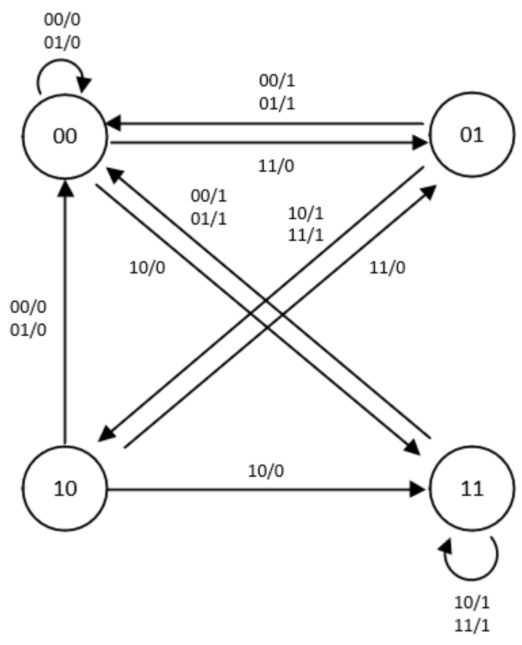
Final Exam Solution & Criteria

1. [Total 40 Pts.] Variants of the problems provided in the course materials.

(a) [10 Pts.] Consider a sequential circuit with two D flip-flops (*A* and *B*), two inputs (*x* and *y*), and one output (*z*), as specified by the following next-state and output equations: A(t+1) = xy’ + xB, B(t+1) = xA + xB’, z = B. Draw the corresponding state diagram.

Sol)

Present		Input		Next		Output
A	B	x	y	A	B	z
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1



Criteria)

- 10pt: Correct state diagram (both mealy and moore format can be accepted)
- 8pt: Correct state table, but incorrect state diagram
- 5pt: Correct state table, but empty state diagram
- 3pt: Incorrect both state table and state diagram
- 2pt: Don’t fulfill above all, but some proper explanations (ex. Logic diagram)

(b) [10 Pts.] Design a 3-bit synchronous counter which counts in the sequence (CBA): 001, 010, 011, 110, 111, 100, 101, (repeat) 001,… by using D flip-flops. You can use any logic gates if necessary.

Sol)

Present			Next			D_ff		
C	B	A	C	B	A	D_C	D_B	D_A
0	0	0	X	X	X	X	X	X
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	1	0	1	1	0
1	0	0	1	0	1	1	0	1
1	0	1	0	0	1	0	0	1
1	1	0	1	1	1	1	1	1
1	1	1	1	0	0	1	0	0

BA

C	**X**	0	**1**	0
	1	0	**1**	**1**

$D_C = CA' + BA$

BA

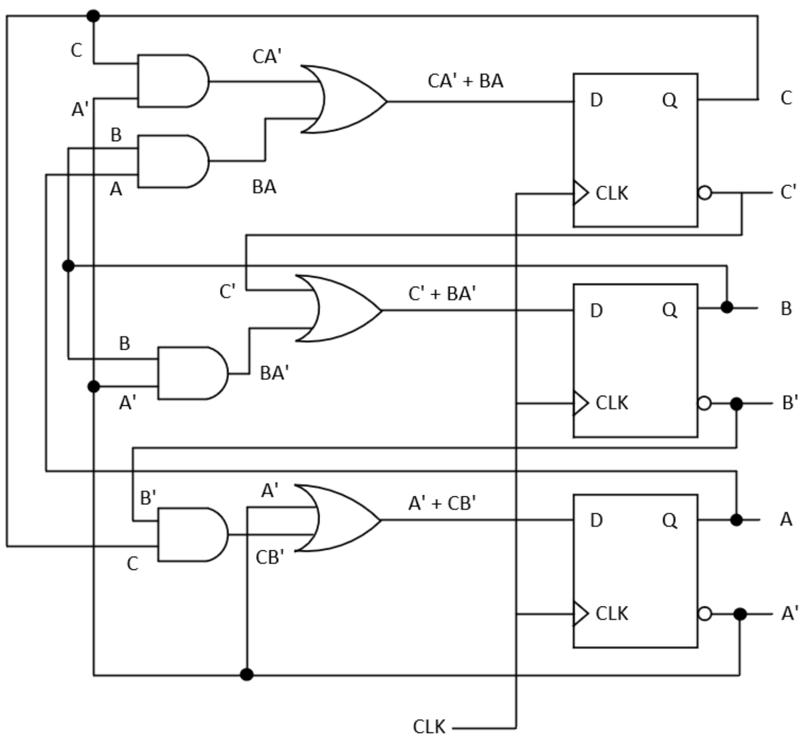
C	**X**	**1**	**1**	**1**
	0	0	0	**1**

$D_B = C' + BA'$

BA

C	**X**	0	0	**1**
	1	**1**	0	**1**

$D_A = A' + CB'$



Criteria)

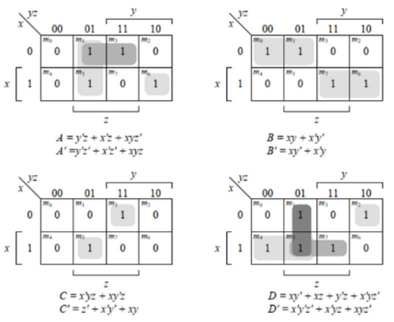
- State table (3pt), Logic expression (3pt), Logic diagram (4pt)
- State table: Correct state table (3pt), Mistake in state table (2pt),
- Misunderstanding of counter operation principle (0pt)
- Logic expression: 1pt of each C, B, A, total 3pt
- Logic diagram: Correct logic diagram (4pt), Incomplete or mistake in logic diagram (2pt)
- Incorrect state table, logic expression of previous solving process (0pt)

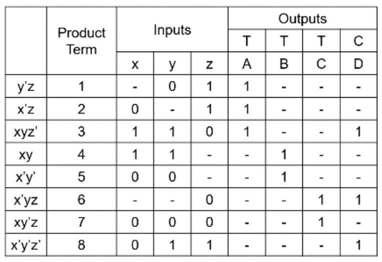
(c) [10 Pts.] Tabulate the PLA programming table for the four Boolean functions listed below. Minimize the numbers of product terms. Mark the fuse map.

$$A(x, y, z) = \Sigma(1, 3, 5, 6)$$
$$B(x, y, z) = \Sigma(0, 1, 6, 7)$$
$$C(x, y, z) = \Sigma(3, 5)$$
$$D(x, y, z) = \Sigma(1, 2, 4, 5, 7)$$

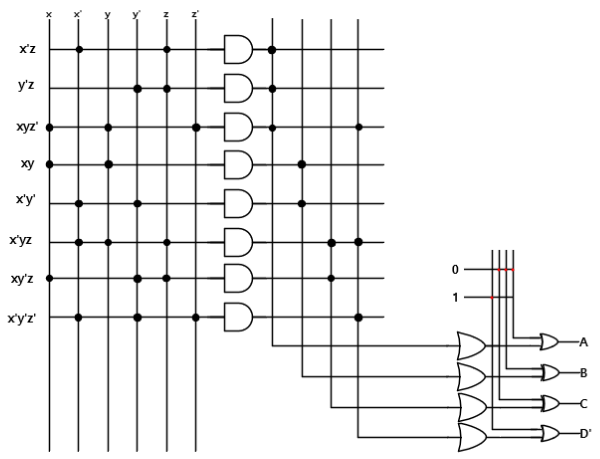
Sol)

$$A(x, y, z) = \Sigma(1, 3, 5, 6)$$
$$B(x, y, z) = \Sigma(0, 1, 6, 7)$$
$$C(x, y, z) = \Sigma(3, 5)$$
$$D(x, y, z) = \Sigma(1, 2, 4, 5, 7)$$





	Product Term	Inputs			Outputs			
					T	T	T	C
		x	y	z	A	B	C	D
y'z	1	-	0	1	1	-	-	-
x'z	2	0	-	1	1	-	-	-
xyz'	3	1	1	0	1	-	-	1
xy	4	1	1	-	-	1	-	-
x'y'	5	0	0	-	-	1	-	-
x'yz	6	-	-	0	-	-	1	1
xy'z	7	0	0	0	-	-	1	-
x'y'z'	8	0	1	1	-	-	-	1

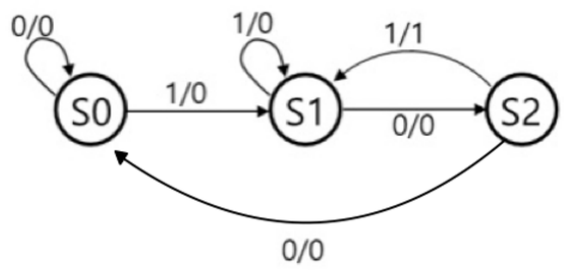


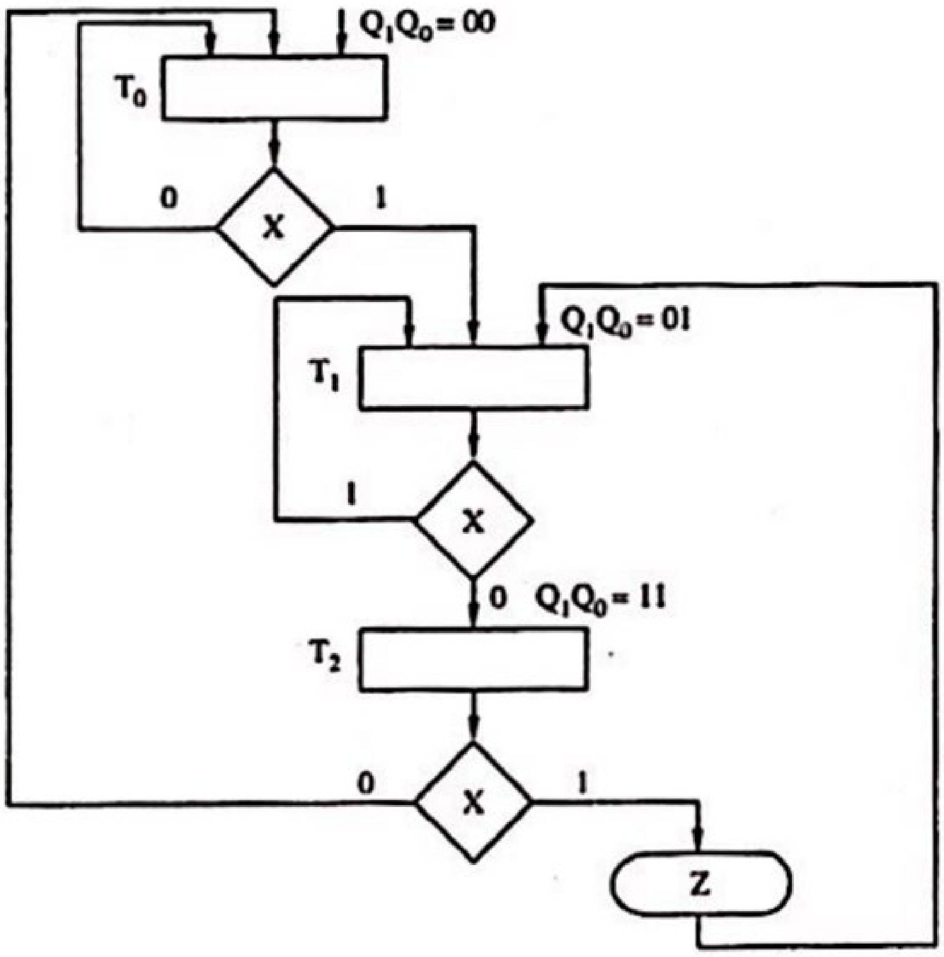
Criteria)

- + 10 point credit for PLA implementation below 8 terms
- -2 point deduction for using more than 9 terms
- -4 point deduction for PAL implementation (fixed OR implementation)
- -6 point for absence of any fuse map or wrong fuse map
- XOR or complement notification is both valid for full credit

(d) [10 Pts.] Draw an ASM chart and state diagram to describe a sequence detector that detects a sequence of 101.

Sol)





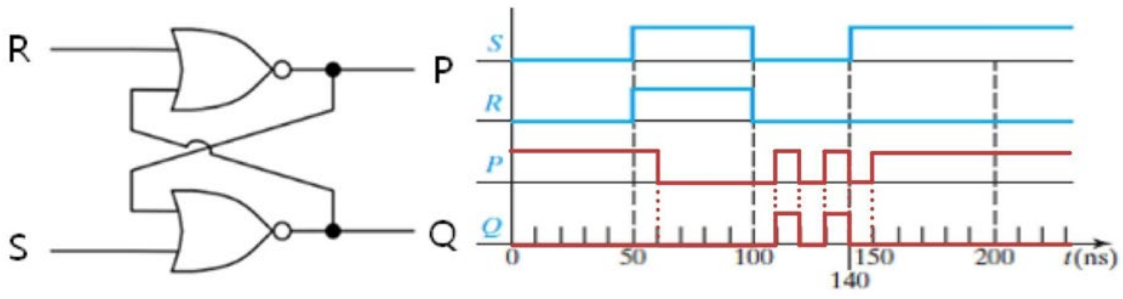
Criteria)

- Correct operation: +10
- Partially Correct Operation: - 5
- Correct operation, Missing ASM: - 5
- Correct operation, Missing State Diagram: - 5

2. [Total 20 Pts.] This problem illustrates the improper operation that can occur if both inputs to an S-R latch are 1 and are then changed back to 0. Complete the following timing chart, assuming that each NOR gate has a propagation delay of exactly 10 ns. Note that when t = 100 ns, S and R are both changed to 0. Indicate the times when P and Q are not complements of each other. Use an 'X' to indicate any times where the values are unknown.

(a) [10 Pts] Assume that initially P = 1 and Q = 0.

Sol)



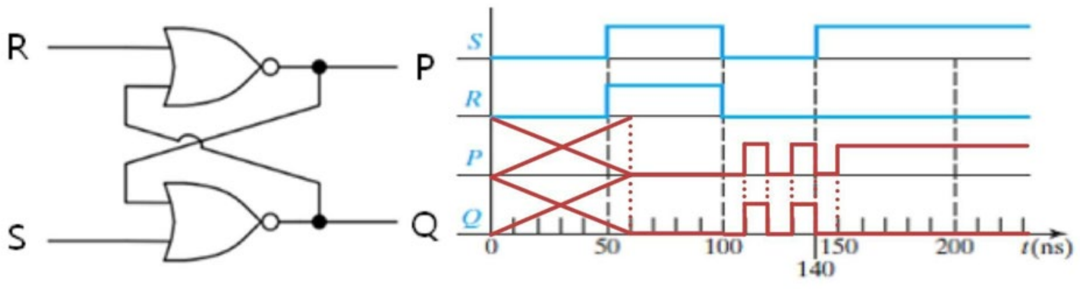
when P and Q are not complements of each other: 60ns ~ 150ns

Criteria)

- +4pts for correct timing diagram for P (+1pt for 0~110ns, +2pts for 110~150ns, +1pt for 150ns~)
- +4pts for correct timing diagram for Q (+1pt for 0~110ns, +2pts for 110~150ns, +1pt for 150ns~)
- +2pts for indicating the times when P and Q are not complements

(b) [10 Pts] Assume that initially P and Q are unknown.

Sol)



when P and Q are not complements of each other: 60ns ~ 150ns

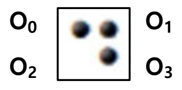
Criteria)

- +4pts for correct timing diagram for P (+1pt for 0~110ns, +2pts for 110~150ns, +1pt for 150ns~)
- +4pts for correct timing diagram for Q (+1pt for 0~110ns, +2pts for 110~150ns, +1pt for 150ns~)
- +2pts for indicating the times when P and Q are not complements

3. [Total 20 Pts.] Braille is a system of raised dots that can be read with the fingers by people who are blind or who have low vision. Design a system that converts binary numbers from 0 to 9 (input $I_3I_2I_1I_0$; I_3 is the MSB) to its corresponding Braille representation (output $O_3O_2O_1O_0$), considering the following table. Represent raise dots with '1' in the binary representation.

Braille	⠁	⠃	⠉	⠙	⠑	⠋	⠛	⠓	⠊	⠚
Number	1	2	3	4	5	6	7	8	9	0

Hint: Use the mapping shown below.



(a) [10 Pts.] Derive the most simplified expression for output O_i.

Sol)

The relationship between $I_3I_2I_1I_0$ and $O_3O_2O_1O_0$ is presented below.

Number	I_3	I_2	I_1	I_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

O_3	O_2	O_1	O_0
1	1	1	0
0	0	0	1
0	1	0	1
0	0	1	1
1	0	1	1
1	0	0	1
0	1	1	1
1	1	1	1
1	1	0	1
0	1	1	0

Simplified expression for output O_i is derived by using K-map.

1) O_3

I_3I_2 \ I_1I_0	**00**	**01**	**11**	**10**
00	1	0	0	0
01	1	1	1	0
11	X	X	X	X
10	1	0	X	X

$$\therefore O_3 = I_1'I_0' + I_2I_0$$

2) O_2

I_3I_2 \ I_1I_0	00	01	11	10
00	1	0	0	1
01	0	0	1	1
11	X	X	X	X
10	1	1	X	X

$$\therefore O_2 = I_2'I_0' + I_2I_1 + I_3$$

3) O_1

I_3I_2 \ I_1I_0	00	01	11	10
00	1	0	1	0
01	1	0	1	1
11	X	X	X	X
10	0	1	X	X

$$\therefore O_1 = I_3'I_1'I_0' + I_2I_0' + I_1I_0 + I_3I_0$$

4) O_0

I_3I_2 \ I_1I_0	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	0	X	X

$$\therefore O_0 = I_2 + I_1 + I_3'I_0 + I_3I_0'$$

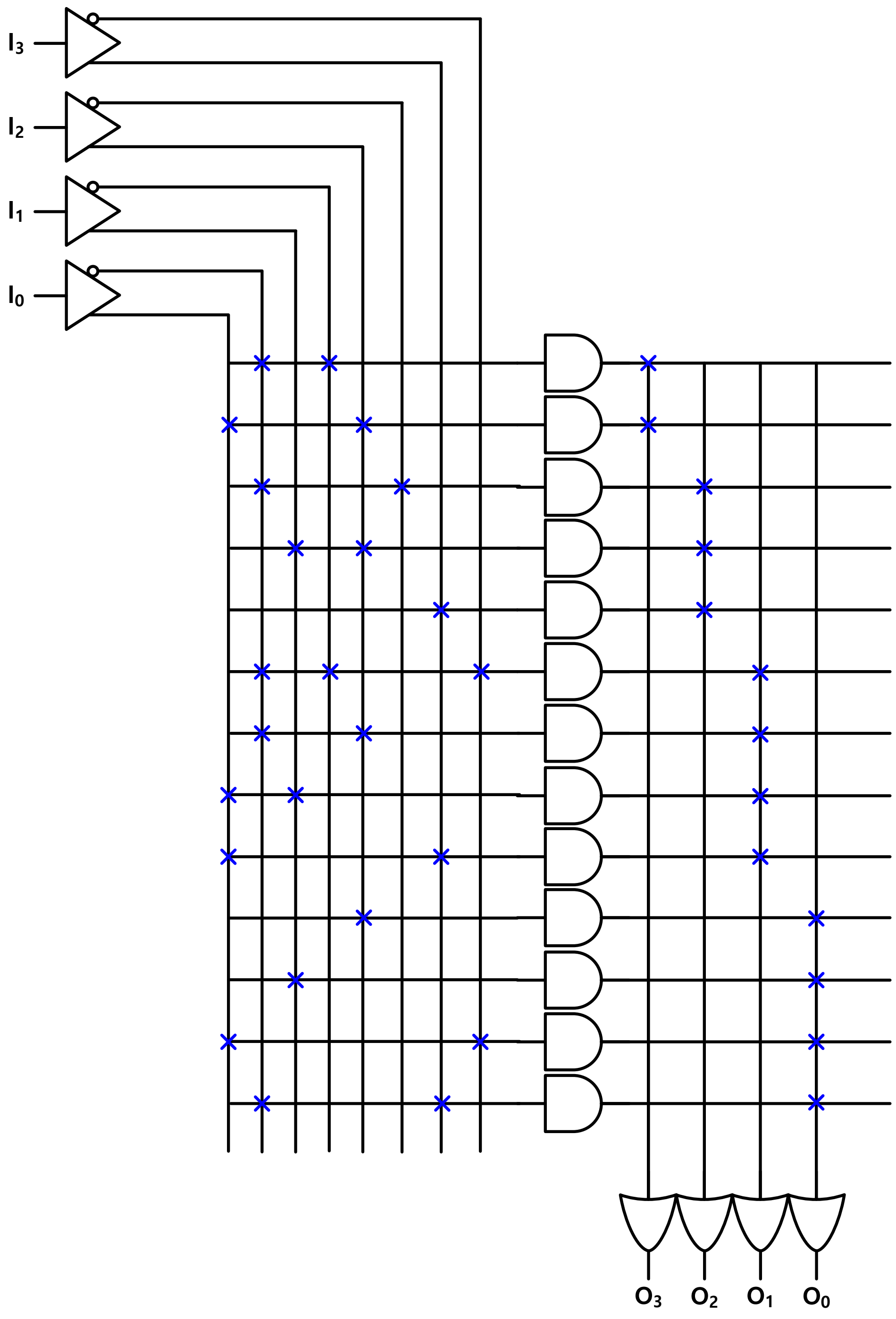
Criteria)

- 2.5 Pts for each expression (Oi, i=0,1,2,3) if you simplified the expression.
- 1 Pts for each expression (Oi, i=0,1,2,3) if you didn't simplify the expression but it's right.
- 0 Pts for each expression (Oi, i=0,1,2,3) if the answer is wrong.

(b) [10 Pts.] Implement this system by programmable logic array (PLA).

Sol)

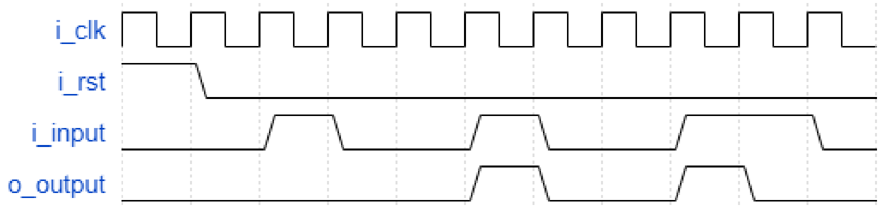
PLA can be implemented by using programmable AND array.



Criteria)

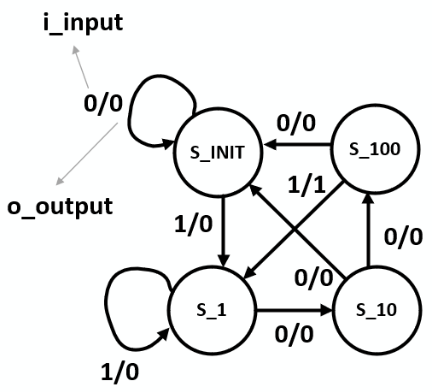
- 2.5 Pts for each expression (Oi, i=0,1,2,3) if you implemented the expression.
- 1 Pts for each expression (Oi, i=0,1,2,3) if you didn't implement the expression by using PLA but it's logically right.
- 0 Pts for each expression (Oi, i=0,1,2,3) if the answer is wrong.

4. [Total 25 Pts.] Design a binary pattern detector. The detector receives a 1-bit input every clock cycle and monitors the binary data received. At each clock cycle where the design detects the 4-bit binary pattern “1001”, the detector sets the 1-bit output HIGH. Note that the detector must be able to detect overlapping patterns as in the example below.



(a) [15 Pts.] Design a pattern detector using a finite state machine (FSM). Construct a state diagram for this circuit, and describe the definition of each state in the diagram. Use the minimum number of states possible for full credit.

Sol)



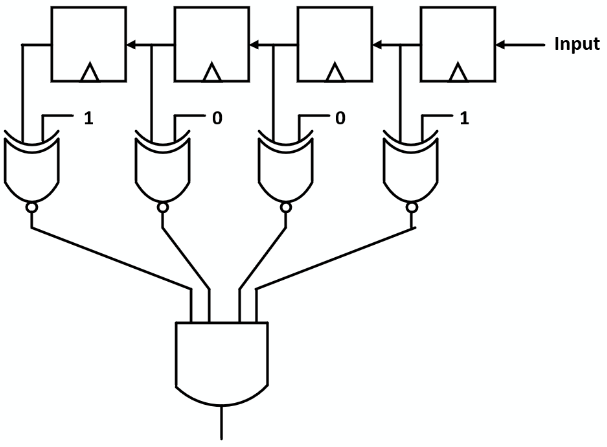
State	Description
S_INIT	Initial State Detected Nothing
S_1	Detected 1
S_10	Detected 10
S_100	Detected 100

Criteria)

- Not enough verbal descriptions (-2)
- 1001001 detect failure (-3)
- If FSM detects 1001, but contains critical failures(ex. Detects other inputs including 1001, stuck in certain state) (-7)
- No points if fails to detect anything
- 1 point deduction per minor mistakes
- If not minimum (-5)
- No points for non-FSM answers

(b) [10 Pts.] Design the pattern detector without using FSM. Provide the logic diagram and verbal descriptions for full credits. (**Note**: There may be many different solutions, but keep your architecture simple for full credit.)

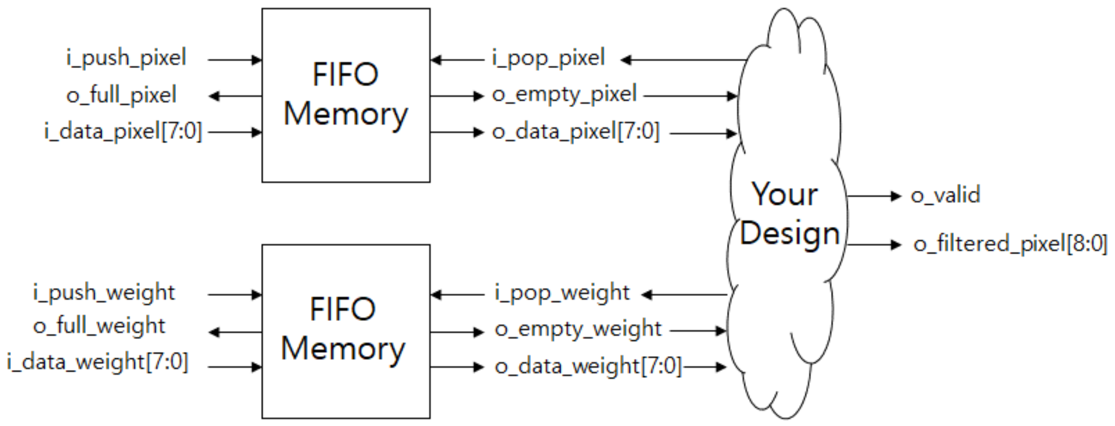
Sol)



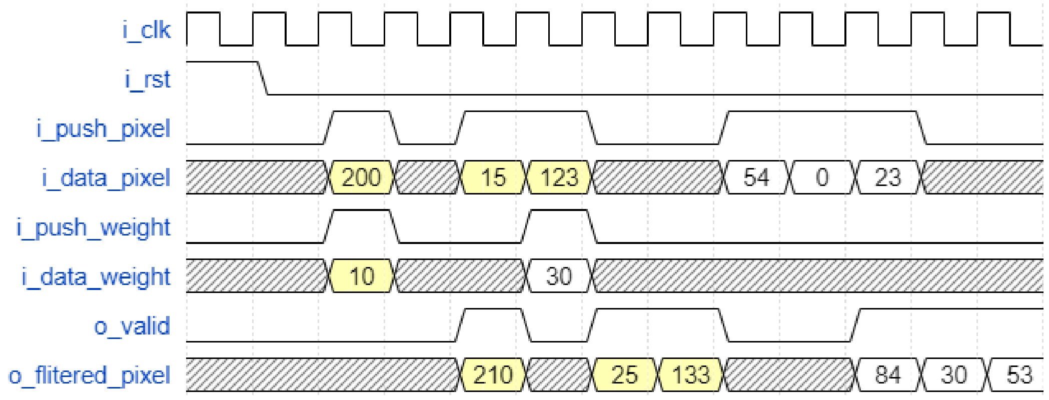
Criteria)

- Not enough verbal descriptions (-2)
- 1001001 detect failure (-3)
- If FSM detects 1001, but contains critical failures(ex. Detects other inputs including 1001, stuck in certain state) (-7)
- No points if fails to detect anything
- 1 point deduction per minor mistakes
- Description was logically right, but did not drew including logic gate/blocks(-7)

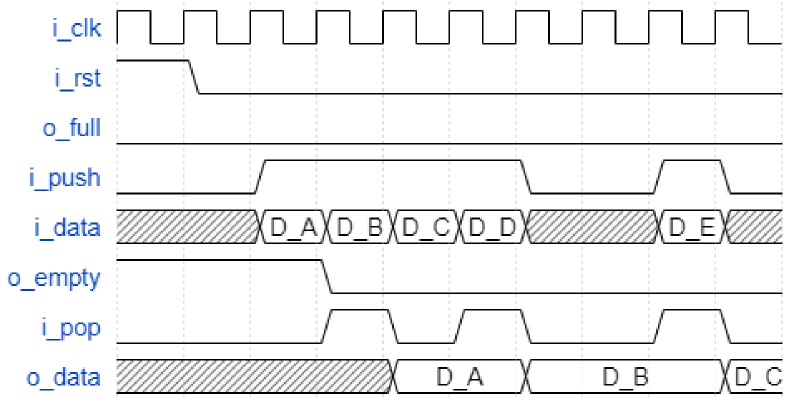
5. [Total 25 Pts.] Consider a digital image filtering system shown below. One input to the system is the image pixel values and the other input is the filter weight values. Assume both pixel and weight values are 8-bit binary numbers (i_data_pixel[7:0], i_data_weight[7:0]). Pixel and weight inputs may or may not be available on each clock cycle, so there are two FIFO memories that receive the two input values.



The system filters the image by adding weights to the pixels. However, each weight must be used 3 times on 3 consecutive pixels as shown in the timing diagram below. The output of the system is the filtered pixels (o_filtered_pixel[8:0]) and the valid signal (o_valid) which is set to HIGH whenever new filtered pixel values are available.



Hint: Timing diagram of the FIFO memory interface.



(a) [15 Pts.] Complete the system by designing "Your Design" part in the diagram above. Provide logic diagrams and verbal descriptions for full credits. (**Note**: There may be many different solutions, but keep your architecture simple for full credit.).

Sol)

There are four things that we have to make: i_pop_pixel, i_pop_weight, o_valid, o_filtered_pixel.

Since we have to use weight values 3 times, we can use a counter that counts from 0 to 2. If the counter value is 2, it means it is the third time that we used the weight, so new weight value should be popped. On the other hand, if the counter value is 0, it means it is the first time we are going to use the weight value.

The pixel should be popped if one of the two situations happen:

1. If we need new weight(counter value = 2) and o_empty_weight and o_empty_pixel are both 0
2. If we do not need new weight(counter value≠2) and o_empty_pixel is 0

Weight should be popped if:
We need new weight(counter value = 2) and o_empty_weight and o_empty_pixel are both 0
Which is the same condition as pixel pop situation 1.

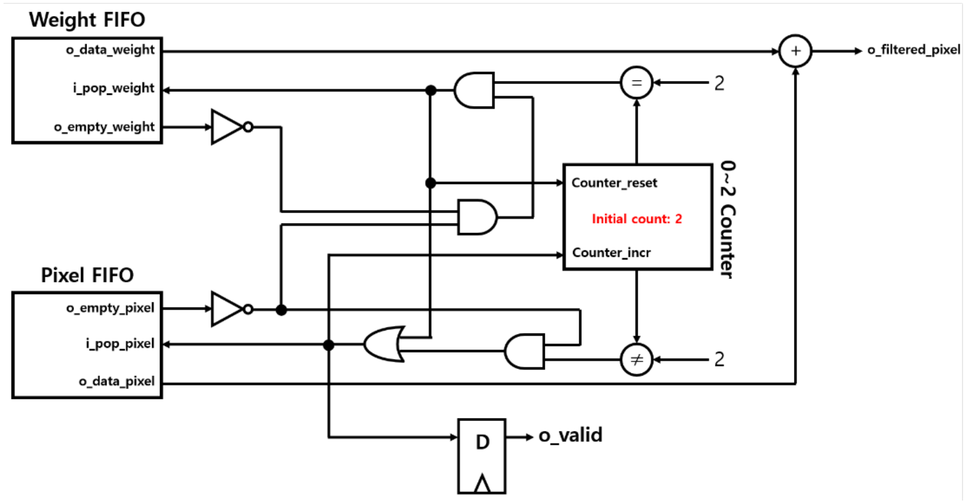
Counter value should be reset whenever new weight values are popped, and the value increases when new pixel values are popped.

Initially, we have to pop both weight and pixel if o_empty_weight and o_empty_pixel are both 0. So we set the initial counter value to 2, which can match the weight pop condition.

o_valid means that the o_filtered_pixel is actually a sum of o_data_pixel and o_data_weight. Thus, o_valid is always 1 if new pixel data is popped.

o_filtered_pixel is simply the sum of o_data_pixel and o_data_weight.

The overall logic diagram looks like below.



Criteria)
only verbal descriptions : maximum 5 pts
If no logic diagrams but ASM, state diagrams, etc : maximum 5pts
If only logic diagram : maximum 10pts
Minor errors : -1pts

reasonable methods to make i_pop_pixel (5pts)
- if only pop pixel when both FIFOs are not empty : 1pts
- additional reasonable logics to make i_pop_pixel, but not quite correct : 3pts

reasonable methods to make i_pop_weight (7pts)
- methods to use weight values for 3 times(ex. Counters, etc.) : 3pts
- additional reasonable logics to make i_pop_weight, but not quite correct : 2pts

reasonable methods to make o_valid (3pts)
- o_valid = o_empty_pixel' AND o_weight_pixel' : 1pts
- o_valid if whenever o_filtered_pixel is out : 1pts
- additional reasonable logics to make o_valid, but not quite correct : 2pts

(b) [10 Pts.] Below is a template design of the image filtering system described in SystemVerilog. Fill in the blank by describing the designs suggested above.
Sol)

```
module image_filter (
        input wire      i_clk,
        input wire      i_rst,
        output logic     o_full_pixel,
        input wire      i_push_pixel,
        input wire      [7:0] i_data_pixel,
        output logic     o_full_weight,
        input wire      i_push_weight,
        input wire      [7:0] i_data_weight,
        output logic     o_valid,
        output logic     [8:0] o_filtered_pixel
);

wire i_pop_pixel, o_empty_pixel;
wire [7:0] o_data_pixel;
wire i_pop_weight, o_empty_weight;
wire [7:0] o_data_weight;

fifo_memory u_pixel_fifo ( // Width 8-bit FIFO
   .Clk (i_clk),
   .Rst (i_Rst),
   .Full (o_full_pixel),
   .Push (i_push_pixel),
   .Push_data (i_data_pixel),
   .Empty (o_empty_pixel),
    .Pop (i_pop_pixel),
    .Pop_data (o_data_pixel)
);

fifo_memory u_weight_fifo ( // Width 8-bit FIFO
    .Clk (i_clk),
    .Rst (i_Rst),
    .Full (o_full_weight),
    .Push (i_push_weight),
    .Push_data (i_data_weight),
    .Empty (o_empty_weight),
    .Pop (i_pop_weight),
    .Pop_data (o_data_weight)
);
// Your design starts here…
logic [1:0] cnt;
logic cnt_rst, cnt_incr, cnt_is_2, cnt_isnot_0;

always_ff @(posedge i_clk) begin
            if (i_rst) begin
                        cnt <= 2;
            end else if (cnt_rst) begin
                        cnt <= 0;
            end else if (cnt_incr) begin
                        cnt <= cnt + 1;
            end

            i_pop_pixel <= cnt_incr;
            i_pop_weight <= cnt_rst;
            o_valid <= cnt_incr;
            o_filtered_pixel <= o_data_pixel + o_data_weight;
end

always_comb begin
            cnt_is_2 = (cnt == 2);
            cnt_isnot_0 = (cnt != 0);
            cnt_rst = (!o_empty_weight & !o_empty_pixel) & cnt_is_2;
            cnt_incr = (cnt_rst | (!o_empty_pixel & cnt_isnot_0))
end
// Your design ends here…
endmodule
```

Criteria)

5-(b)

i_pop_pixel (3)

- i_push_pixel related to i_pop_pixel : 0pts

i_pop_weight (5)

- i_push_weight related to i_pop_weight : 0pts
- if counter reset and i_pop_weight happens at the same time, but not correct full

answer : 4pts
- counter increment : 1pts
- counter reset : 1pts

o_valid (2)
- o_valid = o_empty_pixel' AND o_weight_pixel' : 1pts

o_filtered_weight (1)

6. [Extra point; 5 Pts.] *Questions for Questions.* Propose one question you would give in an exam if you were the instructor of this course. Your question can be any format (multiple-choice questions, T/F, short-answer questions, brief descriptive questions, etc.) and should reflect any term/concept/issue that you think are important in RTL design or ASM. Present the answer in detail.
(**Note:** 5 extra points will be given to students who created good questions but the total score in the final exam cannot exceed 130 points specified on the first page of this exam sheet.)
Criteria)
- Creative problems : +3~5pts
- Ordinary problems : +2pts
- Insincere problems : +1pt
- Empty : 0pts
- no answers : -1pts