

Logic Design

Final Exam

2018. 12. 3

ID :

Name :

Problem 1 (K-map, 5pt)

The logic under has 4-input and 1-output named A, B, C, D and F respectively. Find minimum Sum of Product form of output F.

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	X
0	0	1	1	1
0	1	0	0	X
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	X
1	1	0	1	0
1	1	1	0	0
1	1	1	1	X

Answer)

		C, D			
		00	01	11	10
A, B	00	1	0	1	X
	01	X	0	1	1
	11	X	0	X	0
	10	1	0	1	1

$\overline{B}\overline{D} + \overline{A}C + \overline{B}C$

Problem 2 (QM method, 10pt)

Use the Quine-McCluskey method to find the minimum sum-of-products form for the following Karnaugh map.

	A				
	1	0	0	1	
	×	0	0	0	D
	×	1	1	0	
C	1	1	0	×	
	B				

	A				
	0	4	12	8	
	1	5	13	9	D
	3	7	15	11	
C	2	6	14	10	
	B				

ref)

2.1 Show your process of deriving the prime implicants

2.2 Select a minimum set (minimum # of literal) of prime implicants

ANS)

2.1.

col1		col2		col3	
0	0000	0,2	00-0	0,2,8,10	-0-0
1	0001	0,8	-000	0,1,2,3	00--
2	0010	0,1	000-	2,3,6,7	0-1-
8	1000	2,6	0-10		
3	0011	2,3	001-		
6	0110	2,10	-010		
10	1010	8,10	10-0		
7	0111	1,3	00-1		
15	1111	6,7	011-		
		3,7	0-11		
		7,15	-111		

PI: BCD, A'B', A'C, B'D'

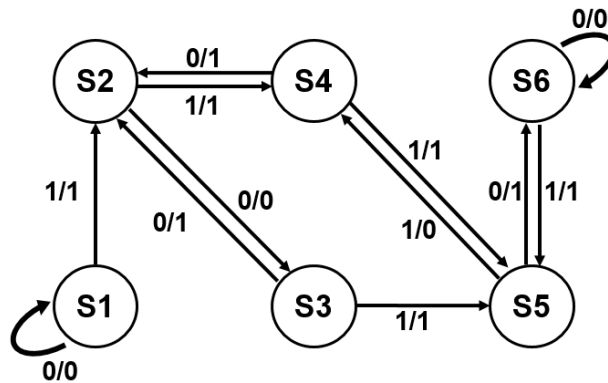
2.2.

			0	2	6	7	8	15
0,2,8,10	-0-0	B'D'	X	X			X	
0,1,2,3	00--	A'B'	X	X				
2,3,6,7	0-1-	A'C		X	X	X		
7,15	-111	BCD				X		X

Minimum literal set of PI: BCD + A'C + B'D'

Problem 3. (State minimization, 10pt)

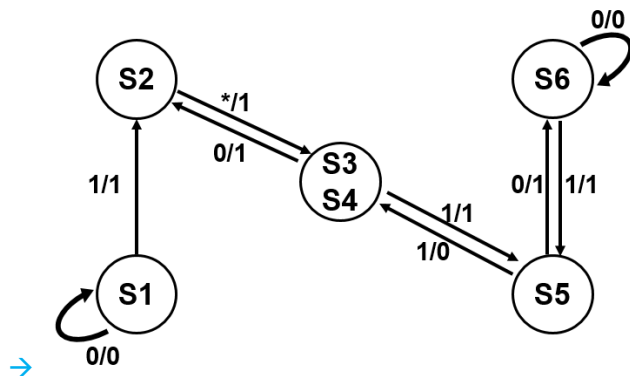
Given the state diagram in figure below, obtain an equivalent reduced-state diagram containing a minimum number of states. Optimize the number of states in implication chart method. Put your final answer in the form of a state diagram rather than a state table. Make it clear which states have been combined.



ANS)

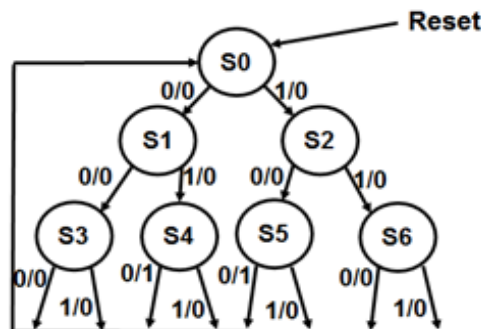
2	1-3 2-4				
3	1-2 2-5	3-2 4-5			
4	1-2 2-5	3-2 4-5	2-2 5-5		
5	1-6 2-4	3-6 4-4	2-6 5-4	2-6 5-4	
6	1-6 2-5	3-6 4-5	2-6 5-5	2-6 5-5	6-6 4-5
	1	2	3	4	5

→ S3 and S4 can be combined



Problem 4 (Sequential Logic design, 35pt)

Design a digital lock which takes 3 serial bits as input and gives an output '1' when one of the patterns 010 and 100 is entered as the input. After receiving the 3 serial bits as input and giving the corresponding output (1 for unlock and 0 for lock), the lock will be reset. Assume that the following state transition diagram is given.



- 4.1. Draw a state transition table for the given state diagram
- 4.2. Minimize the number of states with implication chart method
- 4.3. Draw a state diagram after state minimization
- 4.4. Draw a state transition table of the new state diagram in 4.3.
- 4.5. Draw K-maps for the output and state bits for the state transition table in 4.4.
- 4.6. Obtain the two-level SoP (sum-of-products) logic representation which minimize the number of literals
- 4.7. Draw a circuit schematics of the entire sequential circuit for this digital lock system ONLY with D flip-flops, NAND and INV gates

ANS)

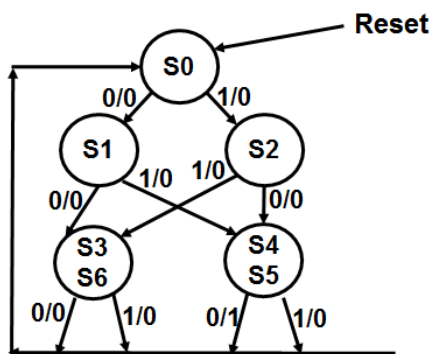
(1)

Input Sequence	Present State	Next State		Output	
		X=0	X=1	X=0	X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	1	0
11	S6	S0	S0	0	0

(2)

S1						
S2						
S3						
S4						
S5					S0-S0 S0-S0	
S6				S0-S0 S0-S0		
	S0	S1	S2	S3	S4	S5

(3)



(4)

Input Sequence	Present State	Next State		Output	
		X=0	X=1	X=0	X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S4	S3	0	0
00 or 11	S3	S0	S0	0	0
01 or 10	S4	S0	S0	1	0

Input Sequence	Present State	Next State		Output	
		X=0	X=1	X=0	X=1
Reset	000	001	010	0	0
0	001	011	100	0	0
1	010	100	011	0	0
00 or 11	011	000	000	0	0
01 or 10	100	000	000	1	0

(5)

Present State			X	Next State			OUTPUT
P2	P1	P0		N2	N1	N0	
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0
1	0	1	X	X	X	X	X
1	1	X	X	X	X	X	X

N2

P0X P2P1	00	01	11	10
00	0	0	1	0
01	1	0	0	0
11	X	X	X	X
10	0	0	X	X

N1

P2P1 P0X	00	01	11	10
00	0	1	0	1
01	0	1	0	0
11	X	X	X	X
10	0	0	X	X

N0

P2P1 P0X	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	X	X	X	X
10	0	0	X	X

OUTPUT

P2P1 P0X	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	0	X	X

(6)

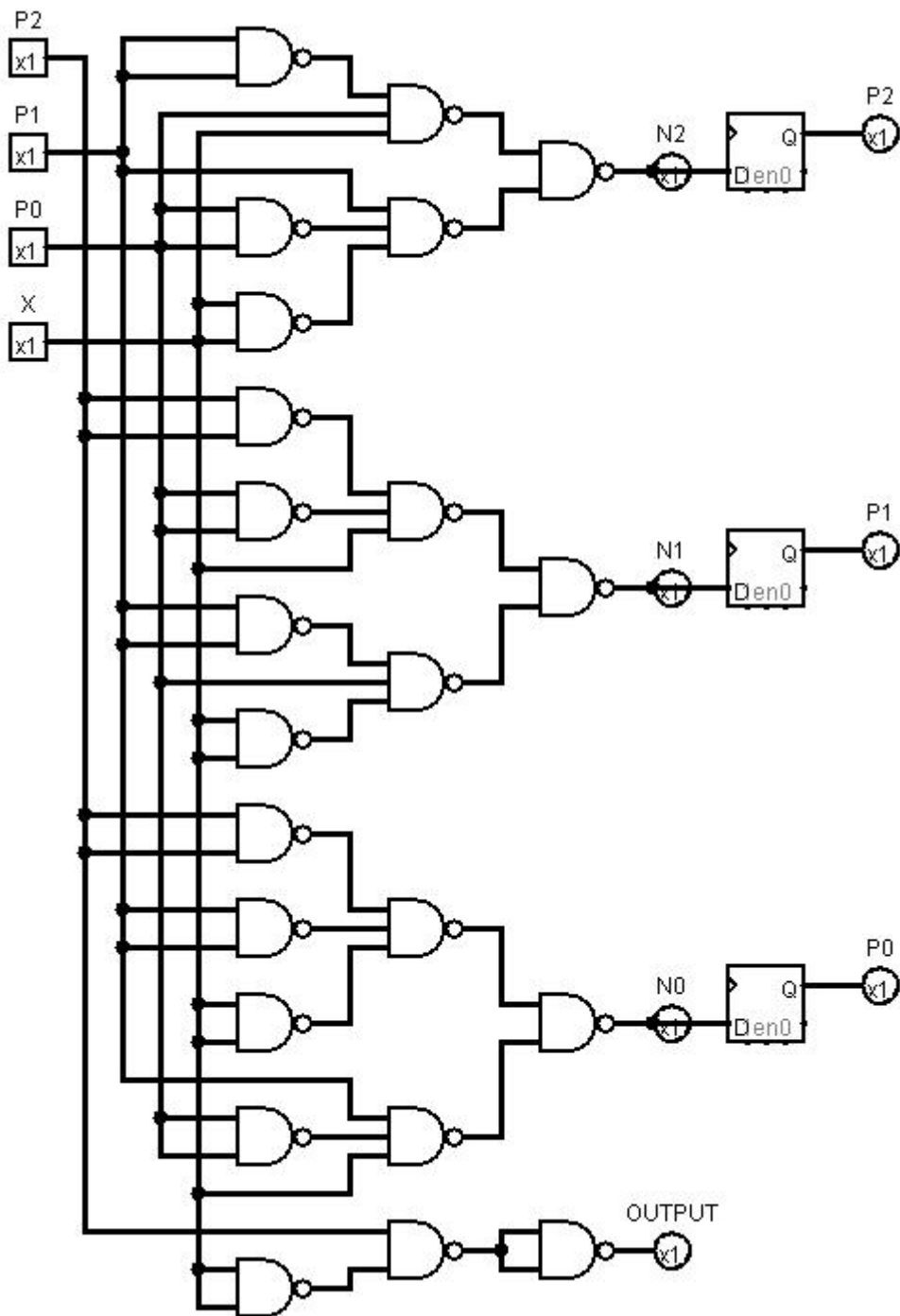
$$N2 = P1'P0X + P1P0'X'$$

$$N1 = P2'P0'X + P1'P0X'$$

$$N0 = P2'P1'X' + P1P0'X$$

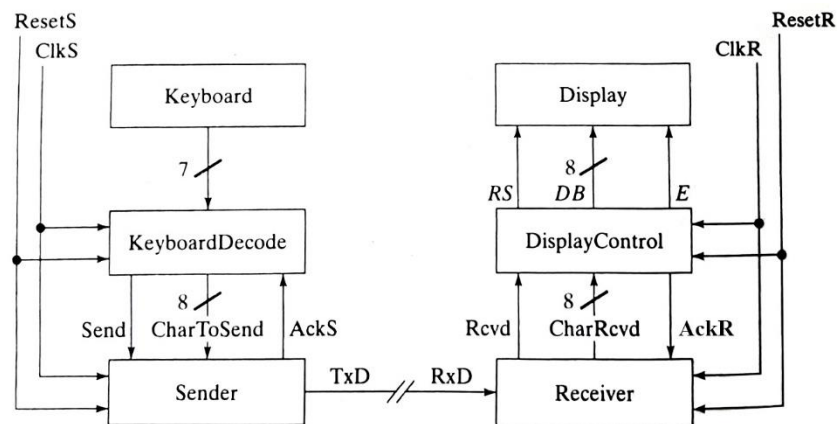
$$OUTPUT = P2X'$$

(7)



Problem 5 (Design Problem, 15pt)

The following figure is block diagram of serial line transmitter and receiver (RS232 protocol). The data will be sent serially, and the circuit works when the clocks on the two ends are completely asynchronous.



Suppose we are going to transmit one byte of data (**8'b1101_0011**) using RS232 protocol.

Draw timing diagram of signals on sender side (**TxD**, **AckS**, **bitCounter**) and receiver side (**CharRcvd**, **Rcvd**, **bitCounter**, **cycleCounter**).

■ Sender module

```
module Sender (ClkS, ResetS, Send, CharToSend, AckS, TxD);

input    ClkS, ResetS, Send;
input    [7:0] CharToSend;
output    AckS, TxD;

reg      TxD;           // register alias for RS232 line
reg [7:0] charToSend; // holds character to be sent over
                        // the serial line
reg [3:0] bitCounter; // keeps count of which bit is
                        // being sent
reg      go;

always @(posedge ClkS) begin
    if (ResetS) begin TxD <= 1; go <= 0; end
    else begin
        if (Send) begin
            // load character sent from keypad and hold
            // it just in case
            charToSend <= CharToSend;
            bitCounter <= 0; go <= 1;
        end
        else begin
            bitCounter <= bitCounter + 1;
            if (go) begin
                if (bitCounter == 0) TxD <= 0;
                else if (bitCounter > 0 && bitCounter <= 8)
                    TxD <= charToSend[8 - bitCounter];
                else if (bitCounter > 8) begin
                    TxD <= 1;
                    go <= 0;
                end
            end
        end
    end
end
end
end
assign AckS = go;
endmodule
```

ClkS

TxD

0

ch[7]

[6]

[5]

[4]

[3]

[2]

[1]

[0]

1

AckS

1

1

1

1

1

1

1

1

1

0

Bc

0

1

2

3

4

5

6

7

8

9

빨간색은 starting/finish signal이므로 필수

■ Receiver module

```
module Receiver (ClkR, ResetR, RxD, AckR, CharRcvd, Rcvd);
input  ClkR, ResetR, RxD, AckR;
output [7:0] CharRcvd;
output  Rcvd;


reg [4:0] bitCounter;           // keep count of number of bits received
reg [1:0] cycleCounter;        // used to divide input clock by 4
reg [8:0] characterReceived;    // start bit plus 8 data bits
reg      go;                   // flag indicating a character is arriving

// finite state machine to receive characters
always @(posedge ClkR) begin

    if (ResetR) begin go <= 0; Rcvd <= 0; end
    else begin
        if (!go && !Rcvd && !AckR && RxD) ;
        if (!go && !Rcvd && !AckR && !RxD) begin
            go <=1; bitCounter <= 0; cycleCounter <= 0;
        end
        if (go) begin // every four cycles . . .
            if (cycleCounter == 0) begin
                characterReceived[8 - bitCounter] <= RxD;
                if (bitCounter < 8)
                    bitCounter <= bitCounter + 1;
            else begin
                go <= 0;
                Rcvd <= 1;
            end
        end
        cycleCounter <= cycleCounter + 1;
    end
    if (AckR) Rcvd <= 0;
end

assign CharRcvd[7:0] = characterReceived[7:0];

endmodule
```

ClkR										
RxD	10000	ch[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]	1111
RxD가 1이다가 0으로 떨어지는 순간 receiving이 시작됨										
bitcnt	0	1111	2222	3333	4444	5555	6666	7777	8888	9999
Cy.Cnt	0123	0123	0123	0123	0123	0123	0123	0123	0123	0123
c.Rcv	[8]	[8:7]	[8:6]	[8:5]	[8:4]	[8:3]	[8:2]	[8:1]	[8:0]	
characterReceived 신호, 이걸 안적어도 무방										
C.Rcv	xxxx	[7]	[7:6]	[7:5]	[7:4]	[7:3]	[7:2]	[7:1]	[7:0]	
c.Rcv, C.Rcv 두 개는 모두 위 범위 외 bit는 모두 X										
Go	1111	계속 1~~						~~110		
Rcvd	계속 0~~								~~01	

Problem 6 (Sequential Logic Design, 25pt)

Design an automobile surveillance system with the following specification:

- Input/Output

Inputs

- **User_lock**: signal for locking the door, set to high when the user locks the door
- **User_unlock**: signal for unlocking the door, set to high when the user unlocks the door
- **Trespass**: signal for trespassing, set to high when someone breaks into the car

Outputs

- **Light**: enables light alarm
- **Horn**: enables horn alarm

- The surveillance system uses an external timer

- Input/output for the external timer

Input

- **Call(N)**: input signal for starting the timer. Timer rings after N seconds. Any format of N is okay. (i.e. Call(1), Call(15), Call(99), etc.)

Output

- **Ring**: set to high when the timer expires (After N seconds)

- The surveillance system sequences the following stages:

OPEN → SET → WATCH → ALARM

- **OPEN**: Surveillance off mode when the doors are unlocked. Whenever **User_unlock** signal is set in any other stages, the system enters this stage. **Light** and **Horn** are both off.
- **SET**: Surveillance is activated by the user. The system makes a transition from **OPEN** stage to **SET** stage when **User_lock** signal is set. **Light** and **Horn** are set for 1 second to indicate the surveillance system is active. After 1 second, the system advances to **WATCH** stage.
- **WATCH**: Surveillance is active. When **Trespass** is set (someone breaks into the car), the state changes to **ALARM** stage.
- **ALARM**: **Light** and **Horn** are set to high for 60 seconds. Whenever **User_unlock** signal is asserted in this stage, **Light** and **Horn** are turned off and the state changes to **OPEN** stage. If **Light** and **Horn** are high for 60 seconds without any input, the state stays in **ALARM** stage with **Light** on, **Horn** off.

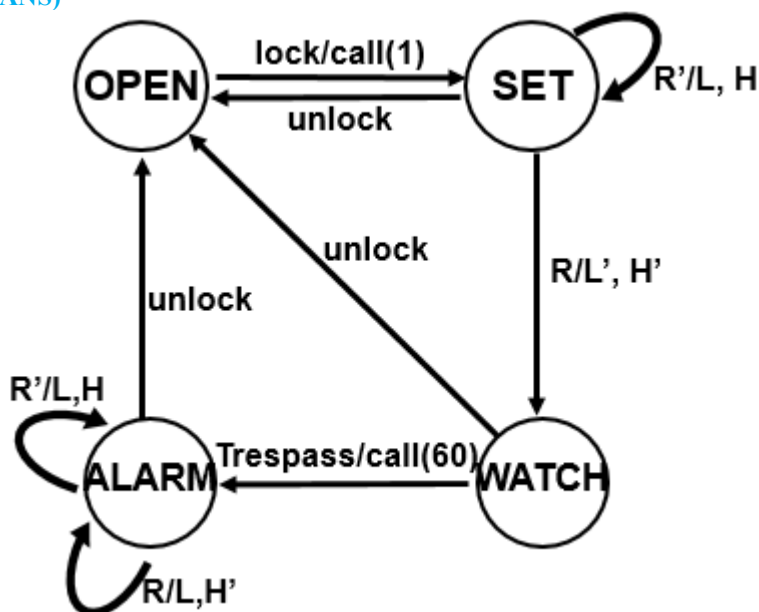
6.1. Draw a state transition diagram for the surveillance system

6.2. Draw a state transition table

6.3. Obtain the two-level SoP (sum-of-products) logic representation for next states and outputs (**Light**, **Horn**)

6.4. Draw a circuit schematic diagram for next states and outputs (**Light**, **Horn**)

ANS)



Present State	unlock	lock	R	T	Next State
OPEN		1			SET
SET	1				OPEN
SET			0		SET
SET			1		WATCH
WATCH	1				OPEN
WATCH				1	ALARM
ALARM	1				OPEN

Present State	unlock	lock	R	T	Next State
00		1			01
01	1				00
01			0		01
01			1		10
10	1				00
10				1	11
11	1				00

$$N1 = P1 U' + P0 U' R$$

$$N0 = P1' P0' + P0' U' + U' R'$$