Logic Design Final Exam

2016. 12. 6

Total 53 pts

ID :

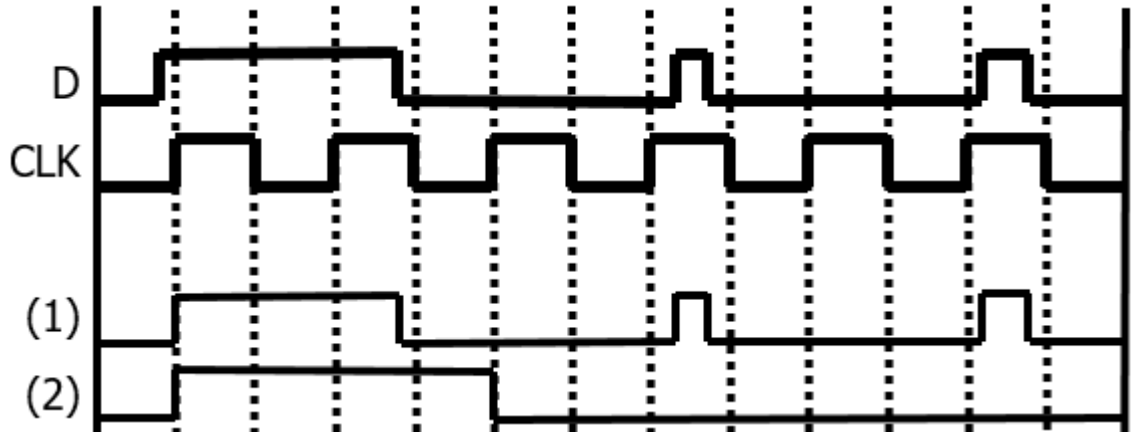
Name :

1. Timing diagram (8pt)

Given the input and clock transitions in the figure below, draw the output of each of the following devices (for the RS latch, input is asserted on S port).

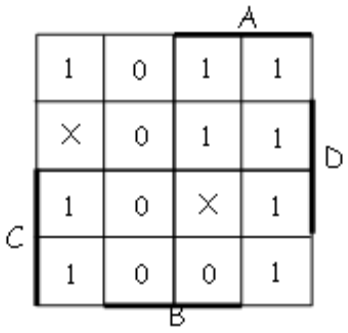
(1) Clocked RS latch
(2) Positive edge-triggered D flip-flop

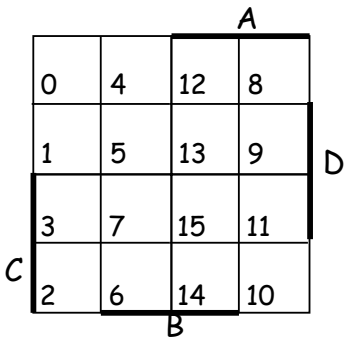
ANS)



2. QM method and hazard-free design (10pt)

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





ref)

(1) Show your process of deriving the prime implicants. (3pt)
(2) Select a minimum set of prime implicants. (3pt)
(3) Does the sum-of-product form obtained in prob (2) have static-1 hazard? If yes, find a hazard-free design. If no, prove it. (2pt)
(4) In synchronous circuit designs where all the setup and hold time constraints are met, the edge-triggered D flip-flops are free from hazards. Explain why. (2pt)

(Ans)

(1) Find all PIs

col1		col2		col3		col4	
0V	0000	(0 1)V	000-	(0 1 2 3)	00--	(0 1 2 3 8 9 10 11)	-0--
1V	0001	(0 2)V	00-0	(0 1 8 9)V	-00-		
2V	0010	(0 8)V	-000	(0 2 8 10)V	-0-0		
8V	1000	(1 3)V	00-1	(1 3 9 11)V	-0-1		
3V	0011	(1 9)V	-001	(2 3 10 11)V	-01-		
9V	1001	(2 3)V	001-	(8 9 10 11)V	10--		
10V	1010	(2 10)V	-010	(8 9 12 13)	1-0-		
12V	1100	(8 9)V	100-	(9 11 13 15)	1--1		
11V	1011	(8 10)V	10-0				
13V	1101	(8 12)V	1-00				
15V	1111	(3 11)V	-011				
		(9 11)V	10-1				
		(9 13)V	1-01				
		(10 11)V	101-				
		(12 13)V	110-				
		(11 15)V	1-11				
		(13 15)V	11-1				

PIs: B', AC', AD

(2) Select a minimum set of PIs by using a PI chart ignoring don't cares.

			0	2	3	8	9	10	11	12	13
(0,1,2,3,8,9,10,11)	-0--	B'	X	X	X	X	X	X	X		
(8,9,12,13)	1-0-	AC'				X	X			X	X
(9,11,13,15)	1--1	AD					X		X		X

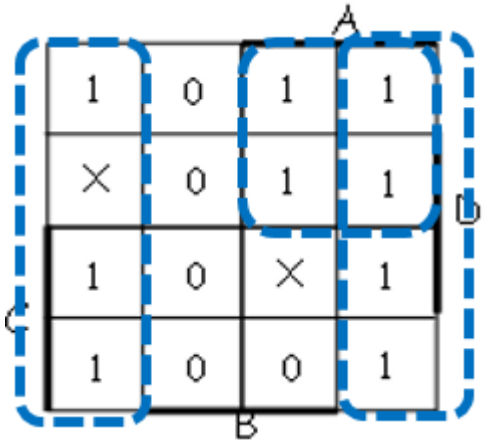
Minimum-literal SOP form:

B' + AC'

(3) (Ans)

Hazard-free

AC' + B'

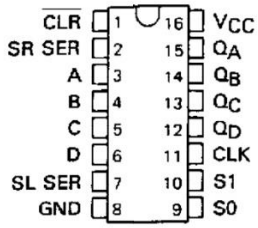


(4) (Ans) Any of the following two answers is accepted.

1: D flip-flops are edge-triggered. Thus, they do not suffer from glitches.

2: D flip-flops are free from 1s catching problem.

3. Data path and Control path of Shifter (10pt)
Ref) TTL 74194:



S1	S0	Action
0	0	Hold
0	1	Shift Right (SR SER to QA)
1	0	Shift Left (SL SER to QD)
1	1	Load

Shifters are normally used to shift data in a circular pattern (the data that shifts out at one end of the shifter is shifted back into the other end), or as a logic shift (filling the shifted positions with 0s) or an arithmetic shift (propagating the sign bit to the right or shift in 0s to the left).

(1) Show how to wire up a 4-bit universal shift register (TTL 74194) to perform the following shift operations. Draw a figure for each of shift operations (6pt)

(a) Circular shift right (e.g., 1110 becomes 0111)
(b) Circular shift left (e.g., 1110 becomes 1101)
(c) Logical shift right (e.g., 1110 becomes 0111)
(d) Logical shift left (e.g., 1110 becomes 1100)
(e) Arithmetic shift right (e.g., 1110 becomes 1111)
(f) Arithmetic shift left (e.g., 1110 becomes 1100)

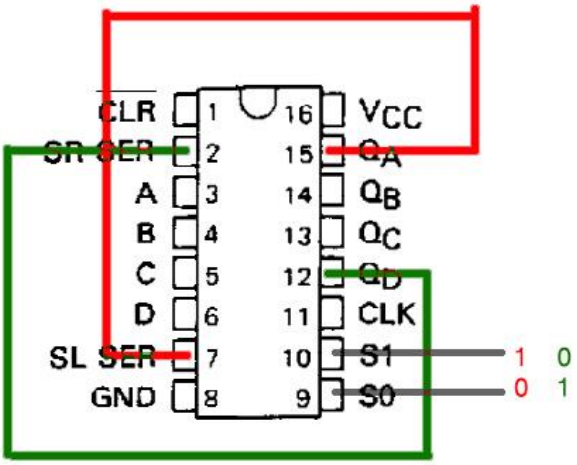
Now, your task is to design a shift-register subsystem based on the TTL 74194. There are 8 operations: 6 operations from problem 3.1, and two operations, Hold and Load. In order to select one of 8 operations, we need 3 control inputs C2, C1 and C0 that are interpreted as follows:

{C2, C1, C0}
000 is Hold (holding the current value)
001 is circular shift right
010 is circular shift left
011 is logical shift right
100 is logical shift left
101 is arithmetic shift right
110 is arithmetic shift left
111 is Load (loading a new 4-bit value).

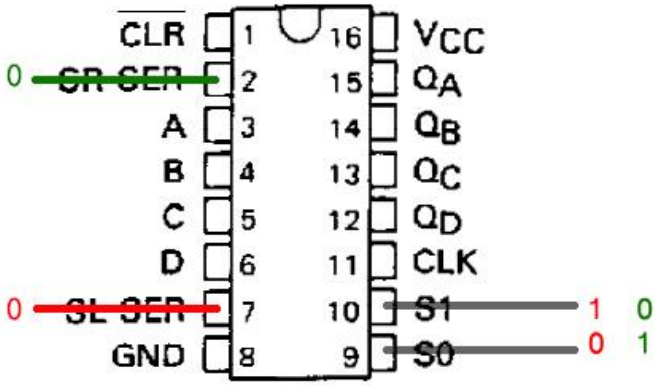
(2) Show the datapath for the shifter subsystem and combinational logic to decode the control inputs C2, C1, and C0 into S1 and S0 signals of TTL74194. (4pt)

ANS)
(1).

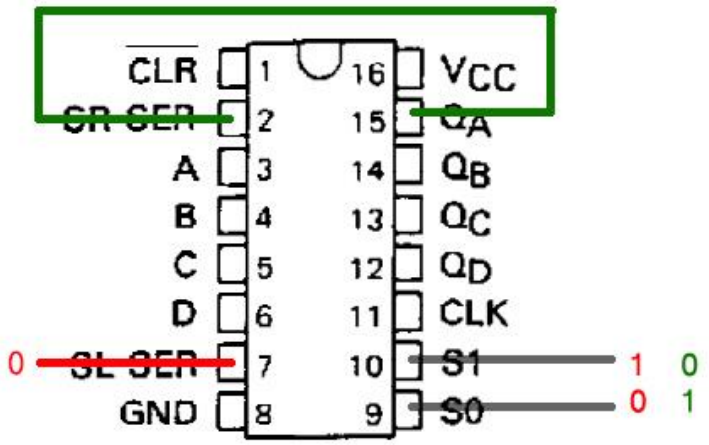
- Circular right
- Circular Left



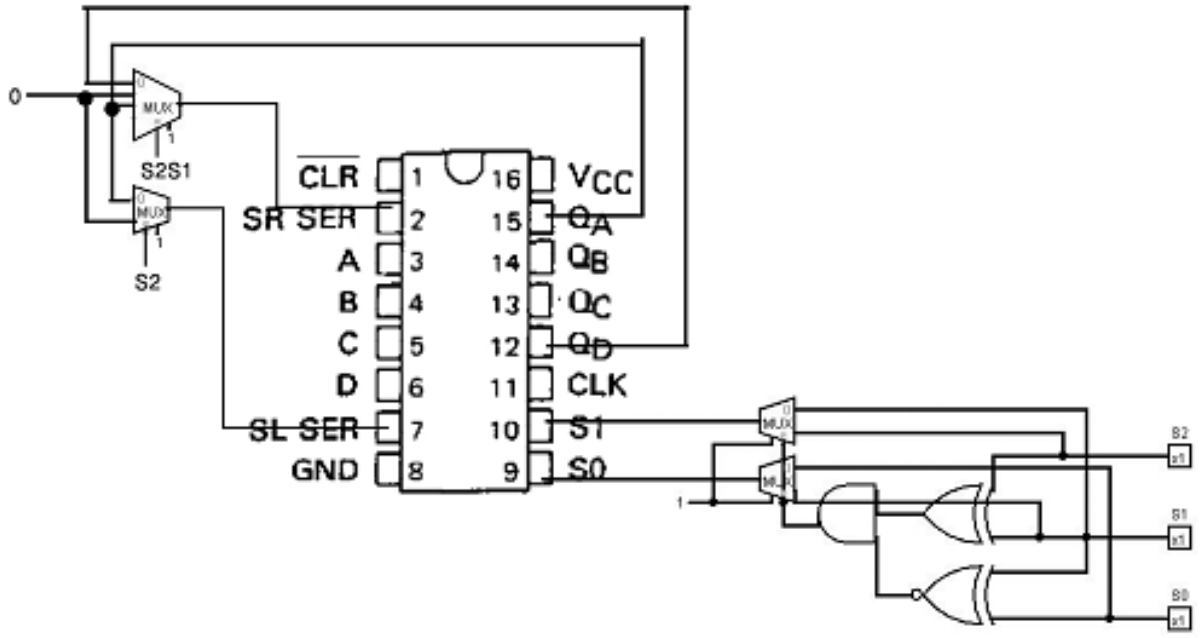
- Logic right
- Logic Left



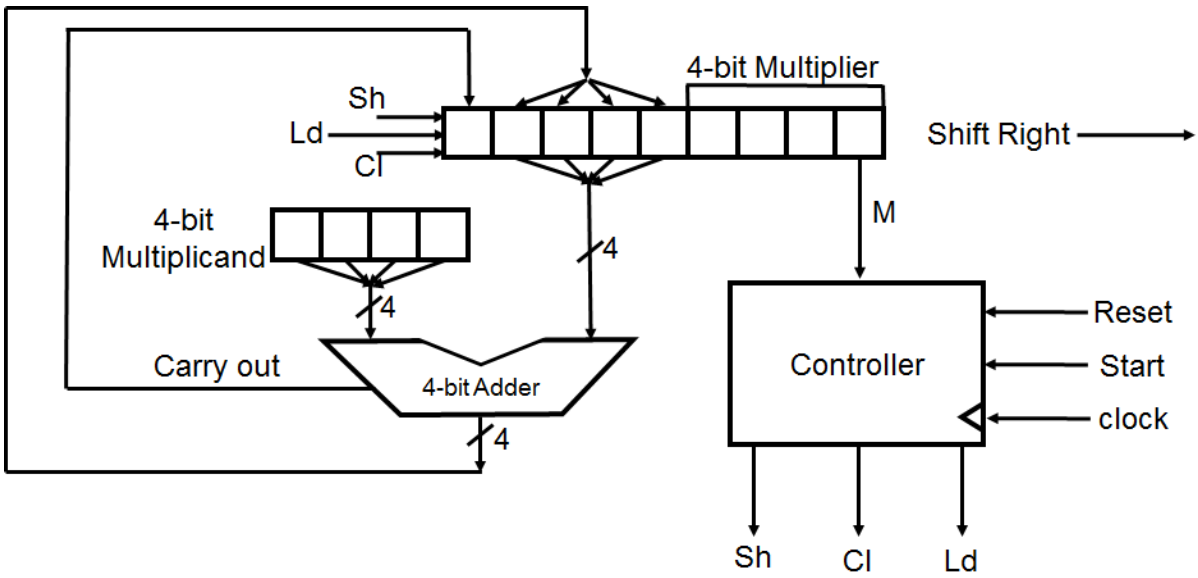
- Arithmetic right
- Arithmetic Left



(2).

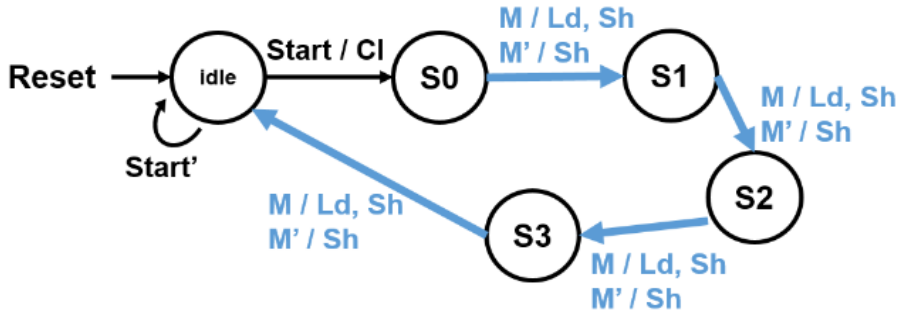


4. Sequential 4-bit multiplier (5pt)



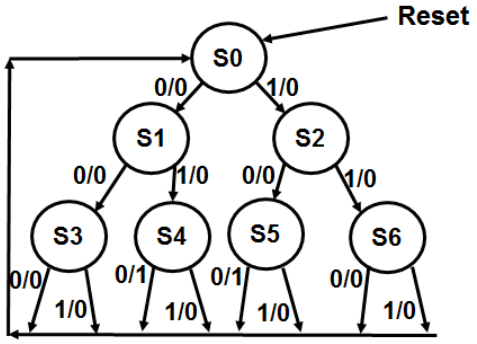
You are to design sequential 4-bit multiplier. In order to save registers, we only use two registers: one 4-bit register for multiplicand, one 9-bit register for multiplier and product. Controller is syncrhonized on global clock signal, and recieves 3 inputs: Start signal which starts multiplication operation, LSB of 9-bit register (M in the figure), and Reset. Controller generates 3 control signals: Sh for shifting 9-bit register right by 1 bit, Ld for loading the output of adder on the upper 5 bits of 9-bit register, and Cl for setting the upper 5 bits of 9-bit register with '0'.
Draw a state diagram for the controller of the sequential 4-bit multipier.

ANS)



5. Sequential Logic Design (20pt)

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.



(1) Draw a state transition table for the given state diagram (2pt)
(2) Reduce the number of states with implication chart method (5pt)
(3) Draw a state diagram after state reduction (2pt)
(4) Draw a state transition table of the new state diagram in (3) (2pt)
(5) Draw K-maps for the output and state bits for the table in (4) (2pt)
(6) Obtain the two-level SoP (sum-of-products) logic representations which minimize the number of literals (2pt)
(7) Draw a circuit schematics of the entire sequential circuit for this digital lock system ONLY with D flip-flops, NAND, and INV gates (5pt)

ANS)

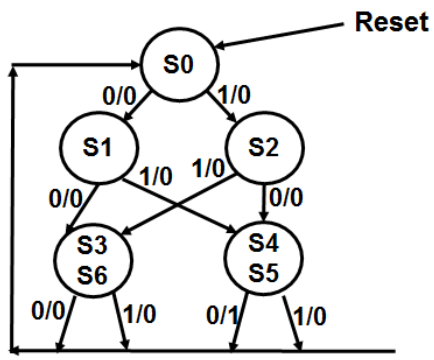
(1)

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)

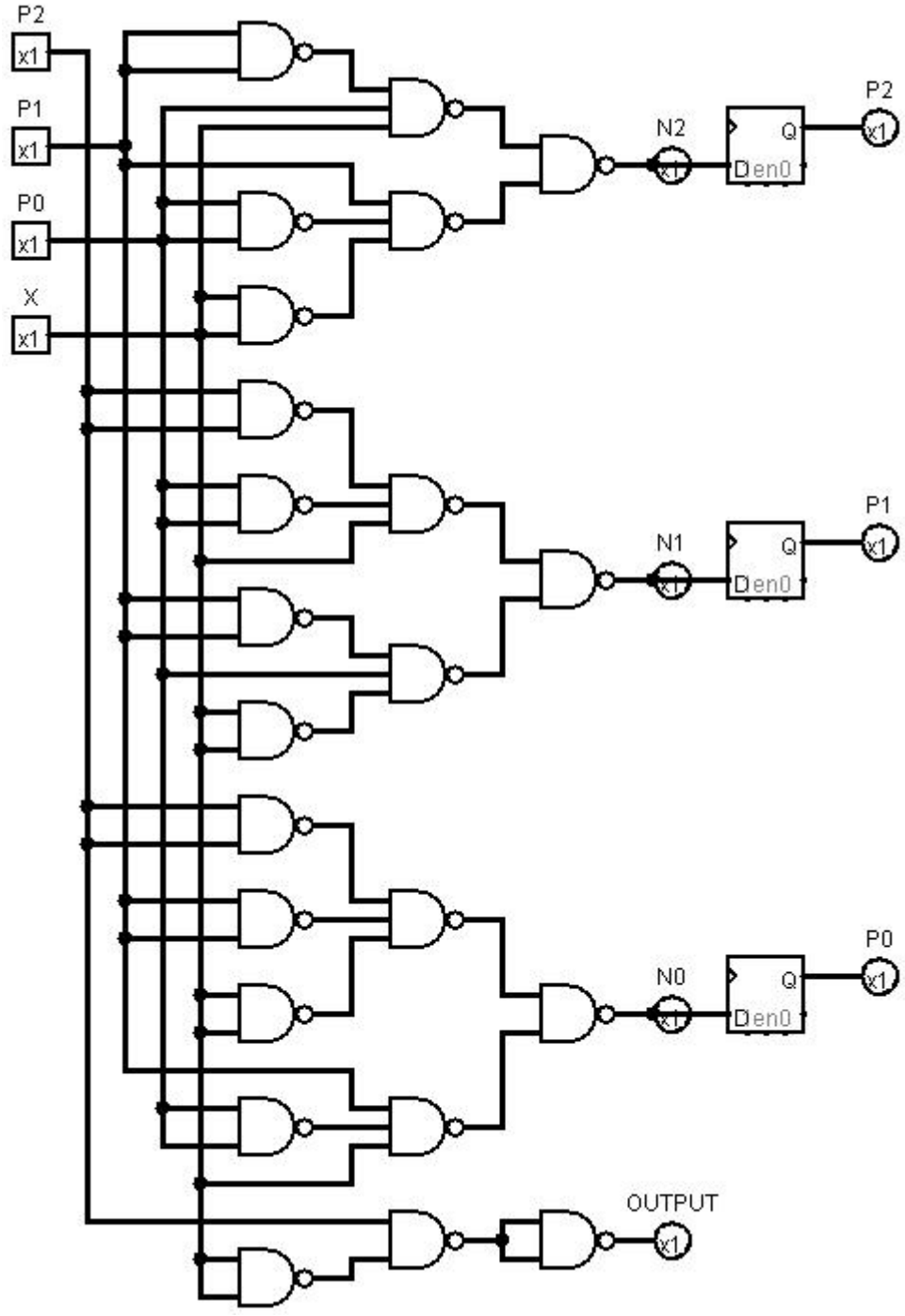
N2 = P1'P0X + P1P0'X'

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

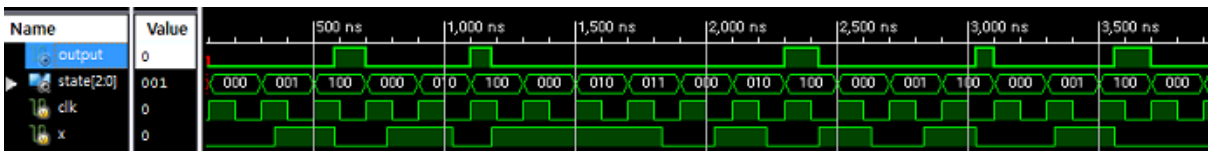
N0 = P2'P1'X' + P1P0'X

OUTPUT = P2X'

(7)


P2
x1
P1
x1
P0
x1
X
x1
N2
Q
Den0
P2
x1
N1
Q
Den0
P1
x1
N0
Q
Den0
P0
x1
OUTPUT
x1

(8)


Name
Value
output
0
state[2:0]
001
clk
0
x
0
500 ns
1,000 ns
1,500 ns
2,000 ns
2,500 ns
3,000 ns
3,500 ns
000 001 100 000 010 100 000 010 011 000 010 100 000 001 100 000 001 100 000