2 Problem Set Part B

1. Hazardous Compounds

In class, you have studied the properties of standard two-level circuits contaminated with static hazards. In this question, you will take this analysis to another level – literally. Put on your hazard suit, and prepare to get your hands glitchy.

For succinctness, we will use the following terms to describe different aspects of a static hazard:
Hazard Variable: The variable for which a switch in value causes a static hazard to occur.
Hazard Environment: The values that non-hazard variables must take in order for a switch in the hazard variable to trigger a static hazard.
Hazard Orientation: The "base state" of the hazard, from which a glitch temporarily deviates; a hazard with orientation "0" has "010" behavior; a hazard with orientation "1" has "101" behavior.
Hazard Direction: The direction in which the *Hazard variable* must change to trigger the hazard; can be "0→1" or "1→0".

(Part A) The minimal implementations of a function with the following Karnaugh map are vulnerable to static hazards:

wx \ yz	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	0	0	1	1

i) Find the minimal Sum-of-Products implementation of the function. Then, for each static hazard, specify its hazard environment, its hazard variable, its hazard orientation, and its hazard direction.

SoP = $w'z + wxy' + wx'y$

Hazard	Variable	Environment	Orientation	Direction
1	w	$x=1, y=0, z=1$	1	$1 \to 0$
2	w	$x=0, y=1, z=1$	1	$1 \to 0$

ii) Find the minimal Product-of-Sums implementation of the function. Then, for each static hazard, specify its hazard environment, its hazard variable, its hazard orientation, and its hazard direction.

PoS = $(w+z)(w'+x+y)(w'+x'+y')$

Hazard	Variable	Environment	Orientation	Direction
1	w	$x=0, y=0, z=0$	0	$0 \to 1$
2	w	$x=1, y=1, z=0$	0	$0 \to 1$

Scanned by CamScanner

(Part B-C) One way to think of static hazards is as timing delays between sub-functions of a larger function. In a Sum-of-Products, for example, we compose a few simple "AND" functions with an OR gate. Static hazards arise when more than one of these sub-functions (in this case, individual implicants) change due to a change in the inputs, and when these simultaneous changes occur at different speeds. In the following sections, we will explore the nature of static hazards in larger composite functions. Let $F = G_1G_2$, where G_1 and G_2 are distinct functions of the same four variables (w, x, y, and z), and G_1 and G_2 are both implemented in two-level Product-of-Sums form.

(Part B) Suppose G_1 and G_2 have the following Karnaugh Maps:

G_1

wx \ yz	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

G_2

wx \ yz	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	0	0

You will notice that neither G_1 nor G_2 exhibit static hazards. Nevertheless, it is possible to trigger a static hazard in $F = G_1G_2$. **For each static hazard in F, identify the associated hazard environment, hazard variable, hazard orientation, and hazard direction.**

Hazard	Variable	Environment	Orientation	Direction
1	w	$x=0, y=0, z=1$	0	$0 \to 1$
2	y	$w=0, x=0, z=1$	0	$0 \to 1$
3	x	$w=0, y=1, z=1$	0	$0 \to 1$

(Part C) Because F is formed by taking the AND of the two functions G_1 and G_2, it is possible to make changes to the Karnaugh maps for G_1 and G_2 without changing the logical behavior of F. Fortunately, these changes can be leveraged to eliminate the static hazards you found in (Part B).

i) Describe the changes you can make to the Karnaugh maps for G_1 and G_2 without changing F's logical behavior. Why can you make these changes?

For any input $k = \{*, w, x, y, z\}$, if $G_1(k)=0$ OR $G_2(k)=0$, we can change them to any of the following: $G_1(k)=0$ & $G_2(k)=0$ OR $G_1(k)=1$ & $G_2(k)=0$ OR $G_1(k)=0$ & $G_2(k)=1$.

This is because $F = G_1G_2$ hence if $F(k)=0$, it doesn't matter which is zero as long as at least one is.

ii) Identify the changes you would make to G_1 and/or G_2 to eliminate the static hazards you identified in (Part B).

Set $G_1 = F$ or $G_2 = F$, this by definition makes a static hazard impossible since the implicants are fully covered by only one.

Eg: $G_1 =$

wx \ yz	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	0
10	0	0	0	0

Scanned by CamScanner

(Part D-E) Suppose that F was instead given by the equation $F = G_1 + G_2$.

(Part D) The Karnaugh maps of G_1 and G_2 from (Part B) have been reproduced below for your convenience.

G_1

wx \ yz	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

G_2

wx \ yz	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	0	0

For each static hazard in F, identify the associated hazard environment, hazard variable, hazard orientation, and hazard direction.

Hazard	Variable	Environment	Orientation	Direction
1	w	x=0, y=0, z=1	1	1 → 0
2	y	w=0, x=0, z=1	1	1 → 0
3	x	w=0, y=1, z=1	1	1 → 0

(Part E) Now that you are using an OR gate instead of an AND gate to compose the function F, the modifications you identified in (Part C) can no longer be made without altering the output behavior of F.

i) Describe the changes you can make to the Karnaugh maps for G_1 and G_2 without changing F's logical behavior, now that F is implemented with an OR composition of G_1 and G_2?

For an input k = {w,x,y,z}, if $G_1(k) = 1$ OR $G_2(k) = 1$, then we can change $G_1(k)$ & $G_2(k)$ to any combination such that atleast one of them is a one, ie: (0,1), (1,0), (1,1)

This is because, we cannot tell from an output 1 in F whether either G_1 or G_2 or both are one.

ii) Identify the changes you would make to G_1 and/or G_2 to eliminate the static hazards you identified above.

Change G_1 →

wx \ yz	00	01	11	10
00	0	1	1	1
01	0	1	1	0
11	1	1	1	1
10	1	1	0	0

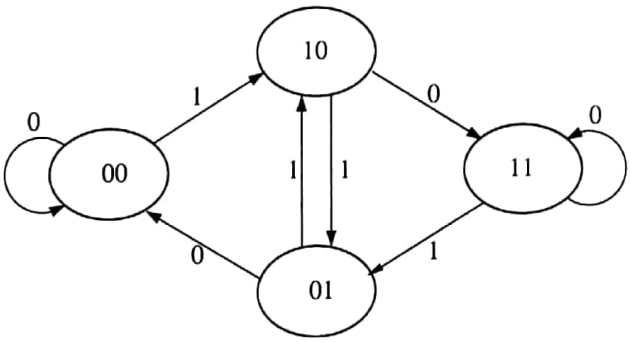
$G_1 = (w + y + z)(w' + x + y')(w + x' + z)$

G_1 has no 0-hazards as PoS. $F = G_1 + G_2$ has G_2 completely overlapped hence no hazard is possible.

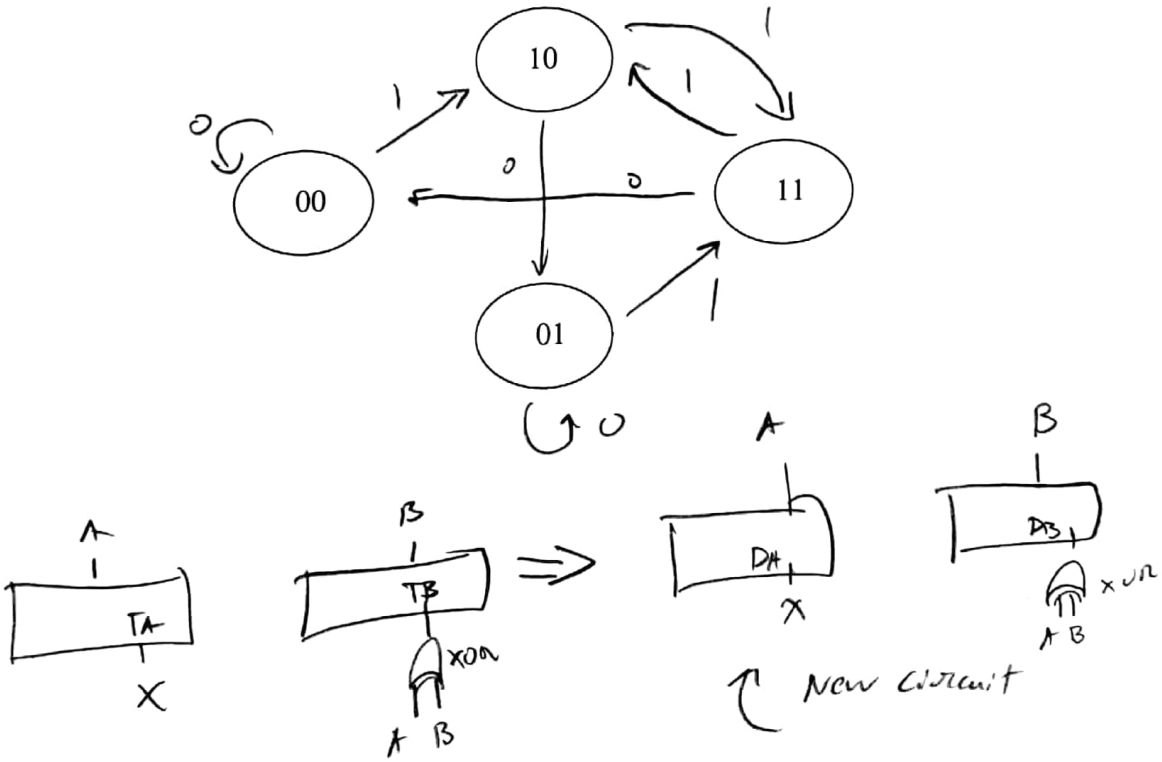
Scanned by CamScanner

2. FSM transformations

(Part A) The following FSM has been implemented using two **T** flip-flops and a set of gates for the next state logic.

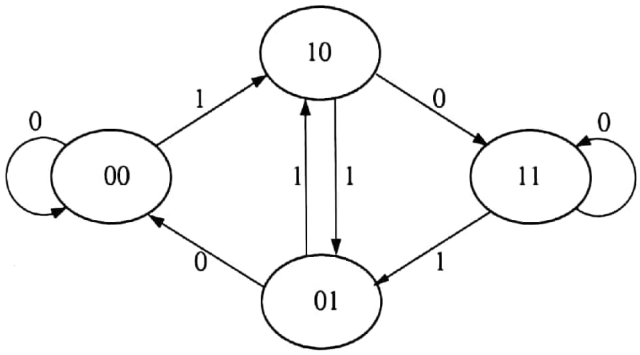


A simple modification is made to the sequential circuit that implements this FSM; both **T** flip-flops are replaced with **D** flip-flops, while the next state logic is kept intact. Please identify the FSM that is implemented by the modified circuit through drawing transitions on the following FSM skeleton.

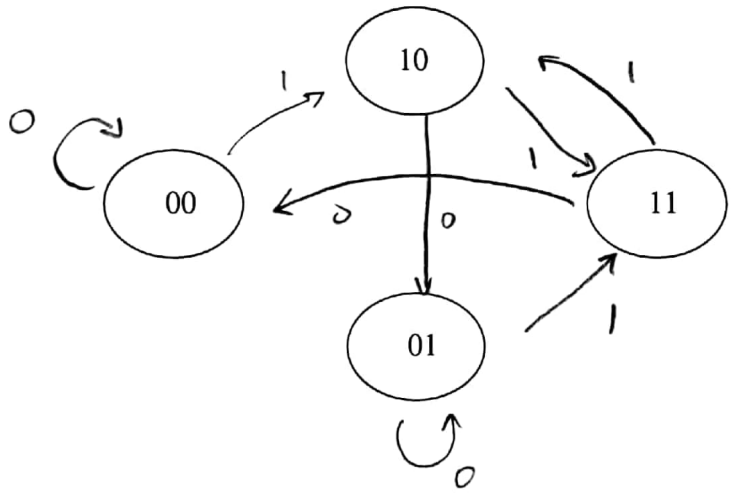


Scanned by CamScanner

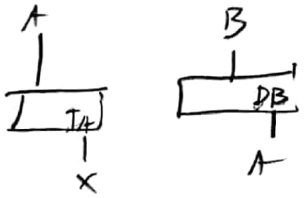
(Part B) The following FSM has been implemented using a **D** flip-flop, a **T** flip-flop, and a set of gates for the next state logic. However, we do not know whether the D flip-flop is used to control the most significant bit or the least significant bit.



A simple modification is made to the sequential circuit that implements this FSM; the **T** and the **D** flip-flops are swapped *(so that the one that used to control the most significant bit now controls the least significant bit and vice versa)*, while the next state logic is kept intact. Please identify the FSM that is implemented by the modified circuit through drawing transitions on the following FSM skeleton.

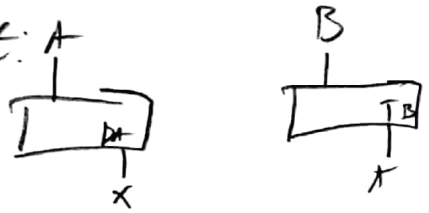


Old circuit:



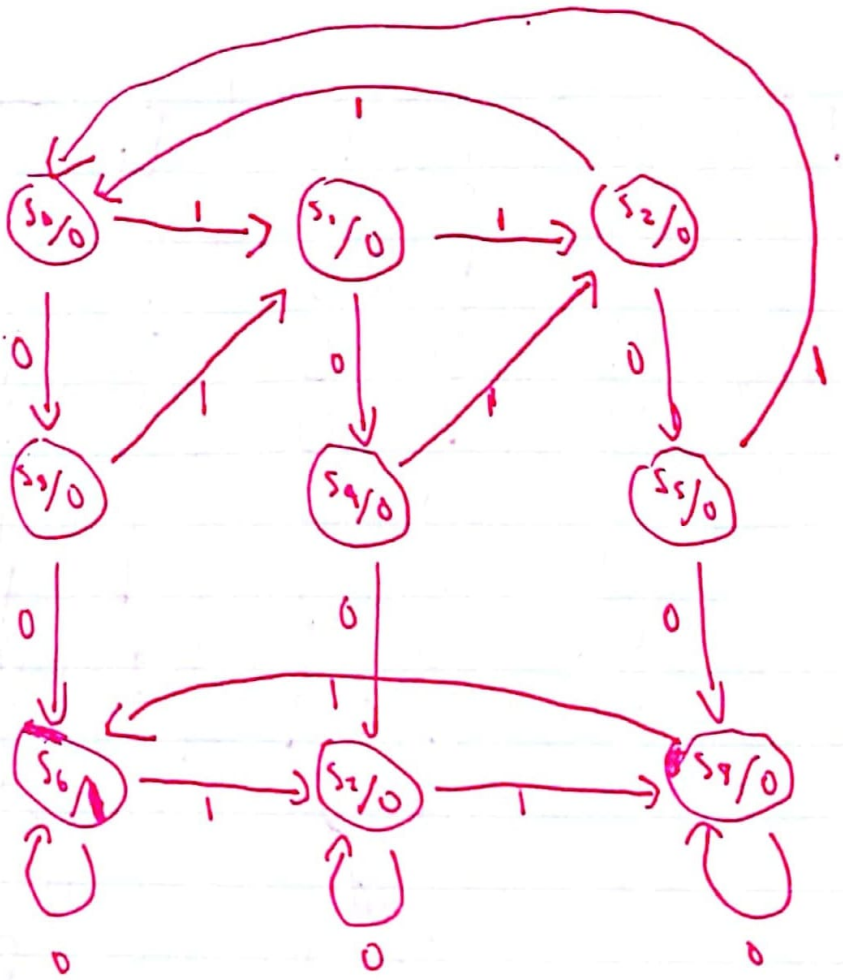
AB	X=0	X=1
00	00	10
01	01	11
10	01	11
11	00	10

New circuit:



Scanned by CamScanner

3 i)



ii)

start

S_0 = reset or 0 mod 3 ones

S_1 = 1 mod 3 ones

S_2 = 2 mod 3 ones

S_3 = ends in 0 & 0 mod 3 ones & no 00

S_4 = ends in 0 & 1 mod 3 ones & no 00

S_5 = ends in 0 & 2 mod 3 ones & no 00

S_6 = 00 has occured & 0 mod 3 ones

S_7 = 00 has occ & 1 mod 3 ones

S_8 = 00 has occured & 2 mod 3 ones

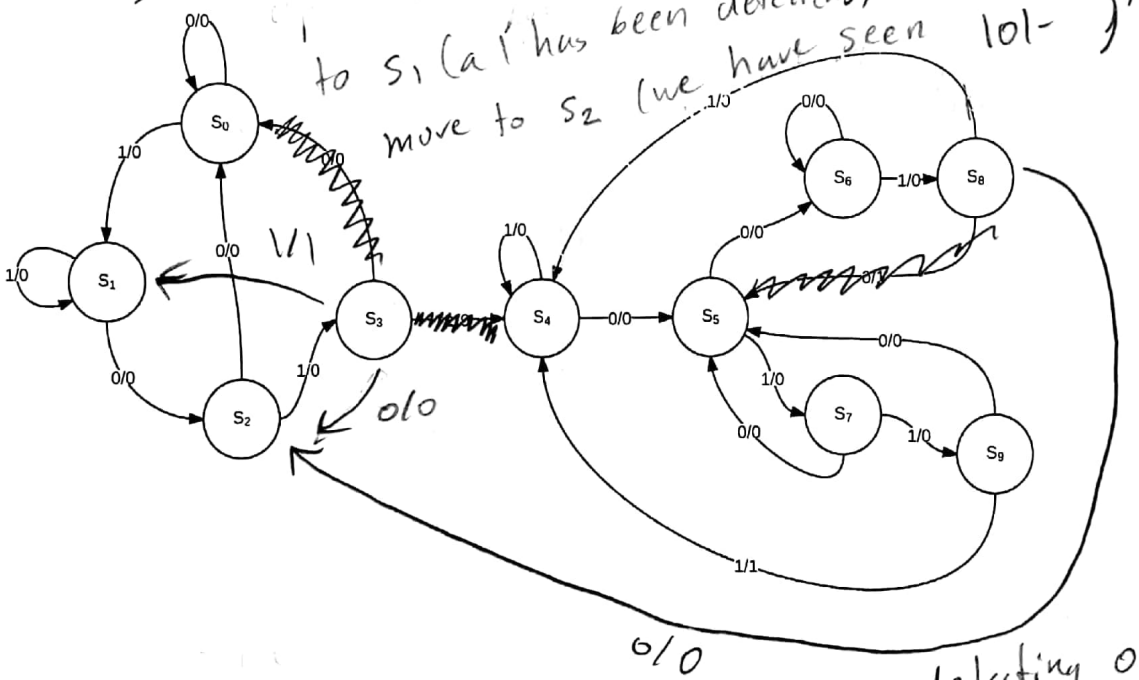
Scanned by CamScanner

(Part B) Shortly after the release of their soon-to-be successful modulo 3 detector that you helped develop, Sequence Detectors "Я" Us has come up with a new design that they believe will take the sequence detector market by storm, giving them a complete monopoly. They have come up with a sequence detector that waits until it sees the sequence "1011", after which it will recognize the sequences "0010" and "0111". If it recognizes either of those two sequences it will output a "1"; otherwise, it will just output "0".

However, the CEO is concerned that the sequence detector is not innovative enough and instead proposes that it should be a sequence detector that recognizes the sequence "0111", until the sequence "0010" is detected, after which it should recognize the sequence "1011" instead. Keep in mind that these sequences may have overlap.

i. As this is exactly the kind of task you were hired for, the team behind the original sequence detector design asks you to modify their original Mealy FSM state diagram (which starts in state S_0) so that it behaves according to the specifications set forth by the CEO. Furthermore, they believe you can alter the functionality by replacing (adding and removing) no more than 3 transitions, and altering the output of no more than 1 transition (not including the ones that were added or removed). For each transition you replace with a new one, please **give your rationale for why the change is necessary.**

-we recognize 1011. At S_3, we have detected 101.
S_0, on a 1 input, we output 1, and move back to S_1 (a 1 has been detected), if we see 0 in S_3, move to S_2 (we have seen 101-)



-Upon detecting 0010, we move to recognizing 1011 sequences. Since we have already detected 10-, we can get a head-start and begin on S_2

ii. What state should the modified FSM start in?

S_4

Scanned by CamScanner