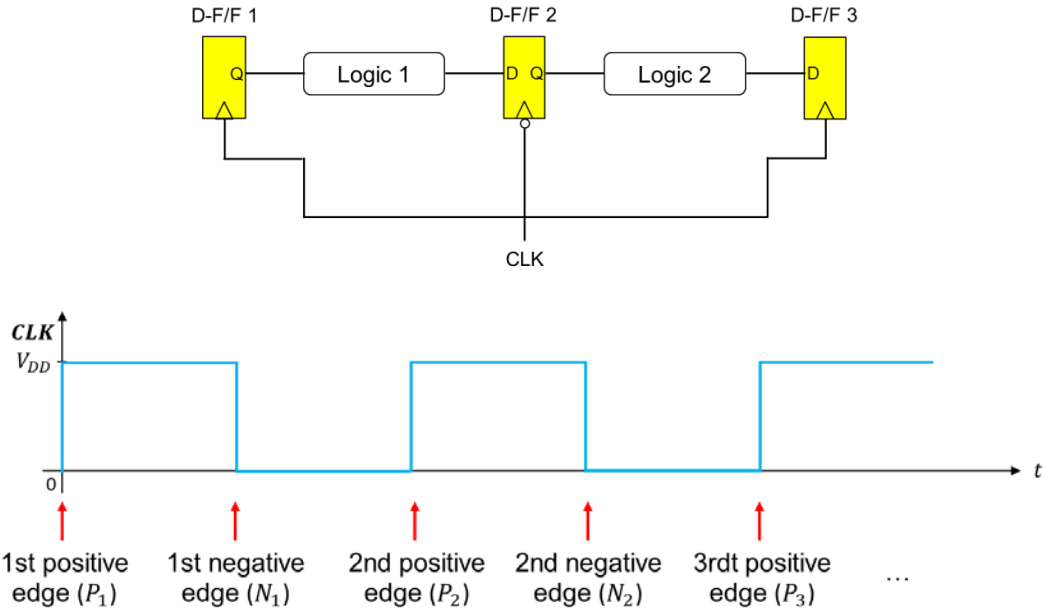
- If C_S increases, we should increase s_1 to minimize the total delay. (**True**/False)
  - If C_S increases, the delay of s_{10} goes up, so we should increase s_1, …, s_9.
- If R_9 increases, we should increase s_1 to minimize the total delay. (**True**/False)
  - If R_9 increases, the delay of s_9 goes up, so we should increase s_1, …, s_8, s_{10}.
- If D_9 increases, we should increase s_1 to minimize the total delay. (True/**False**)
  - Since we always insert 9 buffers, D_9 does not affect the total delay.
- If C_5 increases, we should increase s_8 to minimize the total delay. (**True**/False)
  - True for the same reason as the case of C_S ↑.
- If R_5 increases, we should increase s_8 to minimize the total delay. (**True**/False)
  - True for the same reason as the case of R_9 ↑.
- If D_5 increases, we should increase s_8 to minimize the total delay. (True/**False**)
  - False for the same reason as the case of D_9 ↑.
- If R_D increases, we should increase s_1 to minimize the total delay. (True/**False**)
  - If $R_D(R_0)$ increases, the delay of s_1 goes up, so we should increase s_2, …, , s_{10}.
- Suppose $s_1 \approx 0$ because $R_D \gg R_1, \ldots, R_9$. In this case, if r and c increase at the same time, we should increase s_1 to minimize the total delay. (**True**/False)
  - If $r \to \infty$ and $c \to \infty$, the impact of output resistance and input capacitance on the delay reduces. In this case, the impact of the wire delay portion ($\frac{1}{2}rcl^2$) goes up, so we should evenly distribute the buffers to minimize the total delay. Since s_1 was almost 0, we should increase s_1.

Answer the following questions assuming n is to be determined optimally (i.e., we find # buffers ($n-1$) and $s_1 \sim s_n$ optimally) and $L \gg 0$, so $n \gg 1$:

- If R_D increases, we should increase n to minimize the total delay. (**True**/False)
  - In this case, s_1 should be decreased to reduce the total delay, which increases s_1, …, s_n. Thus, we should insert more buffers in general.

- If C_2 increases, we should increase n to minimize the total delay. (**True**/False)
  - In this case, we should decrease s_2 and increase all the other s_k. If their lengths go up, we should insert more buffers.
- If C_S increases, we should increase n to minimize the total delay. (**True**/False)
  - We should insert more buffers for the same reason as the case C_S increases.
- If the delay of each buffer is increased, we should generally increase n to minimize the total delay. (True/**False**)
  - We should decrease the number of buffers because the buffer delay has negative impact on the total delay.
- If L increases, we should increase n to minimize the total delay. (**True**/False)
- If r increases, we should increase n to minimize the total delay. (**True**/False)
  - The total delay when there is no buffer is $\tau = R_D(c \cdot L + C_S) + r \cdot L \cdot C_S + \frac{1}{2}rcL^2$.
  - In this formula, r can be treated as weighting factors for $L \cdot C_S$ and $\frac{1}{2}cL^2$. If r increases, inserting more buffers can reduce $\frac{1}{2}rcL^2$. For example, if a buffer is inserted, the sum of the values is $\frac{1}{2}rc(0.25L^2 + 0.25L^2) = \frac{1}{4}rcL^2$. Similarly, if three buffers are inserted, the sum of the values is $\frac{1}{2}rc(L^2/16 + L^2/16 + L^2/16 + L^2/16) = \frac{1}{8}rcL^2$. Although the buffer delays and buffer input capacitance values are delay overheads, if r increases significantly, inserting more buffers helps reduce the delay as shown above.
- If c increases, we should increase n to minimize the total delay. (**True**/False)
  - r and c basically have similar impacts on the total delay, so if c goes up, we should insert more buffers.

Problem #6 (Timing Analysis, 10 points)



- Setup time of a D-FF: T_s
- Hold time of a D-FF: T_h
- D-F/F internal delay: T_{CQ}
- Logic 1 delay: T_{L1}
- Logic 2 delay: T_{L2}
- Clock period: T_{CK} (duty cycle: 50%, i.e., the clock is high for $T_{CK}/2$ and low for $T_{CK}/2$.)
- Delay from CLK to D-FF 1: D_1
- Delay from CLK to D-FF 2: D_2
- Delay from CLK to D-FF 3: D_3
- D-F/F 2 is a negative-edge FF (i.e., it captures the input signal at falling edges.)

The above figure shows three FFs connects in series. D-FF 2 is a negative-edge FF, whereas D-FF 1 and 3 are positive-edge FFs. The operation of the circuit is as follows. D-FF 1 captures its input signal at k-th positive clock edge P_k. Logic 1 performs computation for the output of D-FF 1. D-FF 2 captures its input signal at k-th negative clock edge N_k. Logic 2 performs computation for the output of D-FF2. D-FF 3 captures its input signal at $(k+1)$-th positive clock edge P_{k+1}.

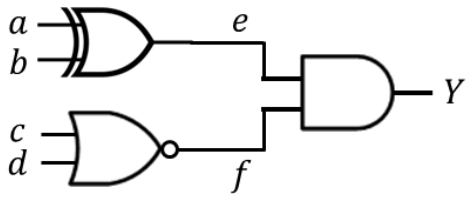
Derive two setup time inequalities (one for Logic 1 and the other for Logic 2).

1) For Logic 1, $D_1 + T_{CQ} + T_{L1} \leq D_2 + \frac{T_{CK}}{2} - T_s \Rightarrow T_{L1} \leq (D_2 - D_1) + \frac{T_{CK}}{2} - T_{CQ} - T_s$

2) For Logic 2, $D_2 + \frac{T_{CK}}{2} + T_{CQ} + T_{L2} \leq D_3 + T_{CK} - T_s \implies T_{L2} \leq (D_3 - D_2) + \frac{T_{CK}}{2} - T_{CQ} - T_s$

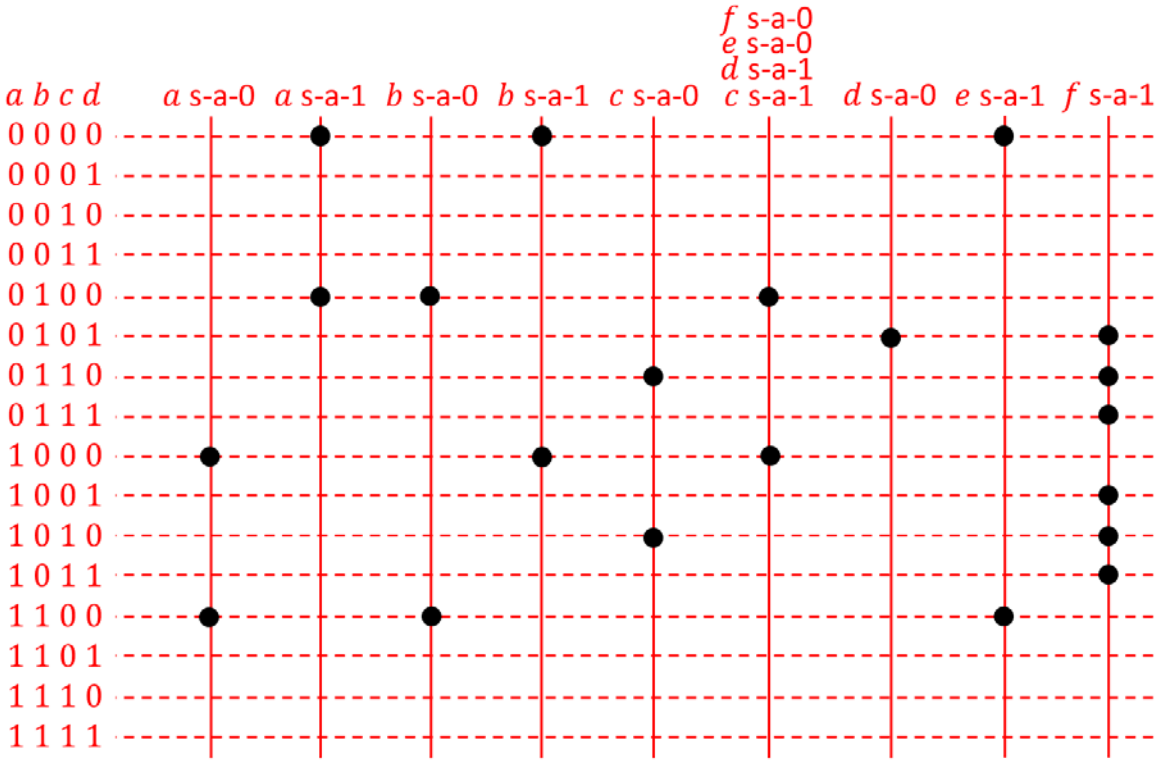
Problem #7 (Testing, 10 points)

We want to detect stuck-at-0 and stuck-at-1 faults at all the primary inputs, a, b, c, d, and the two internal nodes, e, f. Computation of Y to detect a stuck-at-0/1 fault at an internal node can be done by setting the value of the node to constant 0 (for stuck-at-0 faults) or 1 (for stuck-at-1 faults). Find a ***minimal*** set of test vectors that can detect all the s-a-0 and s-a-1 faults at $a, b, c, d, e,$ and f for the following logic (Hint: all the minimal sets have five test vectors).



$$Y = (a \oplus b) \cdot \overline{c + d}$$

- s-a-0 at a: $Y_f = b \cdot \overline{c + d}.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus \{b \cdot \overline{c + d}\} = 1 \Rightarrow (a\ b\ c\ d) = (1\ 0\ 0\ 0)$ or $(1\ 1\ 0\ 0)$
- s-a-1 at a: $Y_f = \bar{b} \cdot \overline{c + d}.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus \{\bar{b} \cdot \overline{c + d}\} = 1 \Rightarrow (a\ b\ c\ d) = (0\ 0\ 0\ 0)$ or $(0\ 1\ 0\ 0)$
- s-a-0 at b: $(a\ b\ c\ d) = (0\ 1\ 0\ 0)$ or $(1\ 1\ 0\ 0)$
- s-a-1 at b: $(a\ b\ c\ d) = (0\ 0\ 0\ 0)$ or $(1\ 0\ 0\ 0)$
- s-a-0 at c: $Y_f = (a \oplus b) \cdot \bar{d}.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus \{(a \oplus b) \cdot \bar{d}\} = 1 \Rightarrow (a\ b\ c\ d) = (0\ 1\ 1\ 0)$ or $(1\ 0\ 1\ 0)$
- s-a-1 at c: $Y_f = 0.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus 0 = 1 \Rightarrow (a\ b\ c\ d) = (0\ 1\ 0\ 0)$ or $(1\ 0\ 0\ 0)$
- s-a-0 at d: $(a\ b\ c\ d) = (0\ 1\ 0\ 1)$ or $(1\ 0\ 0\ 1)$
- s-a-1 at d: $(a\ b\ c\ d) = (0\ 1\ 0\ 0)$ or $(1\ 0\ 0\ 0)$
- s-a-0 at e: $Y_f = 0.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus 0 = 1 \Rightarrow (a\ b\ c\ d) = (0\ 1\ 0\ 0)$ or $(1\ 0\ 0\ 0)$
- s-a-1 at e: $Y_f = \overline{c + d}.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus \{\overline{c + d}\} = 1 \Rightarrow (a\ b\ c\ d) = (0\ 0\ 0\ 0)$ or $(1\ 1\ 0\ 0)$
- s-a-0 at f: $Y_f = 0.Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus 0 = 1 \Rightarrow (a\ b\ c\ d) = (0\ 1\ 0\ 0)$ or $(1\ 0\ 0\ 0)$
- s-a-1 at f: $Y_f = (a \oplus b).Y \oplus Y_f = \{(a \oplus b) \cdot \overline{c + d}\} \oplus \{(a \oplus b)\} = 1 \Rightarrow (a\ b\ c\ d) = (0\ 1\ 0\ 1)$ or $(0\ 1\ 1\ 0)$ or $(0\ 1\ 1\ 1)$ or $(1\ 0\ 0\ 1)$ or $(1\ 0\ 1\ 0)$ or $(1\ 0\ 1\ 1)$



1) Need (0 1 0 1) to cover the 7th column. It also covers the 9th column.
2) Covering the 1st column requires either (1 0 0 0) or (1 1 0 0).
3) Covering the 2nd column requires either (0 0 0 0) or (0 1 0 0).
4) For (1 0 0 0) and (0 0 0 0), we should cover the 3rd and the 5th columns.
5) If we proceed this way, we get the following test vectors.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Set 1	■				■	■	■		■							
Set 2	■				■	■			■		■					
Set 3	■					■	■		■				■			
Set 4	■					■			■		■		■			
Set 5					■	■	■		■				■			
Set 6					■	■			■		■		■			
Set 7	■				■	■	■						■			
Set 8	■				■	■					■		■			

Problem #8 (Testing, 10 points)

A combinational logic G is given. It has n inputs and one output. The inputs are $x_1, x_2, \dots, x_n$ $(n \geq 2)$ and the output is y, i.e., $y = G(x_1, \dots, x_n)$. To find an input vector that can detect a stuck-at-v fault ($v = 0$ or 1) at x_i $(1 \leq i \leq n)$, we solve $G(x_1, \dots, x_n) \oplus G_f(x_1, \dots, x_i = v, \dots, x_n) = 1$. Let S_i be the set of all input vectors that can detect a stuck-at-v_i ($v_i = 0$ or 1) fault at x_i and S_k be the set of all input vectors that can detect a stuck-at-v_k ($v_k = 0$ or 1) fault at x_k $(i \neq k)$.

If we assume that two stuck-at-v faults occur at the same time, we can find an input vector that can detect the faults. For example, we solve $G(x_1, \dots, x_n) \oplus G_f(x_1, \dots, x_i = v_i, \dots, x_k = v_k, \dots, x_n) = 1$ to find an input vector that can detect a stuck-at-v_i fault at x_i and a stuck-at-v_k fault at x_k occurring at the same time $(i \neq k)$. Let $S_{i,k}$ be the set of all input vectors that can detect a stuck-at-v_i ($v_i = 0$ or 1) fault at x_i and a stuck-at-v_k ($v_k = 0$ or 1) fault at x_k $(i \neq k)$.

Prove or disprove the following statement:

$$S_{i,k} = S_i \cap S_k$$

(If you want to disprove it, you can just show a counterexample.)

Counterexample: A two-input AND gate (inputs: a, b, output: Z).

$$Z = a \cdot b$$

Let a s-a-0 fault at input a be f_a and a s-a-1 fault at input b be f_b. For f_a, $Z_f = 0$, so $S_a = \{(a, b) = (1\ 1)\}$. For f_b, $Z_f = a$, so $S_b = \{(a, b) = (1\ 0)\}$. $S_a \cap S_b = \emptyset$.

When the two faults occur at the same time, $Z_f = 0$. In this case, $S_{a,b} = \{(a, b) = (1\ 1)\}$. Thus, $S_{a,b} \neq S_a \cap S_b$.