Exam 3

Question 1A) Assume S_0, S_1 are present state, and S_0', S_1' are next state
From the given FSM schematic, we can have

next state equation:

$$S_0' = \bar{S}_1 \bar{S}_0 A_1 A_0$$

$$S_1' = S_0 \bar{A}_1 A_0$$

output state equation:

$$\text{output} = S_1$$

1B) From the given FSM schematic, the output connected with S_1 so the output $= S_1$ (Moore), output only depends on state not depends on inputs, state "11" "10" have the output = 1.

1C) truth table for next state and output logic will be

present state		Input		next state		output
S_1	S_0	A_1	A_0	S_1'	S_0'	
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	0	0	1
1	1	0	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	1

1D) State transition table

Assume the states be

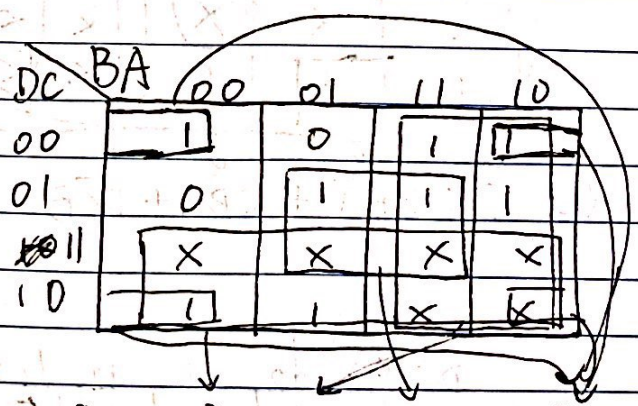
	S_1	S_0
S_0 =	0	0
S_1 =	0	1
S_2 =	1	0
S_3 =	1	1

present state	inputs		next state
S	A_1	A_0	S'
S_0	0	0	S_0
S_0	0	1	S_0
S_0	1	0	S_0
S_0	1	1	S_1
S_1	0	0	S_0
S_1	0	1	S_2
S_1	1	0	S_0
S_1	1	1	S_0
S_2	0	0	S_0
S_2	0	1	S_0
S_2	1	0	S_0
S_2	1	1	S_0
S_3	0	0	S_0
S_3	0	1	S_2
S_3	1	0	S_0
S_3	1	1	S_0

2A) Truth table for the circuit (X – don't care)

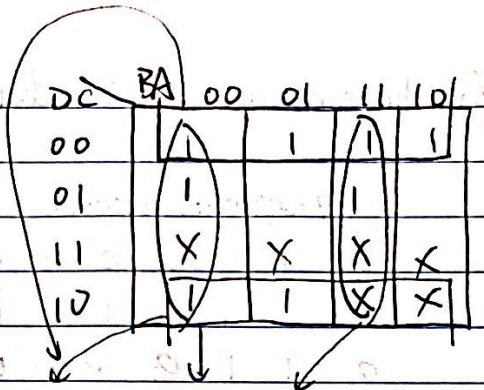
Inputs D C B A	7-segment output a b c d e f g	Display on board
0 0 0 0	1 1 1 1 1 1 0	0
0 0 0 1	0 1 1 0 0 0 0	1
0 0 1 0	1 1 0 1 1 0 1	2
0 0 1 1	1 1 1 1 0 0 1	3
0 1 0 0	0 1 1 0 0 1 1	4
0 1 0 1	1 0 1 1 0 1 1	5
0 1 1 0	1 0 1 1 1 1 1	6
0 1 1 1	1 1 1 0 0 0 0	7
1 0 0 0	1 1 1 1 1 1 1	8
1 0 0 1	1 1 1 1 0 1 1	9
1 0 1 0	X	X
1 0 1 1	X	X
1 1 0 0	X	X
1 1 0 1	X	X
1 1 1 0	X	X
1 1 1 1	X	X

2B) For a



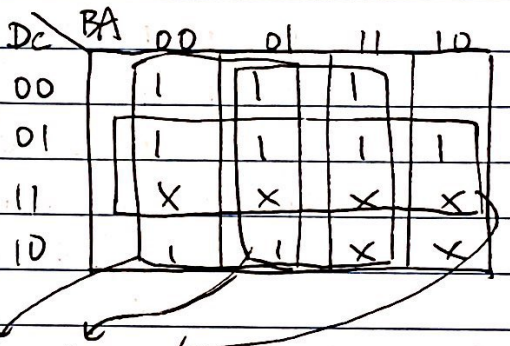
From the graph $a = D + B + CA + \bar{C}\bar{A}$

For b



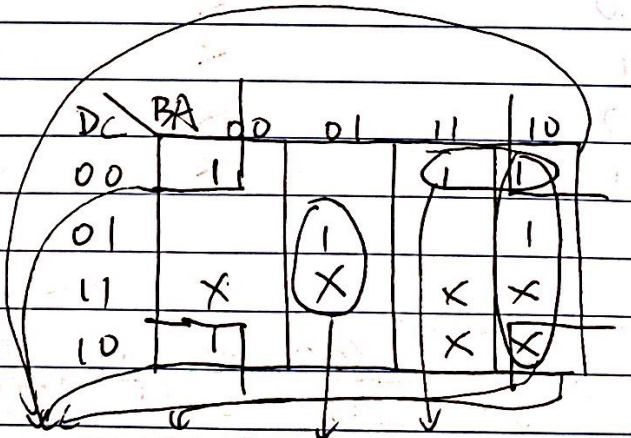
$b = \bar{C} + \bar{A}\bar{B} + A\bar{B}$

For c



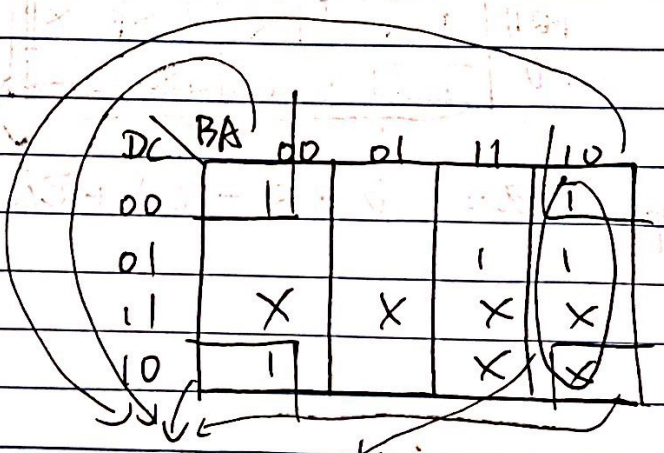
$C = \bar{B} + A + C$

For d



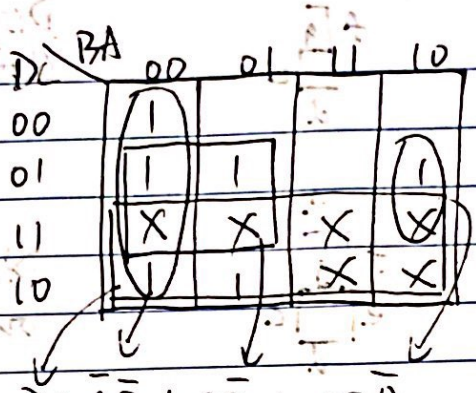
$d = \bar{C}\bar{A} + B\bar{A} + C\bar{B}A + \bar{D}\bar{C}B$

For e



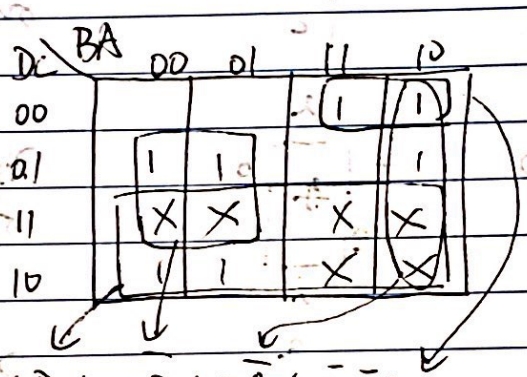
$e = \bar{A}C + B\bar{A}$

For f



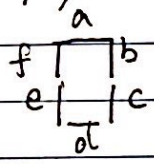
$f = D + \bar{A}\bar{B} + C\bar{B} + C\bar{B}\bar{A}$

For g



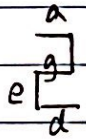
$g = D + C\bar{B} + B\bar{A} + \bar{D}\bar{C}\bar{B}$

2C) For input DCBA 0000 Display on board is 0

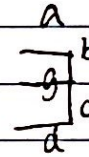


input 0001 Display is 1

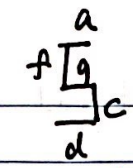
input 0010 Display is 2

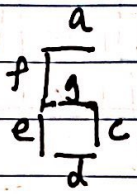


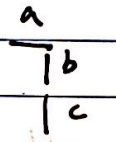
input 0011 Display is 3

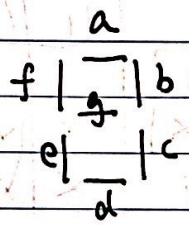


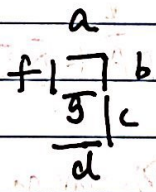
input 0100 Display is 4

Input 0101  Display Output is 5

Input 0110  Display output is 6

Input 0111  Display output is 7

input 1000  Display output is 8

Input 1001  Display output is 9

3. For 4-bit fibonacci. first have a truth table.

4-bit	binary	is Fibo Number?
0000	[0]	1
0001	[1]	1
0010	[2]	1
0011	[3]	1
0100	4	0
0101	[5]	1
0110	6	0
0111	7	0
1000	[8]	1
1001	9	0
1010	10	0
1011	11	0
1100	12	0
1101	[13]	1
1110	14	0
1111	15	0

4. ~~state diagram~~

```
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yxz15>cd c:\iverilog

c:\iverilog>iverilog -o simout Q3.v Q3Test.v

c:\iverilog>vvp simout
A3 = 0, A2 = 0, A1 = 0, A0 = 0, Y = 1
A3 = 0, A2 = 0, A1 = 0, A0 = 1, Y = 1
A3 = 0, A2 = 0, A1 = 1, A0 = 0, Y = 1
A3 = 0, A2 = 0, A1 = 1, A0 = 1, Y = 1
A3 = 0, A2 = 1, A1 = 0, A0 = 0, Y = 0
A3 = 0, A2 = 1, A1 = 0, A0 = 1, Y = 1
A3 = 0, A2 = 1, A1 = 1, A0 = 0, Y = 0
A3 = 0, A2 = 1, A1 = 1, A0 = 1, Y = 0
A3 = 1, A2 = 0, A1 = 0, A0 = 0, Y = 1
A3 = 1, A2 = 0, A1 = 0, A0 = 1, Y = 0
A3 = 1, A2 = 0, A1 = 1, A0 = 0, Y = 0
A3 = 1, A2 = 0, A1 = 1, A0 = 1, Y = 0
A3 = 1, A2 = 1, A1 = 0, A0 = 0, Y = 0
A3 = 1, A2 = 1, A1 = 0, A0 = 1, Y = 1
A3 = 1, A2 = 1, A1 = 1, A0 = 0, Y = 0
A3 = 1, A2 = 1, A1 = 1, A0 = 1, Y = 0
```

4. State Diagram

S_0	00		
S_1	01	grn	00
S_2	10	ylw	01
S_3	11	rd	10

From the code, we know states are divided into

T_A, T_B works as present as input timelines, from where the state will transite to another state

// next state logic

$S_0 \xrightarrow{\text{if}(\overline{T_A})} S_1 \rightarrow S_2 \xrightarrow{\text{if}(\overline{T_B})} S_3 \rightarrow S_0$

S_1: else $\rightarrow S_0$

S_2: else $\rightarrow S_2$

// output logic (L_B, L_A are output lines, with display grn, ylw or rd)

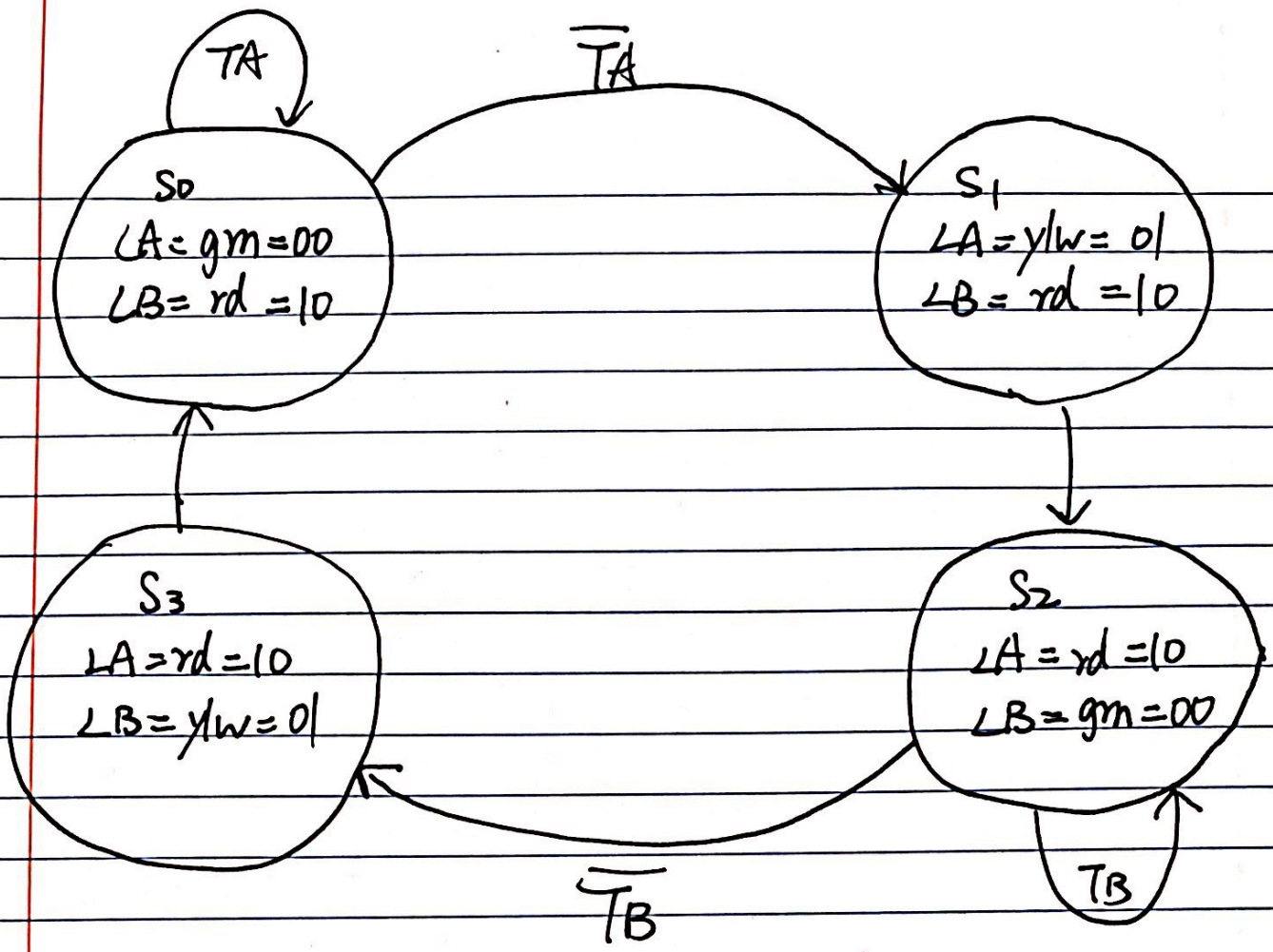
if(S_0) { L_B = rd; L_A = grn

if(S_1) { L_B = grn; L_A = ylw

if(S_2) { L_B = grn; L_A = rd

else(S_3) { L_B = ylw; L_A = rd.

From the logic and next state provided, we can have a state diagram.



It's a traffic light system, two roads LA and LB. (will output traffic light red, green, yellow) along with the change of TA, TB.

Direct mapped : 1 block per set

5A) 16-word cache, 1 word size C=16-word

* block size (b) = number of words in the cache block

S=16 * so block offset = 0

* Since $\log_2 4 = 16$, then we need 4-bits for Block Index

* now we have 4-bits tag + 4-bits block index

* since the question is about hexadecimal deal with 0, 4, 8, C

* as the addresses at 40, 44, 48, 4C, the lease 4-bits is index the most 4-bits are tag,

* so

	tag	index	
40	4	0	
44	4	4	
48	4	8	
4C	4	C	as the initialized.

* the next addresses are 70, 74, 78, 7C

	tag	index	
70	7	0	
74	7	4	
78	7	8	
7C	7	C	as placement

* next addresses are 80, 84, 88, 8C

	tag	index	
80	8	0	
84	8	4	
88	8	8	
8C	8	C	as placement

* next addresses are 90, 94, 98, 9C

	tag	index	
90	9	0	
94	9	4	
98	9	8	
9C	9	C	as placement

* next addresses are 0, 4, 8, C

	tag	index	
0	0	0	
4	0	4	
8	0	8	
C	0	C	as placement

* next addresses are 10, 14, 18, 1C, 20

	tag	index	
10	1	0	
14	1	4	
18	1	8	as placement
1C	1	C	
20	2	0	

* so the cache placement for this sequence, with S=16, b=1 is

tag	index (data)	offset (none)
1	4	
1	8	
1	C	

5B) S=8, b=1, 2-way set associative

number of sets (S = B/N) = (8 = S = 8/2 = 4)

since b=1, then no offset bit needed

it's 2-way set, so 4/2 = 2 bits for set index

* so set tag (6 bits) + set index (2 bits)

for the question, the least significant bit are 0, 4, 8, C (hex)

* we have

Way1		Way2	
index	tag	index	tag

* The first address is 40, in binary is 0|00 0000

44, 0|00 0|00

48, 0|10 1000

4C, 0|00 1100

* since the index set is 2 bits, the least significant bit for 0, 4, 8, C is 00, then access of only on index 00.

address

For example, 40   way1   way2   44

index   tag   tag

insert

00   0|00 00   0|00 0|

  0|00 10   0|00 11

48   4C

replace   replace

* The next address is 70, 74, 78, 7C will replace the cache as above example, then 80, 84, 88, 8C replace the same location as well.

* After all the replacement, the last two addresses left is 1C, 20 which will be in the cache as:

	way1	way2
index 00	001000	000111
	(1C)	(20)

5.2) S=8, b=2, $\log_2(8) = 3$ bits for index.

b=2, so offset = 1

* so cache looks tag(4 bits), index(3 bits), (offset 1 bit)
* addresses ending all with 0, 4, 8, C, the last bit is always 0.

	binary	omit offset bit	transfer	binary(index)
0	0000	000	⟹	0
4	0100	010		2
8	1000	100		4
C	1100	110		6

* for address 40, 44, 48, 4C

so index	Tag
0	40
1	
2	44
3	
4	48
5	
6	4C
7	

* for address 70, 74, 78, 7C

index	Tag
0	70
1	
2	74
3	
4	78
5	
6	7C
7	

* for address 80, 84, 88, 8c,

index	tag
0	80
1	
2	84
3	
4	88
5	
6	8c
7	

* for 90, 94, 98, 9c ... :

0 4 8 c

will be the same procedure

* for 10 14 18 1C ,

index	tag
0	10
1	
2	14
3	
4	18
5	
6	1C
7	

* last for 20.

index	tag
0	20
1	
2	14
3	
4	18
5	
6	1C

(final table)