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

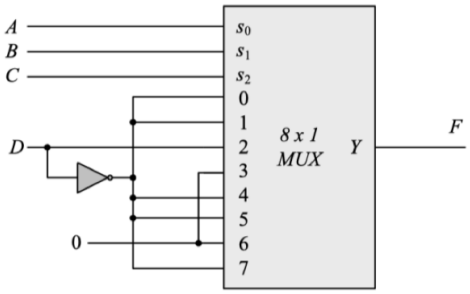
(a) [10 Pts.] Consider the Boolean function given below. Implement the function using the block diagram of a multiplexer. Use external gates if necessary.

$$F(A, B, C, D) = \sum m(0, 2, 5, 8, 10, 14).$$

Sol)

$F(A, B, C, D) = \sum m(0, 2, 5, 8, 10, 14)$

Inputs ABCD	Mux input line (ABC)	Value	$F = \Sigma(0, 2, 5, 8, 10, 14)$	
000 0	0	**0**	1	$F = D'$
000 1	0	1	0	
001 0	1	**2**	1	$F = D'$
001 1	1	3	0	
010 0	2	4	0	$F = D$
010 1	2	**5**	1	
011 0	3	6	0	$F = 0$
011 1	3	7	0	
100 0	4	**8**	1	$F = D'$
100 1	4	9	0	
101 0	5	**10**	1	$F = D'$
101 1	5	11	0	
110 0	6	12	0	$F = 0$
110 1	6	13	0	
111 0	7	**14**	1	$F = D'$
111 1	7	15	0	



Grading criteria

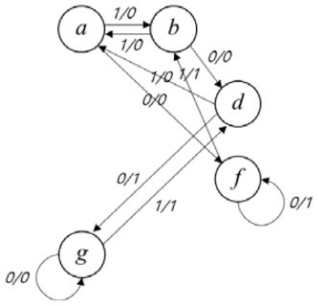
If your block diagram is right,
(+10pts) with proper answer using MUX without brute-force implementation.
(+5pts) with brute-force implementation using 16*1 MUX.
(-2pts) with implementing without not gate or omitting terminal. (A', B', C', D')
Else, (your block diagram is wrong)
(+5pts) with correct Boolean expression.

(b) [5 Pts.] For the following state table, provide the minimized state diagram.

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

Sol)

Present state	Next state 0	Next state 1	Output 0	Output 1
a	f	b	0	0
b	d	a	0	0
d	g	a	1	0
f	f	b	1	1
g	g	d	0	1



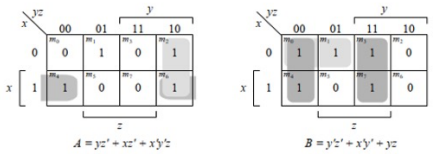
Grading criteria

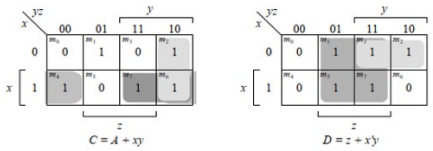
If your state diagram is right,
(+5pts) for right state diagram.
Else, (your block diagram is wrong)
(+2.5pts) for right state table.
No partial point for any wrong state table or diagram.

(c) [10 Pts.] Consider the truth table of a 3-input, 4-output combinational circuit. Provide i) the programmable array logic (PAL) programming table for the circuit (using K maps), and ii) the fuse map in a PAL diagram.

Inputs			Outputs			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	1	0	1	0	1	0
1	1	1	0	1	1	1

Sol)
i)

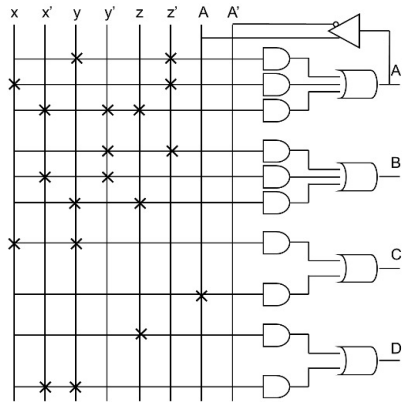




Product term	AND Inputs x	y	z	A	Outputs
1	-	1	0	-	A = yz' + xz' + x'y'z
2	1	-	0	-	
3	0	0	1	-	
4	-	0	0	-	B = y'z' + x'y' + yz
5	0	0	-	-	
6	-	1	1	-	
7	-	-	-	1	C = A + xy
8	1	1	-	-	
9	-	-	-	-	
10	-	-	1	-	D = z + x'y
11	0	1	-	-	
12	-	-	-	-	

$A = yz' + xz' + x'y'z$
$B = y'z' + x'y' + yz$
$C = A + xy$
$D = z + x'y$

ii)

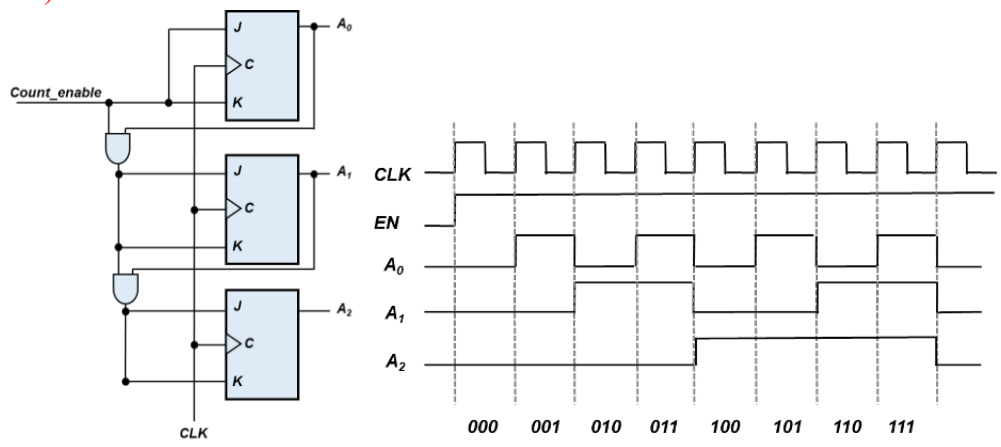


Grading criteria

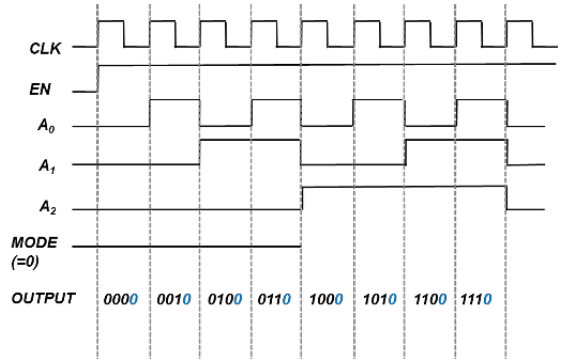
i) (+1pts) for each right Boolean expression or K-maps for A, B, C, D.
(+1pts) for right programmable array logic (PAL) programming table. (with 12 entries)
Total (+5pts), here.
ii) (+5pts) for right fuse map in a PAL diagram. (Wrong answer without AND gate)
(-2pts) if you don't exploit A signal for connecting C.

(d) [10 Pts.] Design a 3-bit up synchronous counters based on JK flip flops. Then, using the block diagram of the 3-bit counter, design a 4-bit counter that counts even (0-2-4-6-8-10-12-14-0…) and odd (1-3-5-7-9-11-13-15-1…) numbers when a mode control input, *M*, equals to 0 and 1, respectively.

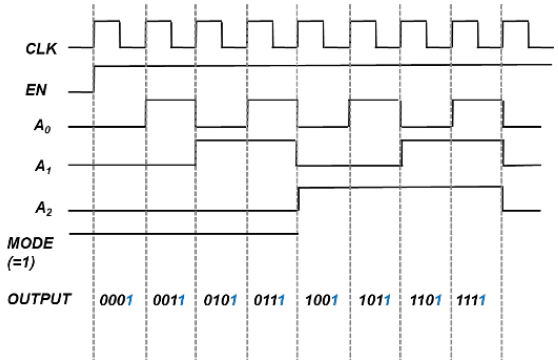
Sol)

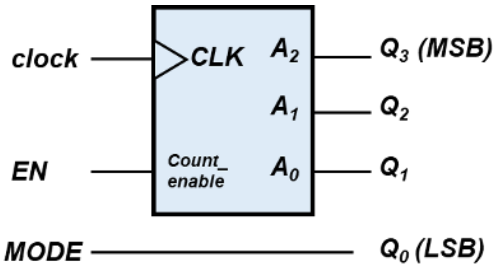


Even count mode (M = 0)



Odd count mode (M = 1)





Grading criteria

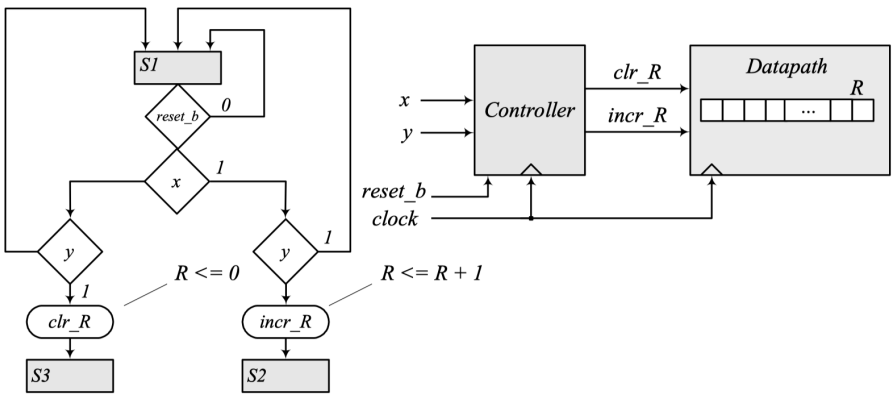
(+5pts) for right 3-bit up synchronous counters based on JK flip flop. (Should implement with external control signal like Count_enable)

(+5pts) for right 4-bit odd/even counter. (You could get score if you implemented with proper block diagram even though your 3-bit up synchronous counter doesn't get point)

No partial point for any wrong counter.

(e) [5 Pts.] A logic circuit with active-low synchronous reset has two control inputs x and y. If x is 1 and y is 0, register R is incremented by 1 and control goes to a second state. If x is 0 and y is 1, register R is cleared to zero and control goes from the initial state to a third state. Otherwise, control stays in the initial state. Draw an ASM chart starting from an initial state.

Sol)



Grading criteria

(+5pts) for right ASM chart starting from an initial state.

(-2.5pts) If you omit reset operation with 0. (Just giving reset as input is considered as wrong answer)

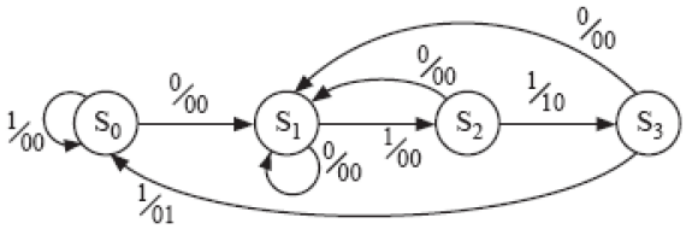
2. [Total 30 Pts.] Provide i) state definitions, ii) a state table and iii) a state diagram for the circuits below. Use the minimum number of states for full credit.

(a) [15 Pts.] Consider a sequential circuit with one input (X) and two outputs (Z_1 and Z_2). The circuit produces an output of $Z_1 = 1$ whenever the sequence 011 is received, and an output of $Z_2 = 1$ whenever the sequence 0111 is received.

Sol)

State	Meaning
S_0	Reset, 0111
S_1	0
S_2	01
S_3	011

State	Next State $X=0$	Next State $X=1$	Z_1Z_2 $X=0$	Z_1Z_2 $X=1$
S_0	S_1	S_0	00	00
S_1	S_1	S_2	00	00
S_2	S_1	S_3	00	10
S_3	S_1	S_0	00	01



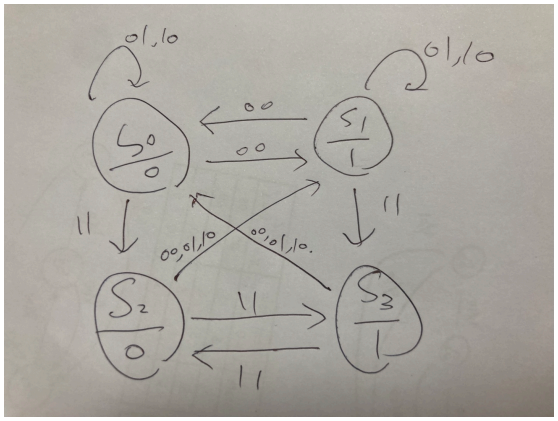
Grading Criteria

Each additional state -2

Each wrong input or output or state -2

(b) [15 Pts.] Consider a Moore sequential circuit that monitors two inputs X_1X_2. When the two inputs X_1X_2 are 00, the output Z toggles at every clock. When the two inputs X_1X_2 are 11, the output Z toggles at every other clock. When the two inputs X_1X_2 are different, the output Z holds its state and would not changes until the inputs are equal again. Assume that the initial state has Z = 0.

Sol)



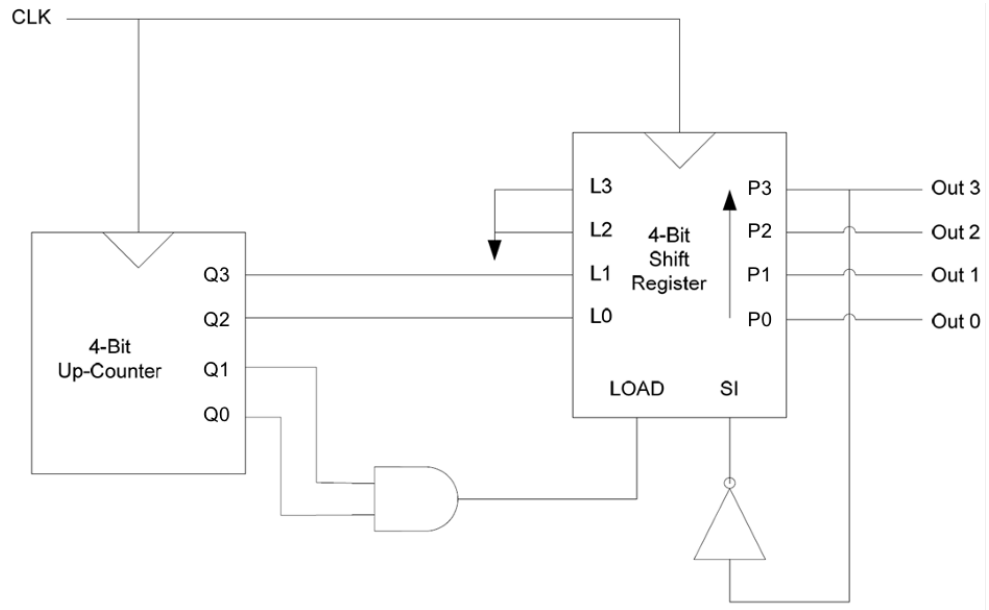
Grading Criteria

Not Moore machine -5 (e.g. Mealy machine)

Each additional state -1

Each wrong input or output or state -1

3. [Total 30 Pts.] Consider the design shown below. The output of the 4-bit shift register (Out 3 - Out 0; Out[3:0]) is the binary representation of a number Z between 0 and 15 (Out 3 is the MSB). The arrow drawn inside the block indicates the shift direction. LOAD and SI denotes a parallel load control input and a serial input, respectively.



(a) [15 Pts.] Assuming the 4-bit up-counter starts in state Q3Q2Q1Q0 = 0011, write the binary sequences of output (Out[3:0]) to repeat counting.

Sol)

Clock	Up-Counter	(Up 0 0 Q_3 Q_2)	(Up Q_1&Q_0)	Shift Register Out[3:0]	(Shift P_3')
	Q_3 Q_2 Q_1 Q_0	L_3 L_2 L_1 L_0	Load	P_3 P_2 P_1 P_0	SI
0~1	0 0 1 1	0 0 0 0	1	* * * *	*
1~2	0 1 0 0	0 0 0 1	0	0 0 0 0	1
2~3	0 1 0 1	0 0 0 1	0	0 0 0 1	1
3~4	0 1 1 0	0 0 0 1	0	0 0 1 1	1
4~5	0 1 1 1	0 0 0 1	1	0 1 1 1	1
5~6	1 0 0 0	0 0 1 0	0	0 0 1 1	1
6~7	1 0 0 1	0 0 1 0	0	0 1 1 1	1
7~8	1 0 1 0	0 0 1 0	0	1 1 1 1	1
8~9	1 0 1 1	0 0 1 0	1	0 0 1 0	0
9~10	1 1 0 0	0 0 1 1	0	0 1 0 1	1
10~11	1 1 0 1	0 0 1 1	0	1 0 1 1	1
11~12	1 1 1 0	0 0 1 1	0	0 1 1 0	0
12~13	1 1 1 1	0 0 1 1	1	0 0 1 1	1
13~14	0 0 0 0	0 0 0 0	0	0 1 1 1	1

14~15	0	0	0	1	0	0	0	0	0	1	1	1	1	1
15~16	0	0	1	0	0	0	0	0	0	1	1	1	0	0
16~17	0	0	1	1	0	0	0	0	1	1	1	1	0	0
17~18										0	0	0	0	

Counter					Shift Register						
Q3	Q2	Q1	Q0		LOAD	Out 3	Out 2	Out 1	Out 0	SI	OUT[3:0]
0	0	1	1		1	X	X	X	X	X	
0	1	0	0		0	0	0	0	0	1	0
0	1	0	1		0	0	0	0	1	1	1
0	1	1	0		0	0	0	1	1	1	3
0	1	1	1		1	0	1	1	1	1	7
1	0	0	0		0	0	0	0	1	1	1
1	0	0	1		0	0	0	1	1	1	3
1	0	1	0		0	0	1	1	1	1	7
1	0	1	1		1	1	1	1	1	0	15
1	1	0	0		0	0	0	1	0	1	2
1	1	0	1		0	0	1	0	1	1	5
1	1	1	0		0	1	0	1	1	0	11
1	1	1	1		1	0	1	1	0	0	6
0	0	0	0		0	0	0	1	1	1	3
0	0	0	1		0	0	1	1	1	1	7
0	0	1	0		0	1	1	1	1	0	15
0	0	1	1		1	1	1	1	0	0	14
0	1	0	0		0	0	0	0	0	1	0

(0000)→(0001)→(0011)→(0111)→(0001)→(0011)→(0111)→(1111)→(0010)→
(0101)→(1011)→(0110)→(0011)→(0111)→(1111)→(1110)→(0000)→(0001) …

Grading Criteria

- Correct full output values of 4-Bit shift register OUT[3:0] : 15 points
- Penalty for not obtaining full repeat counting : - 5 points
- No partial points

(a) [15 Pts.] Write the Verilog design for the 4-bit shift register.

Sol)

```
module shift_reg (
    input wire CLK,
    input wire LOAD,
    input wire SI,
    input wire [3:0] L,
    output logic [3:0] Q
);
always_ff @ (posedge CLK) begin
    if (LOAD) begin
        Q <= L;
    end else begin
        Q <= {Q[2:0], SI};
    end
end
endmodule
```

Grading Criteria

- 전체 시스템 (4-bit up counter + 4-bit shift register)에 대한 Verilog design을 하는 것이 아닌 문제의 4-bit shift register에 대한 Verilog design을 해야함.

- 4-bit shift register의 4개의 input (CLK, LOAD, SI, L[3:0])과 1개의 output (P[3:0]) 선언 : 총 5점 (각 1 점)

- 'LOAD=1'인 경우의 operation이 정확하게 구현 : 5 점

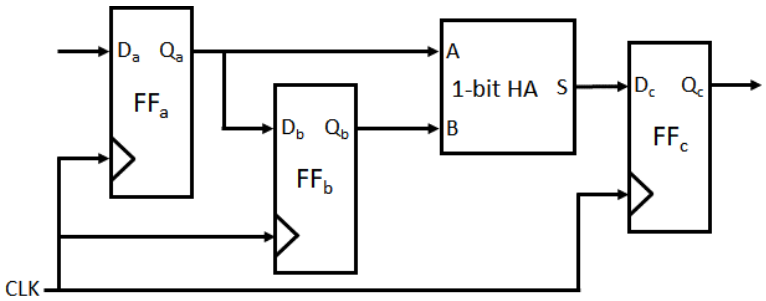
 L3=L2=0 으로 LOAD operation을 구현한 경우 -5 점

- 'LOAD=0'인 경우의 operation이 정확하게 구현 : 5 점

 SI를 외부 input을 받아 LSB에 직접 assign하는 것이 아닌 내부적으로 assign하여 구현하면 'LOAD=0'인 operation을 구현하지 않았다고 간주 (ex. SI = !P[3]) : - 5 점

- You have to design only 4-bit shift register in problem. Not whole system (4-bit up counter + 4-bit shift register)
- Correct declaration for 4 inputs (CLK, LOAD, SI, L) and 1 output (P) : total 5 points (each 1 point)
- Correct implementation of operation for 'LOAD = 1' : +5 points
 If you assume that L3=L2=0, you cannot get any score
- Correct implementation of operation for 'LOAD = 0' : +5 points
 If you assign SI internally (ex. input logic SI = !P[3]), it is considered that 'LOAD = 0' case is not implemented : - 5 points

4. [Total 30 Pts.] Consider the design shown below with the timing characteristics listed in the table. Assume that all three flip-flops (FFs) have the same timing characteristics and the 1-bit half adder (HA) is built using only 2-input NAND gates (no inverter gates). Ignore the carry out from the HA.



t_{ccq} (FF contamination delay clock-to-Q)	1ns
t_{pcq} (FF propagation delay clock-to-Q)	1ns
t_{setup} (FF setup time)	3ns
t_{hold} (FF hold time)	2ns
t_{cd} (NAND gate contamination delay)	1ns
t_{pd} (NAND gate propagation delay)	2ns

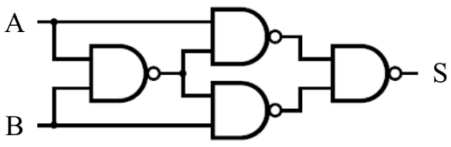
(a) [10 Pts.] Design the fastest 1-bit HA based on 2-input NAND gates (no inverter gates).

Sol)

Sum output of the 1-bit HA is defined as: S = A (XOR) B. The fastest XOR function can be implemented by 3 2-input NAND stages as shown below.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

S = AB' + A'B



Grading Criteria

Correct HA design and longest path is passing through 3 NAND gates → 10pts

No partial points

(b) [10 Pts.] Compute the maximum clock frequency of this circuit. Use the HA in **(a)**.

Sol)

$T_{a\text{-}b} = t_{pcq} + t_{setup} = 4ns$

$T_{a\text{-}c} = T_{b\text{-}c} = t_{pcq} + 3*t_{pd} + t_{setup} = 10ns$ → maximum delay → $f_{max} = 1/10ns = 0.1GHz$

The minimum clock period of the circuit is: $t_{pcq} + 3 \times t_{pd} + t_{setup} = 1 + 3 \times 2 + 3 = 10ns$.

Hence, the maximum clock frequency is 1/10ns = 100MHz. Note that the result is the same for either the path Q_a → A or Q_b → B.

Grading Criteria
- Correct delay → 5pts
- Correct frequency → 5pts

(c) [10 Pts.] Check for any path that violates the hold time constraint. If there is such a path, elaborate on which path violates the hold time constraint and the reason.

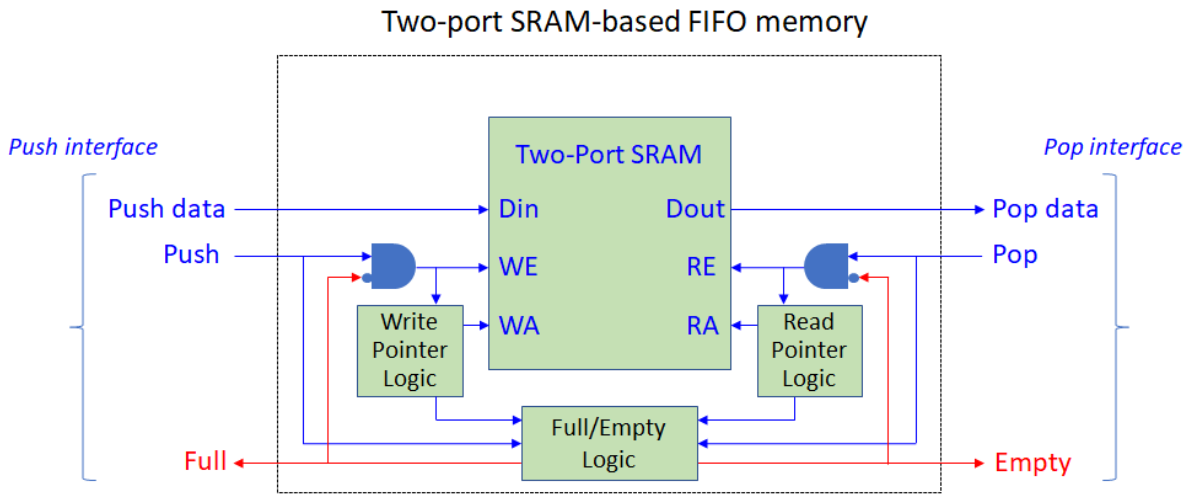
Sol)

FF_a to FF_b path: t_{ccq} (1ns) < t_{hold} (2ns) → violates the hold time constraint

The path from Q_a to D_b violates the hold time constraint of FF_b. Since there is no logic between Q_a and D_b, t_{ccq} (FF_a) < t_{hold}, violating the hold time constraint of FF_b. On the other hand, the hold time constraint of FF_c is not violated as t_{ccq} (FF_b) + 2 x t_{cd} > t_{hold}.

Grading Criteria
- Which path is hold violated → 5pts
- Explain correct reason → 5pts

5. [Total 30 Pts.] A two-port SRAM is used to build a typical FIFO memory to handle concurrent push and pop requests as shown below.



(a) [5 Pts.] Discuss pros and cons of a single-port SRAM and a two-port SRAM in detail. (Hint: You may consider logic complexity, flexibility, etc.)

Sol)

A two-port SRAM is better in terms of flexibility and system performance because it can support concurrent read and write accesses at the same clock cycle. On the other hand, a single-port SRAM can support either read or write access at a given cycle. However, in general, a two-port SRAM has higher logic complexity than a single-port SRAM. It is because a two-port SRAM must take two different addresses (read and write), requiring two sets of address decoders. An address decoder is used to select a word line based on a given address. A single-port SRAM can have only one address decoder for both read and write because there's only one address input shared by both operations.

Grading Criteria

Proper discussion about flexibility and complexity of two different SRAMs : 5pts

Discussion about one of flexibility and complexity : 3pts

Wrong discussion : 0pts

(b) [10 Pts.] We have assumed during the lesson that concurrent write and read accesses to a two-port SRAM cannot use the same address. Can a two-port SRAM-based FIFO memory design actually hold up to this assumption all the time? Choose an answer with yes or no and explain the reason in detail.

Sol)

The two-port SRAM write and read addresses are write and read pointers in the FIFO memory. When the write and read pointers are the same, it is either empty or full status. When empty, the FIFO cannot be popped, so there will be no SRAM read access. On the other hand, when full, the FIFO cannot be pushed, so there will be no SRAM write access. As a result, concurrent write and read accesses can never occur at the same

address of the two-port SRAM inside the FIFO memory.

Grading Criteria
Sufficient description about empty and full status : 10pts
Sufficient description about one of empty and full : 7pts
No description about empty and full status : 5pts
Insufficient description about empty and full status : 5pts
No or wrong description : 0pts
Minor errors : -1pts
Wrong answer : 0pts

(c) [15 Pts.] Design a FIFO memory using single-port SRAMs, not two-port SRAMs. The design should have exactly the same interface and operation as a two-port SRAM-based FIFO memory. Your design can contain several single-port SRAM instances. Describe the design architecture including a new logic/algorithm that differs from two-port SRAM-based FIFO memory. Include a description of how the FIFO's full and empty conditions are defined in your design.

(**Note**: There may be many different solutions, but keep your architecture simple for full credit. You may draw a logic diagram or provide verbal description without a diagram.)

Sol)
First, design a FIFO memory implemented using only one single-port SRAM, which does not allow the concurrent push and pop requests. Then, we will use two copies of such a single-port-SRAM-based FIFO. Let us call them FIFO0 and FIFO1.

In addition, we need a new control logic, deciding which one of the two FIFOs will be pushed / popped upon external push and pop requests. First, the logic should have a counter register, which keeps tracking of the total number of valid words in both FIFOs.

Second, the logic should have a shift register that records which FIFO holds each valid word. Let's assume the counter value is n, meaning there are total n valid words in both FIFOs. Then, whenever a new data is pushed to FIFO0 (or FIFO1), the n-th bit of the shift register must be set to 0 (or 1). In addition, the shift register shifts right whenever there's a pop request. Then, the LSB value of the shift register will always tell us which FIFO holds the oldest data (0 → FIFO0, 1 → FIFO1).

Now, when there's only a push request, the logic can choose either FIFO0 or FIFO1 to push the data as long as the FIFO is not full. When there's only a pop request, the logic must choose either FIFO0 or FIFO1 according to the shift register LSB value. When there are both push and pop requests, the logic must pop one of the FIFOs according to the shift register LSB value. At the same time, the incoming new data must be pushed to the other FIFO that is not chosen for the pop operation.

Thus, we can define the full condition of the entire FIFO memory. When the LSB of

the shift register is 0, the entire FIFO must claim it is full if FIFO1 is full. On the other hand, when the LSB of the shift register is 1, the entire FIFO must claim it is full if FIFO0 is full. Meanwhile, the empty condition of the entire FIFO memory is when both FIFO0 and FIFO1 are empty.

Grading Criteria

The key point of this problem is to make concurrent push/pop available.

Proper design that enables concurrent push/pop: 15pts
Major/minor issues lead to point deduction. Some frequently seen issues are :
1. Design that enables concurrent push/pop sometimes, but doesn't work in some situation : -5pts
2. Design that assumes certain situation such as clock cycle change, memory sharing between different SRAMs, etc. : -5pts
3. Design(or explanation) incomplete(because of lack of time or other causes), but can be partially understood would be graded based on the design quality.

Design that doesn't enable concurrent push/pop, but have explanation about using SRAM as a FIFO memory : 3pts

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 160 points specified on the first page of this exam sheet.)

Grading Criteria

- Creative problem: +3~5pts
- Ordinary problem: +2pts
- Insincere problem: +1pt
- Empty: 0pts