Indian Institute of Technology (IIT-Bombay)

AUTUMN Semester, 2025
COMPUTER SCIENCE AND ENGINEERING

CS230: Digital Logic Design and Computer Architecture

Tutorial - I

Full Marks: 0

Time allowed: ∞ hours

1. Determine the canonical sum-of-products representation of the following functions:

(a) $f(a,b,c) = c' + (a+b')(a'+b)$
(b) $f(a,b,c) = a' + (ab + a'c')'$
(c) $f(a,b,c) = (a+b)(a'+c) + bc'$

Solution:

(a) $f(a,b,c) = \sum m(0,1,2,4,6,7)$
(b) $f(a,b,c) = \sum m(0,1,2,3,4,5)$
(c) $f(a,b,c) = \sum m(1,2,3,5,6,7)$

---

2. Consider the Karnaugh map given below in fig 1, where X represents "don't care" and blank represents 0.

$dc \backslash ba$	00	01	11	10
00		X	1	
01	1			X
11	1			1
10		X	X	

Figure 1: K-Map of question 2

Assume for all inputs (a, b, c, d), the respective complements $(\bar{a}, \bar{b}, \bar{c}, \bar{d})$ are also available. The above logic is implemented using 2-input NOR gates only. The minimum number of gates required is ___________

Solution

The Function is XOR, and 3 NOR gates are required because note that $(\bar{a}, \bar{b}, \bar{c}, \bar{d})$ are also available.

Steps:
After K-map reduction: $a \oplus c = ca' + c'a$
Consider POS form: (thanks Vedant for this approach)

$$\begin{aligned}
c \oplus a &= (\overline{c} + \overline{a}).(c + a) \\
Say, NOR(a, c) &= \overline{(c + a)} \\
&= \overline{c}.\overline{a} \\
keeping\ this\ in\ mind, & \\
c \oplus a &= \overline{\overline{(\overline{c} + \overline{a})}}.\,\overline{\overline{(c + a)}} \\
&= \overline{(c.a)}.\,\overline{(\overline{c}.\overline{a})} \\
With\ compliments\ given,\ (c.a)\ ---> NOR\ 1 & \\
(\overline{c}.\overline{a})\ ---> NOR\ 2 & \\
then\ overall\ expression,\ \overline{(c.a)}.\,\overline{(\overline{c}.\overline{a})}\ ---> NOR\ 3 &
\end{aligned}$$

3. Show that $f(X,Y,Z) = XY'Z' + X'Y + YZ'$ is a universal operation.

Solution:

$f(X,Y,Z) = XY'Z' + X'Y + YZ'$
$= Z'(XY' + Y) + X'Y$
$= Z'(X + Y) + X'Y$
$= X'Y + YZ' + Z'X$ [using consensus theorem]
$= X'Y + XZ'$

$f(X,1,Z) = X' + XZ'$
$= X' + Z'$
$= (XZ)'$
We have produced a NAND gate, which is a universal operation / functionally complete.

4. Design an 8×1 multiplexer using a combination of 2×1 multiplexers.
solution

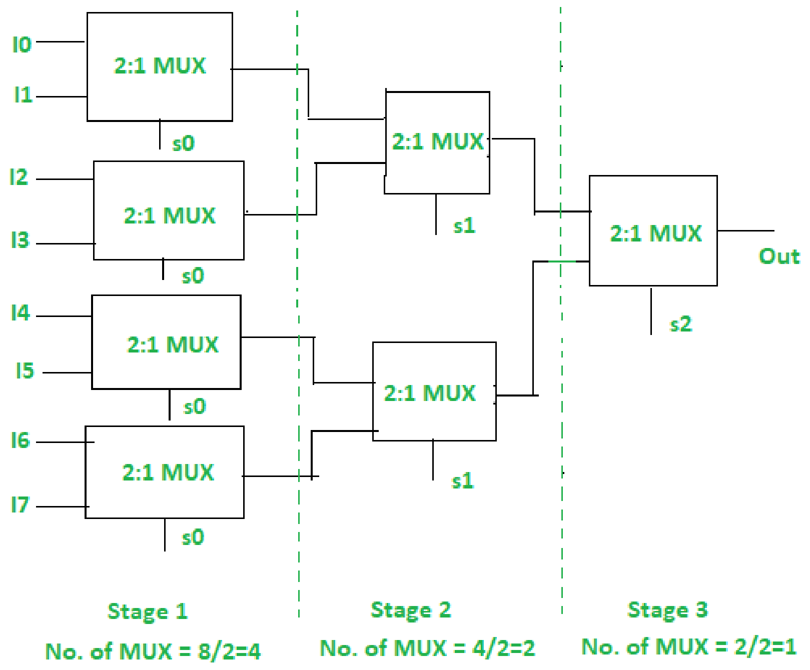


Figure 2: 8 x 1 mux from 2 x 1

5. Find the simplest function $h(A, B, C, D)$ that will make the function

$$f = A'BC + (AC + B)D + h(A, B, C, D)$$

self-dual.
Hint: The dual of a Boolean function can be obtained by replacing all the ANDs with ORs and vice versa. The constant terms are replaced with their respective complements. Boolean function can be self-dual if the truth table contains the same number of 0's and 1's, and none of the minterm pairs are mutually exclusive (that is, no two minterms are complements to each other).

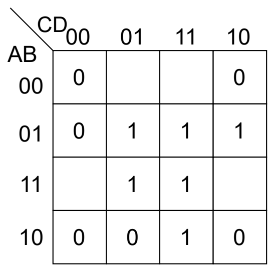
Solution:

For this question, first, construct the K-Map for the already given minterms of f. The K-Map is presented in Figure 3. From the given minterms of f (except those from h), we can fill in the 1's in the K-Map. However, since the function is self-dual, we can also find out the locations of the 0's from the given 1's. This is because, for a self-dual function if $x_0 x_1 \cdots x_n$ is a minterm, then $x_0 + x_1 + x_2 + \cdots x_n$ must also be a maxterm. Therefore, if we have a 1 in a cell, then we must have a 0 in a cell whose code is the complement to the 1-cell. With this observation, we can fill in 6 0's. Now, there are four cells which can be filled. By the property of self-dual, two of them must be filled in with 1's. For the given K-Map, we observe that either $(0001, 0011)$ or $(1110, 1100)$ must be 1. Therefore, we can have two possible minimal forms for f (i.e. two minimal h), given as:

$$f = A'BC + ACD + BD + A'D$$

and

$$f = AB + BD + A'BC + ACD$$



AB \ CD	00	01	11	10
00	0			0
01	0	1	1	1
11		1	1	
10	0	0	1	0

Figure 3: K-Map for question 5

6. You are given three-input logic gates, each realizing the function $g(x, y, z) = x \oplus yz$. Use this function as a block to implement the following function

$$f = (a + b)c + ab'.$$

Prove/disprove that by using these gates only, you can realize any Boolean function. Show all your steps.

Solution:

$f = (a + b)c + ab'$
$f = ac + bc + ab'$ [using consensus theorem]
$f = ab' + bc$

whereas,
$g(x, y, z) = x \oplus yz$
$g(x, 1, z) = x \oplus z$
$g(1, y, z) = (yz)'$
$g(0, y, z) = yz$

using the new derivations from g,
$g(0, b, c) = bc$.......................(1)
$g(1, y, 1) = y'$
$g(0, a, g(1, b, 1)) = ab'$..........(2)
$g(1, g(1, y, 1), g(1, z, 1)) = y + z$ [add operation]...(3)
Using (1), (2) with the help of (3),
$g(1, g[1, g(0, a, g(1, b, 1)), 1], g[1, g(0, b, c), 1]) = ab' + bc$

---

7. Design a circuit that takes in a 3-bit signed (2's complement) number X and produces an output $Z = X^2 + 2X + 1$

---

solution

$Z = (X + 1)^2$
X is a 3-bit two's complements number (X_2(MSB/ Sign), then X_1,X_0). The output is always non-negative and fits in 5 bits $[0, 16]$

X (bin)	X (dec)	$Z = (X + 1)^2$	$z_4z_3z_2z_1z_0$
100	-4	9	01001
101	-3	4	00100
110	-2	1	00001
111	-1	0	00000
000	0	1	00001
001	1	4	00100
010	2	9	01001
011	3	16	10000

Table 1: Truth table for $Z = (X + 1)^2$ with 3-bit signed input X

using K-Maps, we can derive

$$\begin{aligned}
z_4 &= \overline{x_2}\, x_1\, x_0, \\
z_3 &= \overline{x_0}\, (x_1 \oplus x_2), \\
z_2 &= \overline{x_1}\, x_0, \\
z_1 &= 0, \\
z_0 &= \overline{x_0}.
\end{aligned}$$

8. Design a 1-bit comparator that takes in a bit X and a bit Y and outputs $X < Y, X > Y, X = Y$. Use a single 2-to-4 decoder and 1 single OR gate

---

Solution:

There are 4 possible input combinations:

X	**Y**	$X > Y$	$X < Y$	$X = Y$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$X > Y = A\bar{B}$
$X < Y = \bar{A}B$
$X = Y = \bar{A}\bar{B} + A\bar{B}$

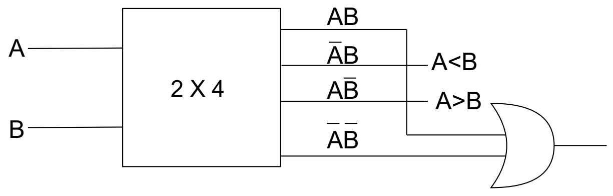


Figure 4

---

9. A new flip-flop, called MN flip-flop, is constructed from a JK flip-flop as follows:

$$J = M, \quad K = N \oplus M$$

(a) Derive the state-transition table and excitation table for this new flip-flop (b) Derive the characteristic equation (c) Construct a D flip-flop from this new flip-flop

---

Solution:

(a) State-Transition Table and Excitation Table

JK Flip-Flop Characteristic Table

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$

MN Flip-Flop State Transition Table

Using:

$$J = M, \quad K = M \oplus N$$

M	N	Q_t	J	K	Q_{t+1}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	1

Excitation Table

Q_t	Q_{t+1}	Required (M, N)
0	0	(0,0), (0,1)
0	1	(1,0), (1,1)
1	0	(0,1), (1,0)
1	1	(0,0), (1,1)

(b) Characteristic Equation

$Q_t \backslash MN$	00	01	11	10
0	0	0	1	1
1	1	0	1	0

Minterms where $Q_{t+1} = 1$:

- $Q_t = 0, MN = 11 \Rightarrow M \cdot N \cdot \overline{Q_t}$
- $Q_t = 0, MN = 10 \Rightarrow M \cdot \overline{N} \cdot \overline{Q_t}$
- $Q_t = 1, MN = 00 \Rightarrow \overline{M} \cdot \overline{N} \cdot Q_t$
- $Q_t = 1, MN = 11 \Rightarrow M \cdot N \cdot Q_t$

From the state transition table, we derive the characteristic equation:

$$\begin{aligned}
\text{Express as sum of products:}\quad & Q_{t+1} = (\overline{M}\cdot\overline{N}\cdot Q_t) + (M\cdot\overline{N}\cdot\overline{Q_t}) + (M\cdot N\cdot\overline{Q_t}) + (M\cdot N\cdot Q_t) \\
\text{Simplify:}\quad & Q_{t+1} = \overline{M}\,\overline{N}\,Q_t + M\,\overline{N}\,\overline{Q_t} + MN(\overline{Q_t} + Q_t) \\
\text{Since}\quad \overline{Q_t} + Q_t = 1 : & \\
& Q_{t+1} = \overline{M}\,\overline{N}\,Q_t + M\,\overline{N}\,\overline{Q_t} + MN \\
\text{Alternatively,}\quad & Q_{t+1} = MN + \overline{N}(\overline{M}Q_t + M\overline{Q_t})
\end{aligned}$$

(c) Construct a D Flip-Flop from MN Flip-Flop

We want the MN flip-flop to behave like a D flip-flop:

$$Q^+ = D$$

Choose:

$$M = D, \quad N = Q$$

Then:

$$J = D, \quad K = D \oplus Q$$

This results in the desired D flip-flop behavior.

10. Construct a counter for the following sequence using T flip-flops: 000 → 010 → 111 → 100 → 011. (a) Draw the state-table (b) Construct the next-state map (c) Derive the inputs of the T flip-flops (d) Draw the final circuit

---

Solution:

(a) State Table

Present State ($Q_2Q_1Q_0$)	Next State ($Q_2^+Q_1^+Q_0^+$)
000	010
010	111
111	100
100	011
011	000

—

(b) Next-State Map

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+
0	0	0	0	1	0
0	1	0	1	1	1
1	1	1	1	0	0
1	0	0	0	1	1
0	1	1	0	0	0

—

(c) Derive the T Flip-Flop Inputs

Recall the excitation table for T flip-flops:

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Thus,

$$T = Q \oplus Q^+$$

Calculate the T inputs for each flip-flop:

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	$T_2 = Q_2 \oplus Q_2^+$	$T_1 = Q_1 \oplus Q_1^+$	$T_0 = Q_0 \oplus Q_0^+$
0	0	0	0	1	0	0	1	0
0	1	0	1	1	1	1	0	1
1	1	1	1	0	0	0	1	1
1	0	0	0	1	1	1	1	1
0	1	1	0	0	0	0	1	1

—

Simplify the T Inputs

For T_2:

$$T_2 = \sum m(2,4)$$

Where:

$$m_2 = Q_2'Q_1Q_0', \quad m_4 = Q_2Q_1'Q_0'$$

Thus,

$$T_2 = Q_2'Q_1Q_0' + Q_2Q_1'Q_0'$$

—

For T_1: T_1 is 1 for all states except at 010.
Thus,

$$T_1 = \overline{Q_2'Q_1Q_0'} = Q_2 + Q_1' + Q_0$$

—

For T_0: T_0 is 0 only at 000, so

$$T_0 = Q_2 + Q_1 + Q_0$$

—

(d) Final Circuit

The final circuit consists of three T flip-flops with inputs:

$$\boxed{\begin{cases} T_2 = Q_2'Q_1Q_0' + Q_2Q_1'Q_0' \\ T_1 = Q_2 + Q_1' + Q_0 \\ T_0 = Q_2 + Q_1 + Q_0 \end{cases}}$$

Logic gates needed:

- NOT gates for Q_2', Q_1', Q_0'
- AND gates for the two terms in T_2
- OR gates to combine terms in T_2, and to generate T_1 and T_0
- Connect T_i inputs to corresponding T flip-flops Q_i

All flip-flops are triggered by the common clock signal.

11. A sequential circuit has one flip flop Q, two inputs X and Y and one output S. The circuit consists of a full subtractor circuit connected to a D flip flop as shown in Figure 5 below. Derive the state table and state diagram for the sequential circuit.

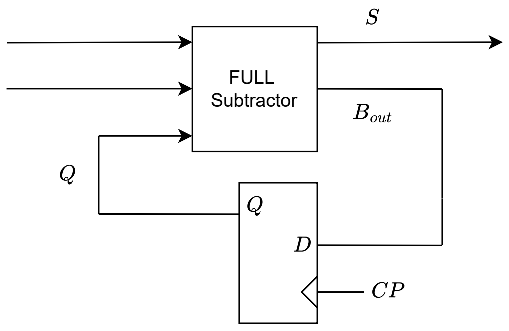


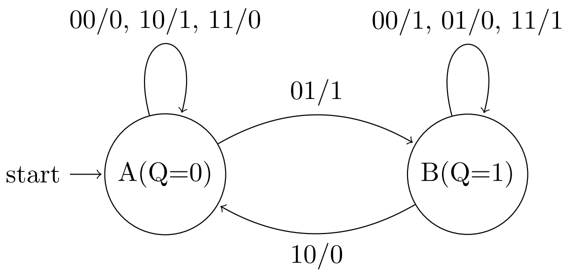
Figure 5

Solution

total 8 rows

Q (Present State)	x	y	B_{out} = Q(t+1)	S = x ⊕ y ⊕ Q
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

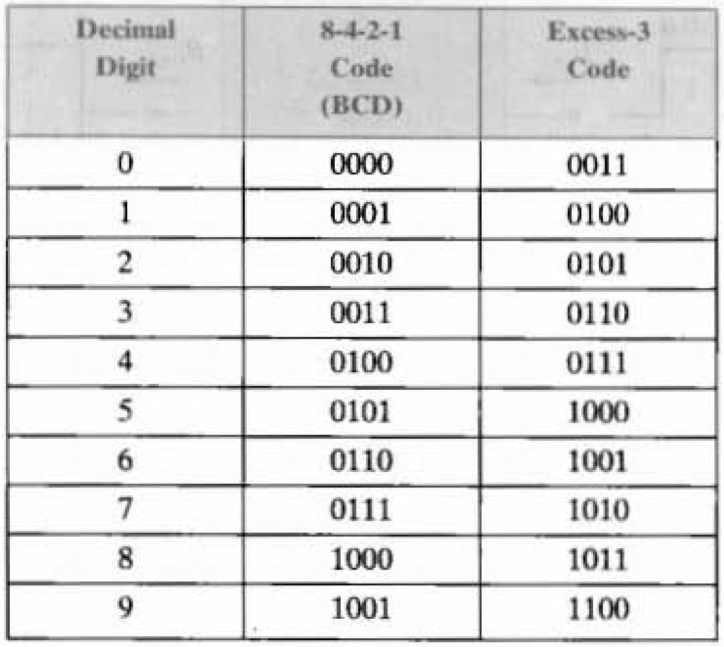
Table 2: State Table of Sequential Circuit with Full Subtractor



12. In this question we construct a serial BCD to Excess-3 code converter from serial input. In other words, the input will be provided, and the output will be generated one bit at a time. Excess-3 code can be generated by adding $(0011)_2$ with the BCD code. Therefore, you need to consider 4 bits of input at a time as a valid BCD code. The expected input and outputs provided in Figure 6. Observe that the inputs will be provided from LSB to MSB and the outputs will be generated in the same manner (more precisely, the LSB of the input will be processed first and the LSB of the output will be generated first). So you have to consider each entry in the table from right to left while processing.

- In this question you will be generating a circuit for this state machine. First generate the state-transition graph and the state-transition table. *Hint:* The state-transition graph will have the states as the nodes and transitions of the form input/output as the links. The number of states can be quite large in this case. To keep the number of states low, we use some tricks.
  - **Observation 1:** Since the input is of 4 bits, you need to transit through 4 states for each input.
  - **Observation 2:** Let S_t be any state processing the i-th bit of input for some string. It will have two possible transitions out of it for $x_i = 0$ and $x_i = 1$. Observe (from the table) that, for any bit location i, the output $y_i = \overline{x_i}$ or $y_i = x_i$ for $x_i \in \{0, 1\}$. In other words, the two transitions from any given state will be either (0/1), (1/0) or (0/0), (1/1). There will be no other transitions, such as, (0/1), (1/1).
  - **Observation 3:** There can be total 16 possible states. But many of these states will be equivalent and, therefore, can be merged together.
  - **Trick 1:** There will some state transitions which are impossible. Find them out and add dummy transitions for them keeping observation 2 in mind.
  - **Trick 2:** The initial state S_0 will also be the final state.
  - **Trick 3:** Eliminate the equivalent states. Two states S_i and S_j are equivalent if and only if for every possible input sequence, the same output sequence is produced, regardless of whether S_i or S_j is the starting state. This much should be sufficient for you to solve the problem.

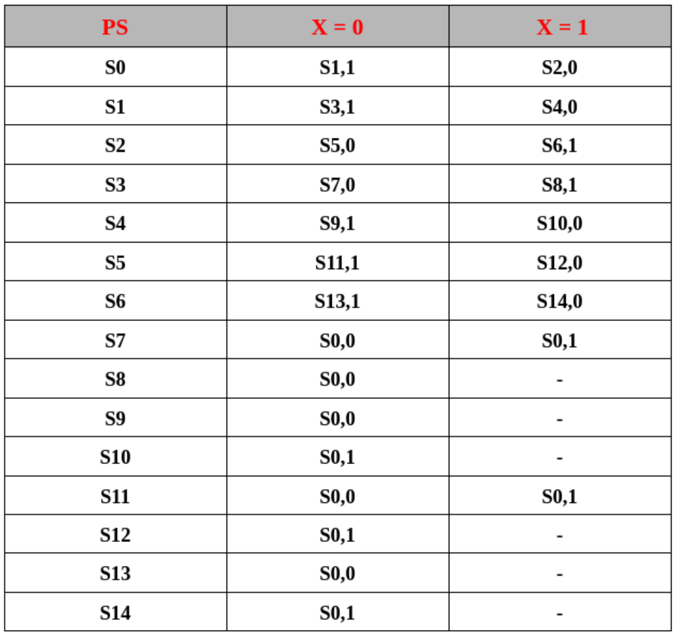
(b) Perform the state assignment using binary encoding. If you have any unused state, use it as don't care.



Decimal Digit	8-4-2-1 Code (BCD)	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

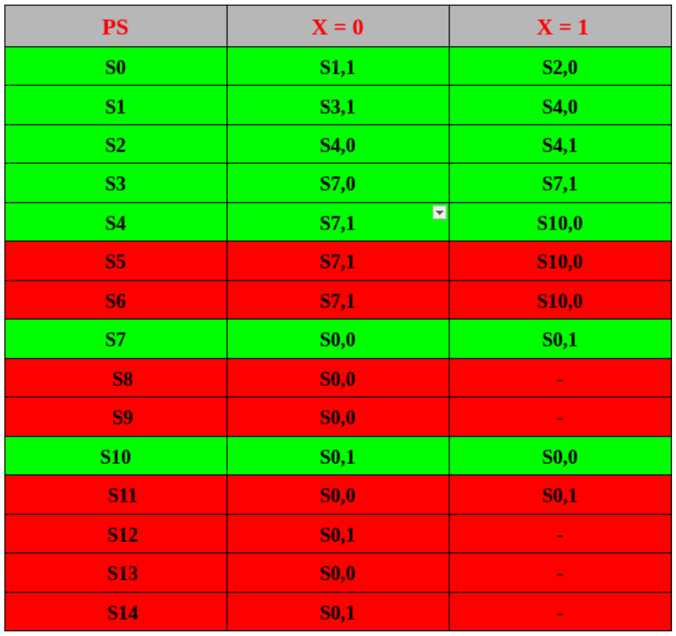
Figure 6: BCD to Excess-3 Code Converter

solution



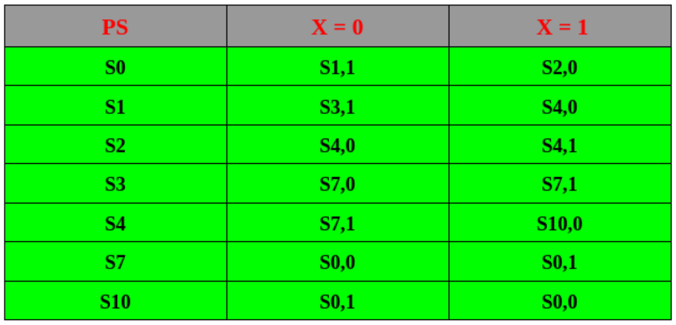
PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S5,0	S6,1
S3	S7,0	S8,1
S4	S9,1	S10,0
S5	S11,1	S12,0
S6	S13,1	S14,0
S7	S0,0	S0,1
S8	S0,0	-
S9	S0,0	-
S10	S0,1	-
S11	S0,0	S0,1
S12	S0,1	-
S13	S0,0	-
S14	S0,1	-

Figure 7: State Table



PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S4,0	S4,1
S3	S7,0	S7,1
S4	S7,1	S10,0
S5	S7,1	S10,0
S6	S7,1	S10,0
S7	S0,0	S0,1
S8	S0,0	-
S9	S0,0	-
S10	S0,1	S0,0
S11	S0,0	S0,1
S12	S0,1	-
S13	S0,0	-
S14	S0,1	-

Figure 8: Minimizing State Table

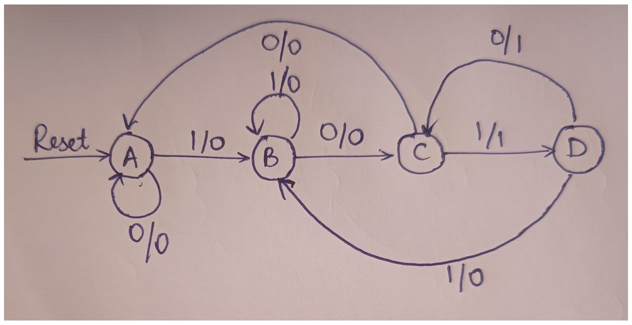


PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S4,0	S4,1
S3	S7,0	S7,1
S4	S7,1	S10,0
S7	S0,0	S0,1
S10	S0,1	S0,0

Figure 9: Minimized State Table

13. Design a circuit that detects the signal 1010 or 101 in a sequence of bits. For example, when given a sequence of 101011011110001 as an input, the output has to be 001100010000001. Use any flip-flop possible.

solution



Defining the states as follows:

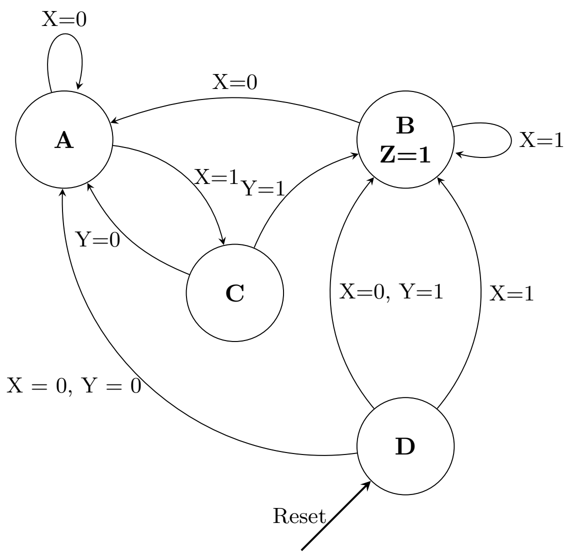
State	Encoding (MN)
A	00
B	01
C	11
D	10

The state transition table is as follows:

M	N	x	M_-	N_-	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	1	0	0	0	0
1	1	1	1	0	1
1	0	0	1	1	1
1	0	1	0	1	0

$$M_- = M'Nx' + MNx + MN'x'$$
$$N_- = M'x + MN'$$
$$z = MN'X' + MNX$$

14. Design the circuit for the following finite state machine with inputs X and Y. Define a method to represent the states in the circuit



---

solution

This can be solved with Karnaugh maps with the data

Considering the states as follows:

State	Encoding (MN)
A	00
B	01
C	11
D	10

The state transition table is as follows:

Previous State	X	Y	Next State	Z
A	0	0	*A*	0
A	0	1	*A*	0
A	1	0	*C*	0
A	1	1	*C*	0
B	0	0	*A*	0
B	0	1	*A*	0
B	1	0	*B*	1
B	1	1	*B*	1
C	0	0	*A*	0
C	0	1	*B*	1
C	1	0	*A*	0
C	1	1	*B*	1
D	0	0	*A*	0
D	0	1	*B*	1
D	1	0	*B*	1
D	1	1	*B*	1

or using encoding way

MN	X	Y	M^+N^+	Z
00	0	0	00	0
00	0	1	00	0
00	1	0	11	0
00	1	1	11	0
01	0	0	00	0
01	0	1	00	0
01	1	0	01	1
01	1	1	01	1
11	0	0	00	0
11	0	1	00	0
11	1	0	01	1
11	1	1	01	1
10	0	0	00	0
10	0	1	01	1
10	1	0	01	1
10	1	1	01	1

Table 3: State transition/output table

Using two D Flip Flops, one for M+ and another for N+. The K-Maps for M+, N+, z is given in the fig- 10

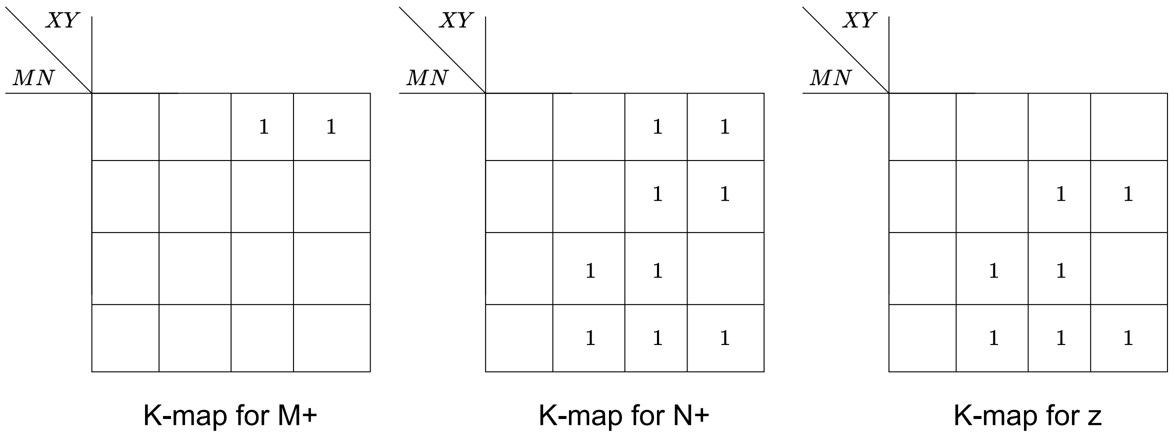


Figure 10: K-Maps for M+, N+, z

Using the K-Maps, we get,

$$M^+ = M'N'X$$
$$N^+ = M'X + MY + MN'X$$
$$Z = MY + (M \oplus N)X$$

The final circuit Diagram looks like this

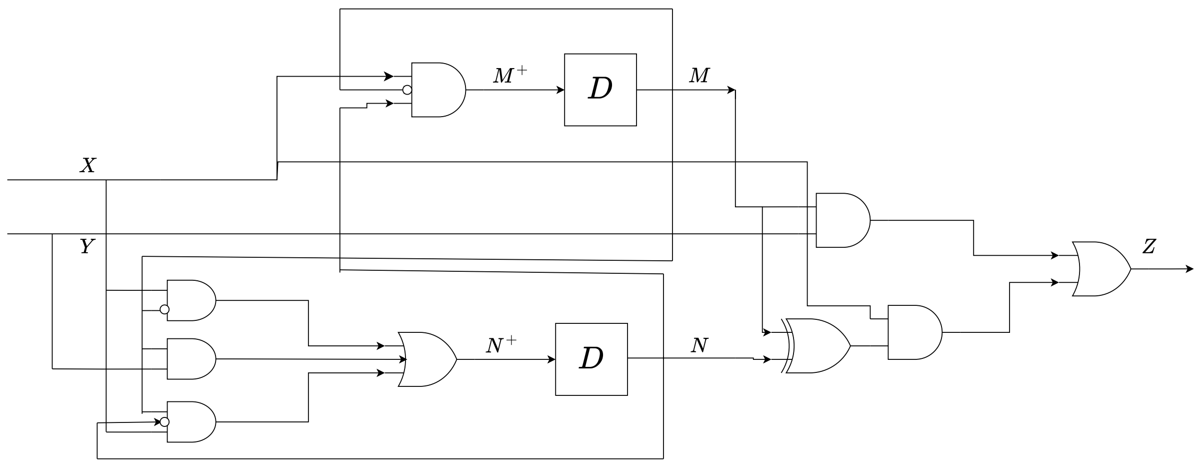


Figure 11: Final circuit