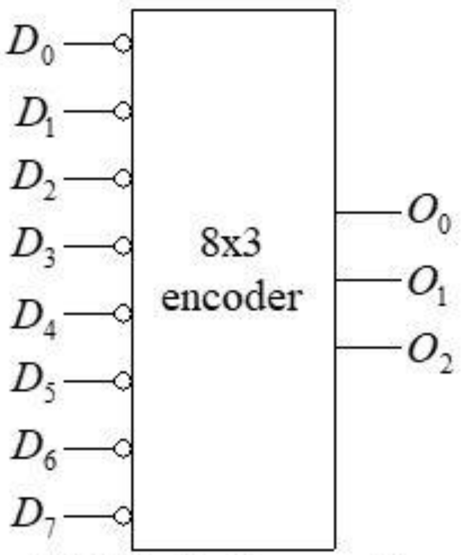
Chapter 8
Encoders, Decoders, Multiplexers, and Demultiplexers

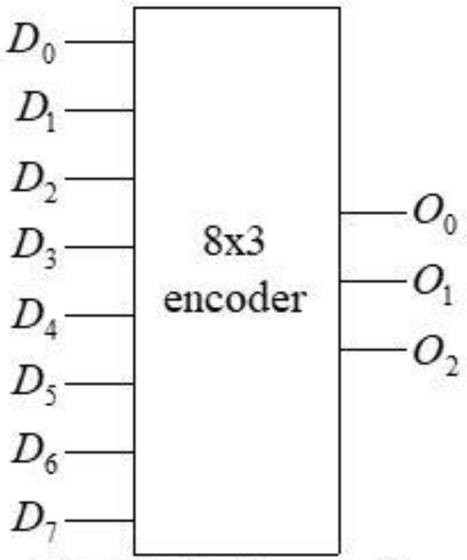
8.2 Encoders

8.2.1 Design of Encoders

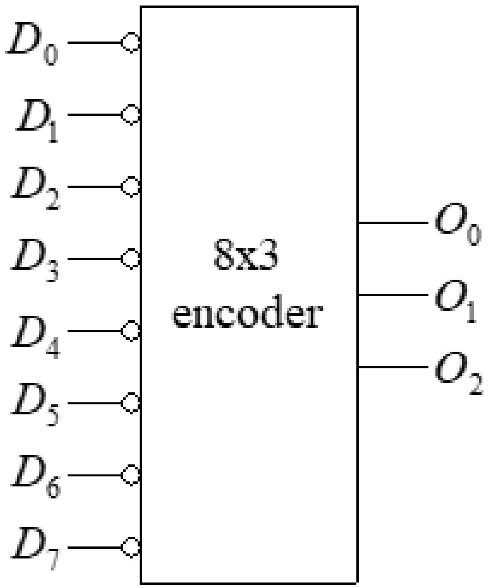
An *encoder* is a combinational circuit that has 2^n (or less) input lines and n output lines. Only one of the input lines is activated at a given time. The output lines generate the binary code corresponding to the activated input. The input lines can be activated in two ways – by setting the activated input to 0 and the other inputs to 1 (active-LOW input) or by setting the activated input to 1 and the other inputs to 0 (active-HIGH input). The block diagram and truth tables of an



(a) Block diagram for active-LOW input



(c) Block diagram for active-HIGH input



(a) Block diagram for active-LOW input

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	O_2	O_1	O_0
1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1	0	1	0
1	1	1	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1

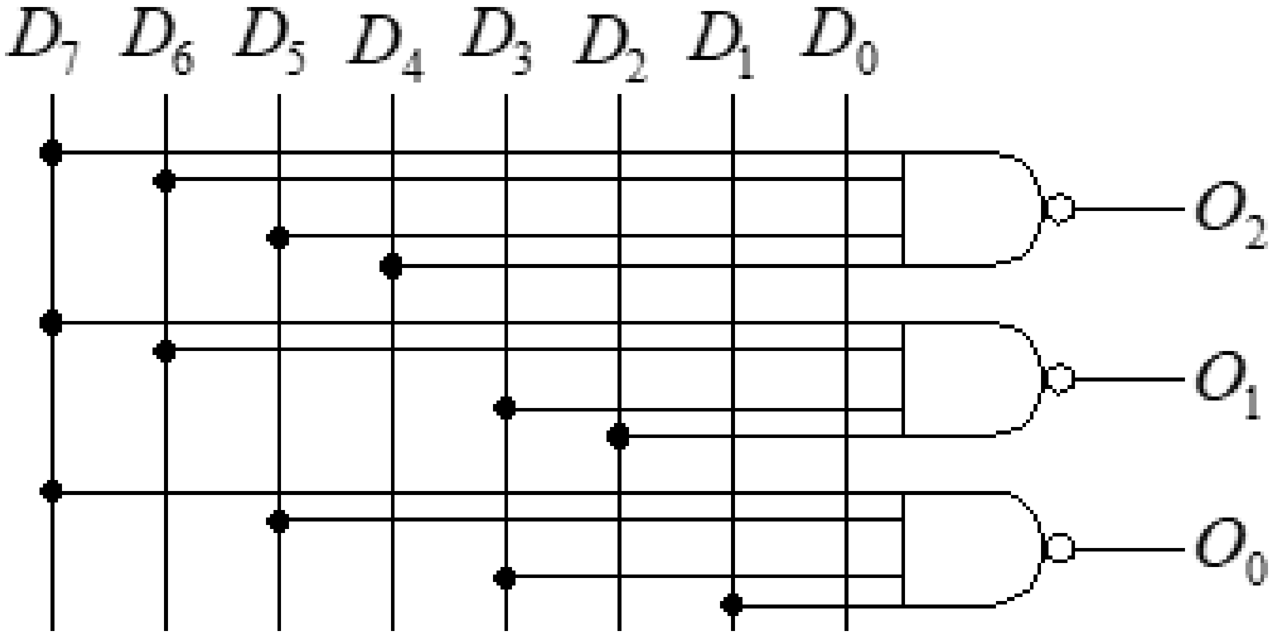
(b) Truth table for active-LOW input

For active-LOW input

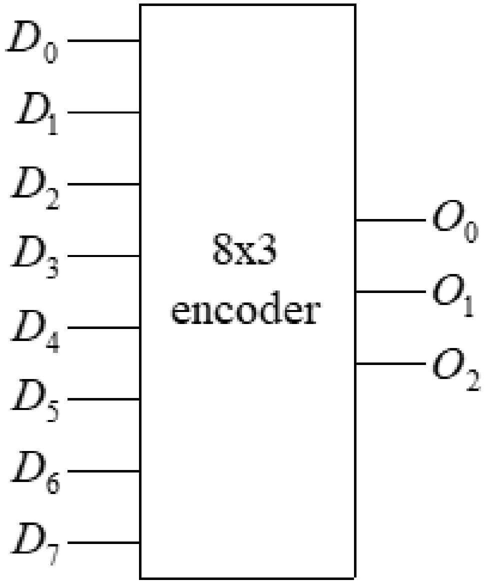
$$O_0 = D_1' + D_3' + D_5' + D_7' = (D_1 D_3 D_5 D_7)'$$
$$O_1 = D_2' + D_3' + D_6' + D_7' = (D_2 D_3 D_6 D_7)'$$
$$O_2 = D_4' + D_5' + D_6' + D_7' = (D_4 D_5 D_6 D_7)'$$

For active-LOW input

$$O_0 = D_1' + D_3' + D_5' + D_7' = (D_1 D_3 D_5 D_7)'$$
$$O_1 = D_2' + D_3' + D_6' + D_7' = (D_2 D_3 D_6 D_7)'$$
$$O_2 = D_4' + D_5' + D_6' + D_7' = (D_4 D_5 D_6 D_7)'$$



(a) Active-LOW input



(c) Block diagram for active-HIGH input

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	O_2	O_1	O_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

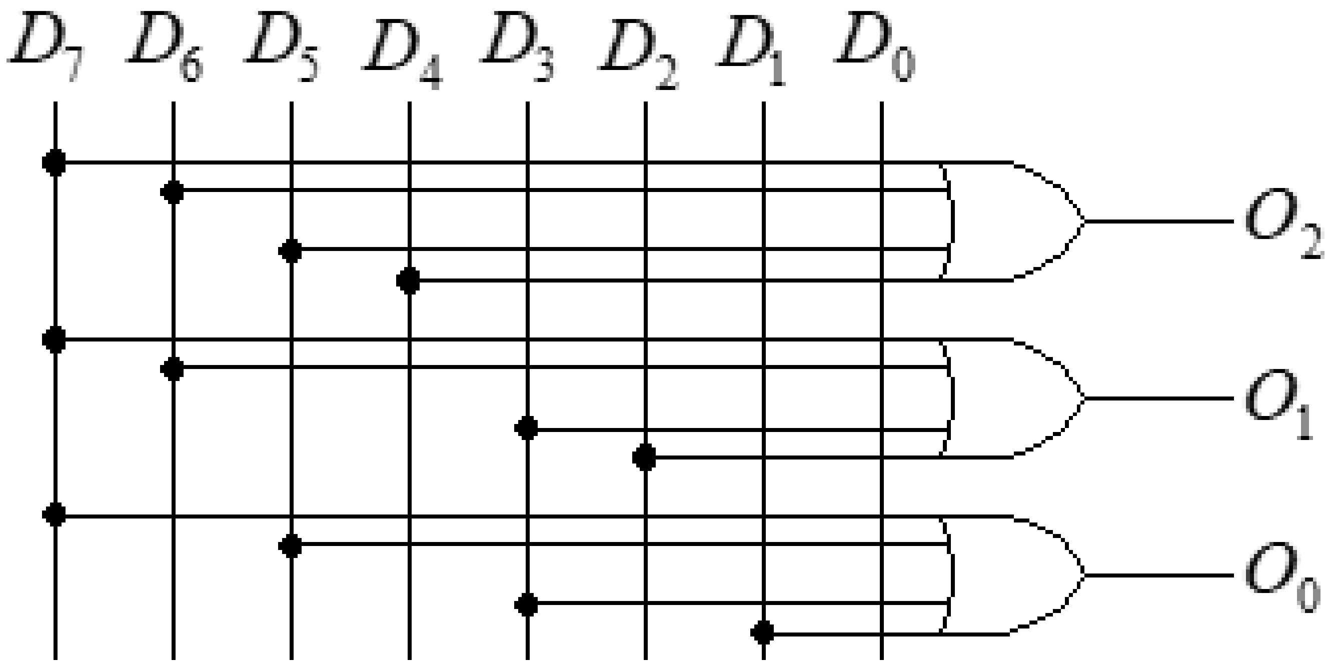
(d) Truth table for active-HIGH input

For active-HIGH input

$$O_0 = D_1 + D_3 + D_5 + D_7$$
$$O_1 = D_2 + D_3 + D_6 + D_7$$
$$O_2 = D_4 + D_5 + D_6 + D_7$$

For active-HIGH input

$$O_0 = D_1 + D_3 + D_5 + D_7$$
$$O_1 = D_2 + D_3 + D_6 + D_7$$
$$O_2 = D_4 + D_5 + D_6 + D_7$$

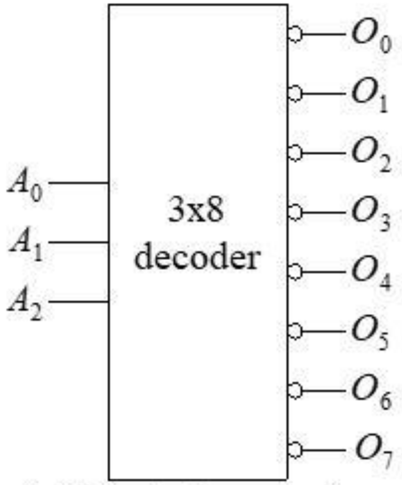


(b) Active-HIGH input

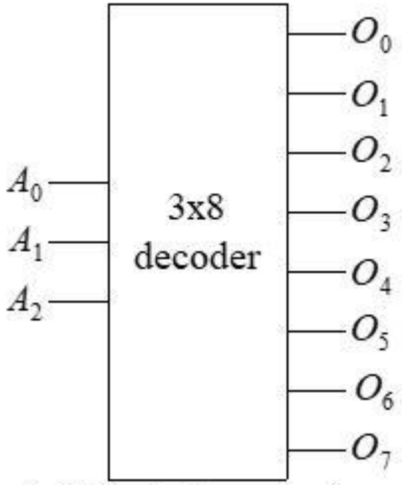
8.3 Decoders

8.3.1 Design of Decoders

A *decoder* is a combinational circuit that accepts *n* inputs that represent a binary code and activate only the output that corresponds to the input code. The number of output lines may be a maximum of 2^n. If the *n*-bit decoded information has unused or don't care combinations, the decoder output will have less than 2^n outputs. The output lines can be activated in two ways – by setting the activated output to 0 and the other outputs to 1 (active-LOW output) or by setting the activated output to 1 and the other outputs to 0 (active-HIGH output). If the number of output lines is less than 2^n, the decoder is often designed such that all the outputs remain inactivated when an invalid input combination (unused code) is applied.

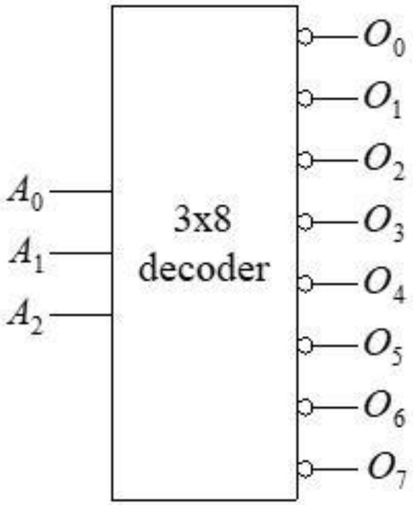


(a) Block diagram for active-LOW output

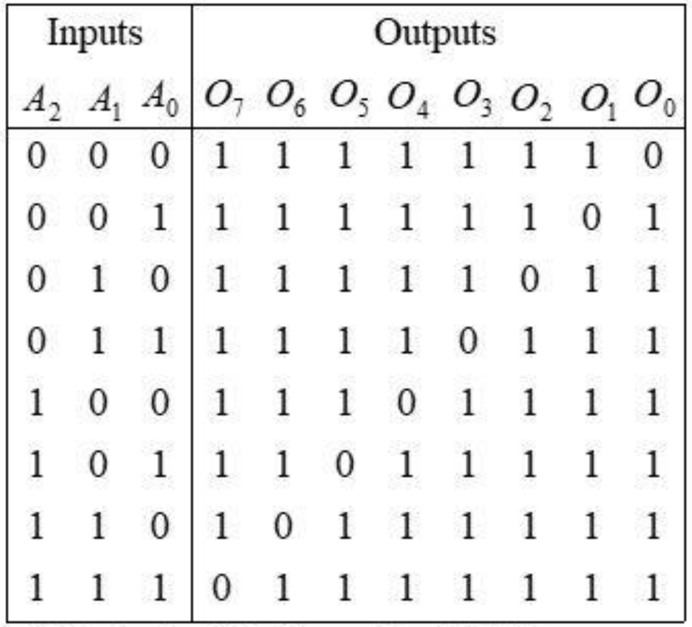


(c) Block diagram for active-HIGH output

Binary-to-Octal Decoder



(a) Block diagram for active-LOW output

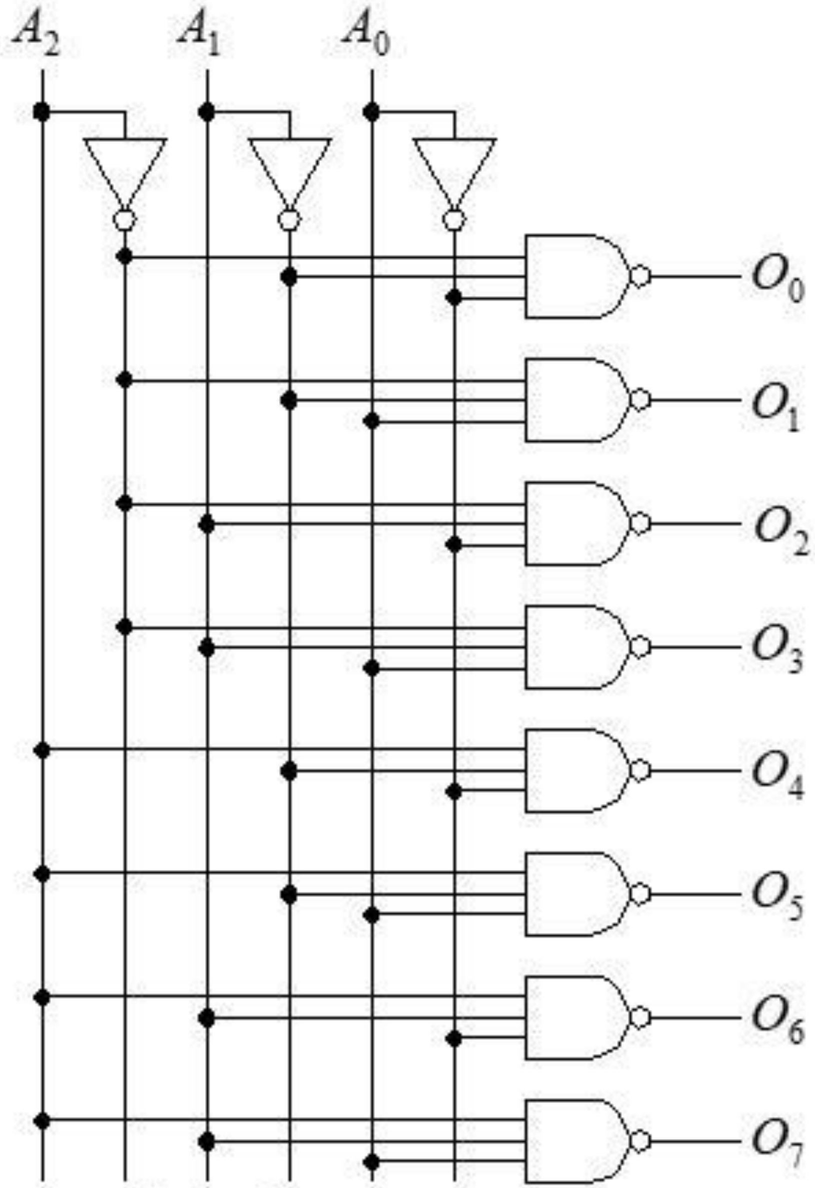


Inputs			Outputs							
A_2	A_1	A_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1
0	1	0	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1

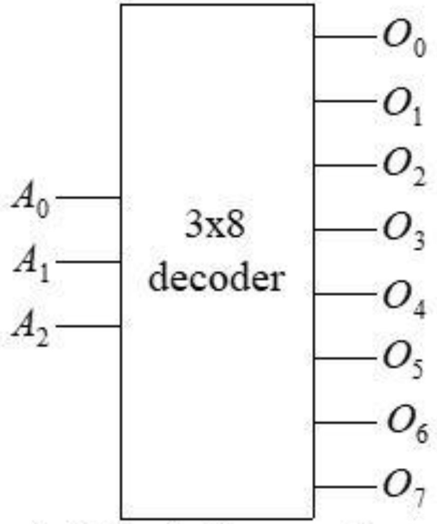
(b) Truth table for active-LOW output

For active-LOW output

$$O_0 = A_2 + A_1 + A_0 = (A_2'A_1'A_0')'$$
$$O_1 = A_2 + A_1 + A_0' = (A_2'A_1'A_0)'$$
$$O_2 = A_2 + A_1' + A_0 = (A_2'A_1A_0')'$$
$$O_3 = A_2 + A_1' + A_0' = (A_2'A_1A_0)'$$
$$O_4 = A_2' + A_1 + A_0 = (A_2A_1'A_0')'$$
$$O_5 = A_2' + A_1 + A_0' = (A_2A_1'A_0)'$$
$$O_6 = A_2' + A_1' + A_0 = (A_2A_1A_0')'$$
$$O_7 = A_2' + A_1' + A_0' = (A_2A_1A_0)'$$



(a) Active-LOW output



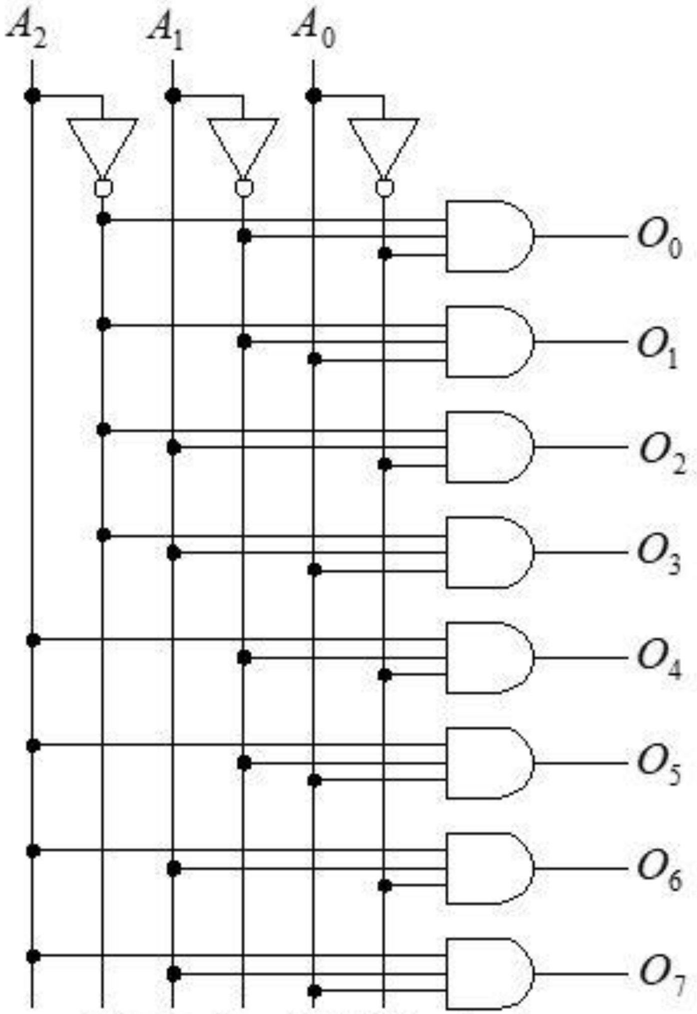
(c) Block diagram for active-HIGH output

Inputs			Outputs							
A_2	A_1	A_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

(d) Truth table for active-HIGH output

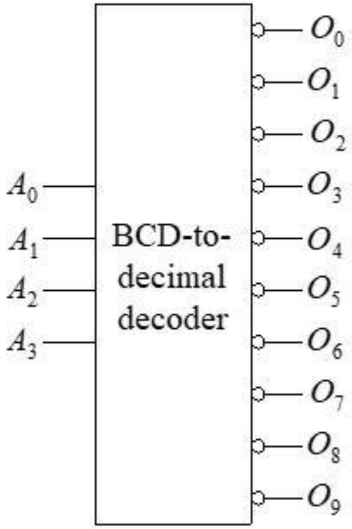
For active-HIGH output

$$O_0 = A_2'A_1'A_0'$$
$$O_1 = A_2'A_1'A_0$$
$$O_2 = A_2'A_1A_0'$$
$$O_3 = A_2'A_1A_0$$
$$O_4 = A_2A_1'A_0'$$
$$O_5 = A_2A_1'A_0$$
$$O_6 = A_2A_1A_0'$$
$$O_7 = A_2A_1A_0$$

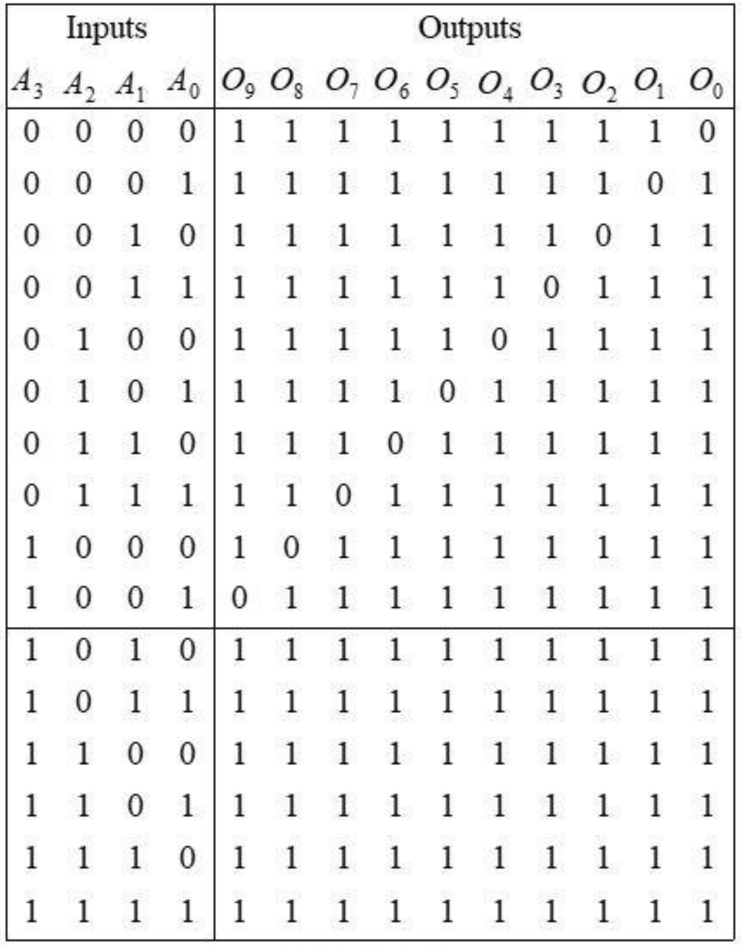


(a) Active-HIGH output

BCD-to-Decimal Decoder with active-LOW output:

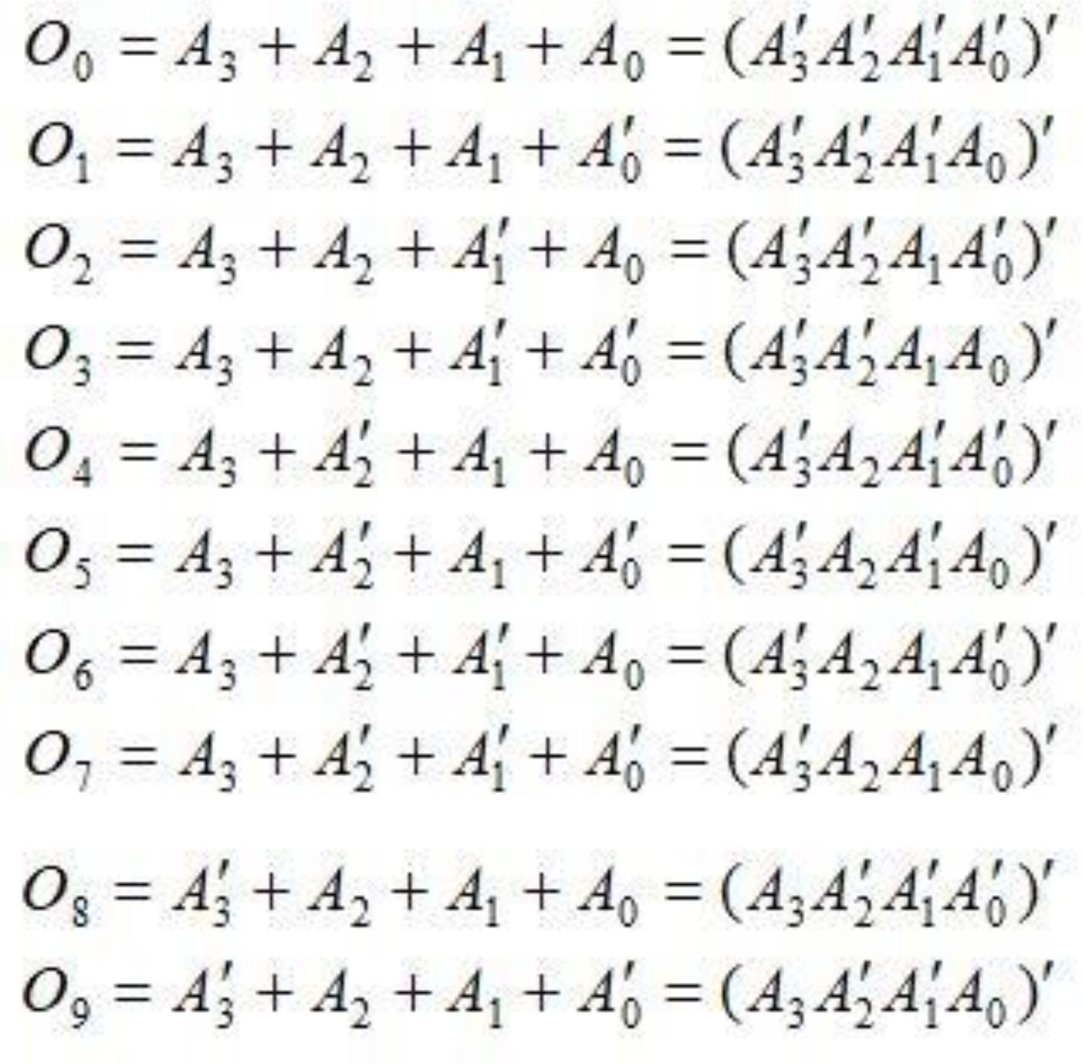


(a) Block diagram



Inputs				Outputs									
A_3	A_2	A_1	A_0	O_9	O_8	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	1	1	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	0	0	1	0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	1	1	1
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

(b) Truth table



$$O_0 = A_3 + A_2 + A_1 + A_0 = (A_3'A_2'A_1'A_0')'$$

$$O_1 = A_3 + A_2 + A_1 + A_0' = (A_3'A_2'A_1'A_0)'$$

$$O_2 = A_3 + A_2 + A_1' + A_0 = (A_3'A_2'A_1A_0')'$$

$$O_3 = A_3 + A_2 + A_1' + A_0' = (A_3'A_2'A_1A_0)'$$

$$O_4 = A_3 + A_2' + A_1 + A_0 = (A_3'A_2A_1'A_0')'$$

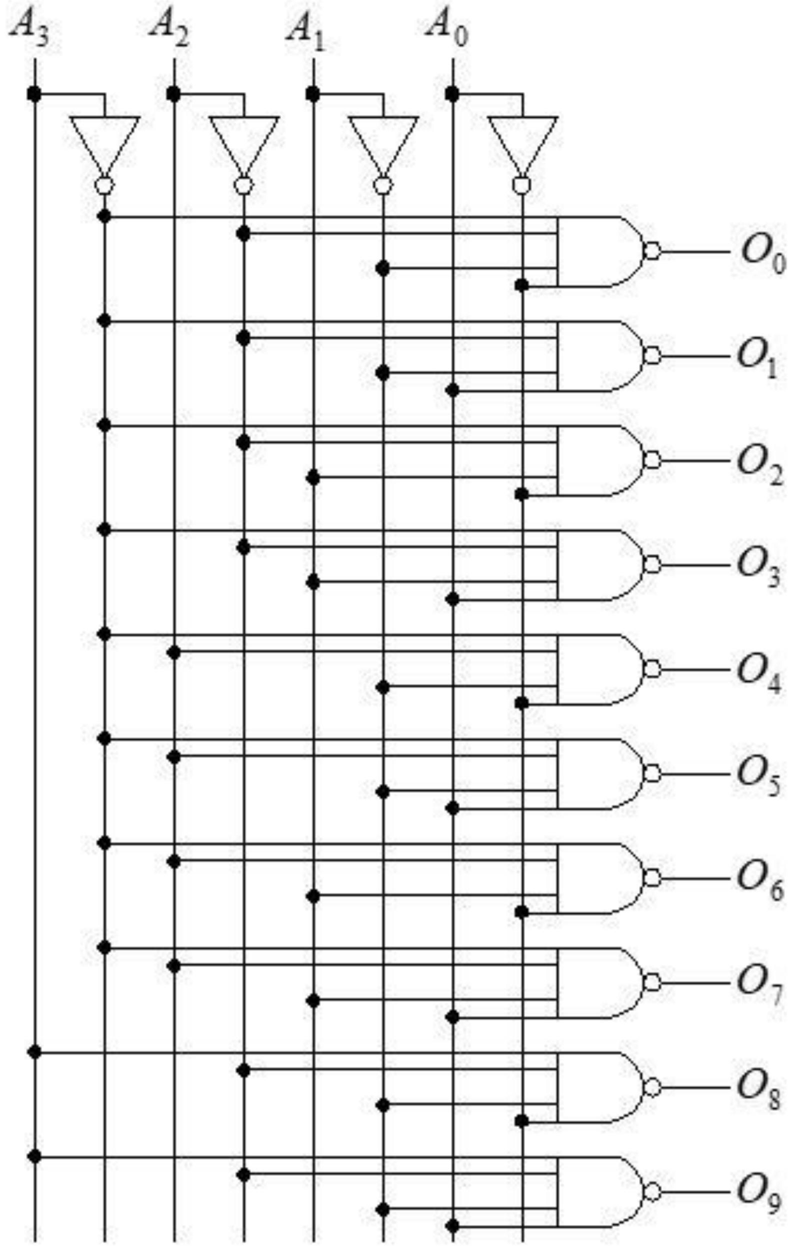
$$O_5 = A_3 + A_2' + A_1 + A_0' = (A_3'A_2A_1'A_0)'$$

$$O_6 = A_3 + A_2' + A_1' + A_0 = (A_3'A_2A_1A_0')'$$

$$O_7 = A_3 + A_2' + A_1' + A_0' = (A_3'A_2A_1A_0)'$$

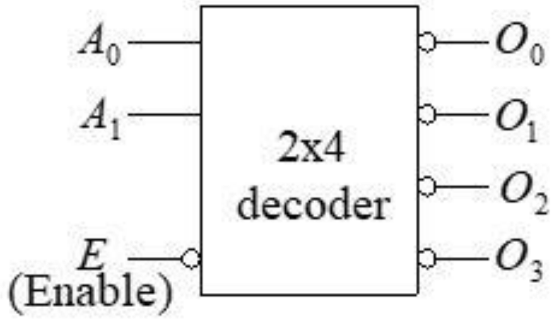
$$O_8 = A_3' + A_2 + A_1 + A_0 = (A_3A_2'A_1'A_0')'$$

$$O_9 = A_3' + A_2 + A_1 + A_0' = (A_3A_2'A_1'A_0)'$$


A_3
A_2
A_1
A_0
O_0
O_1
O_2
O_3
O_4
O_5
O_6
O_7
O_8
O_9

8.3.2 Decoders with Enable Input

Some decoders have enable input that is used to control the operation of the decoder. If the enable input is active, the output corresponding to the input is activated. If the enable input is not active, no output is activated. The block diagram and truth table of a 2-to-4-line decoder with



(a) Block diagram

Inputs			Outputs			
E	A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

(b) Truth table

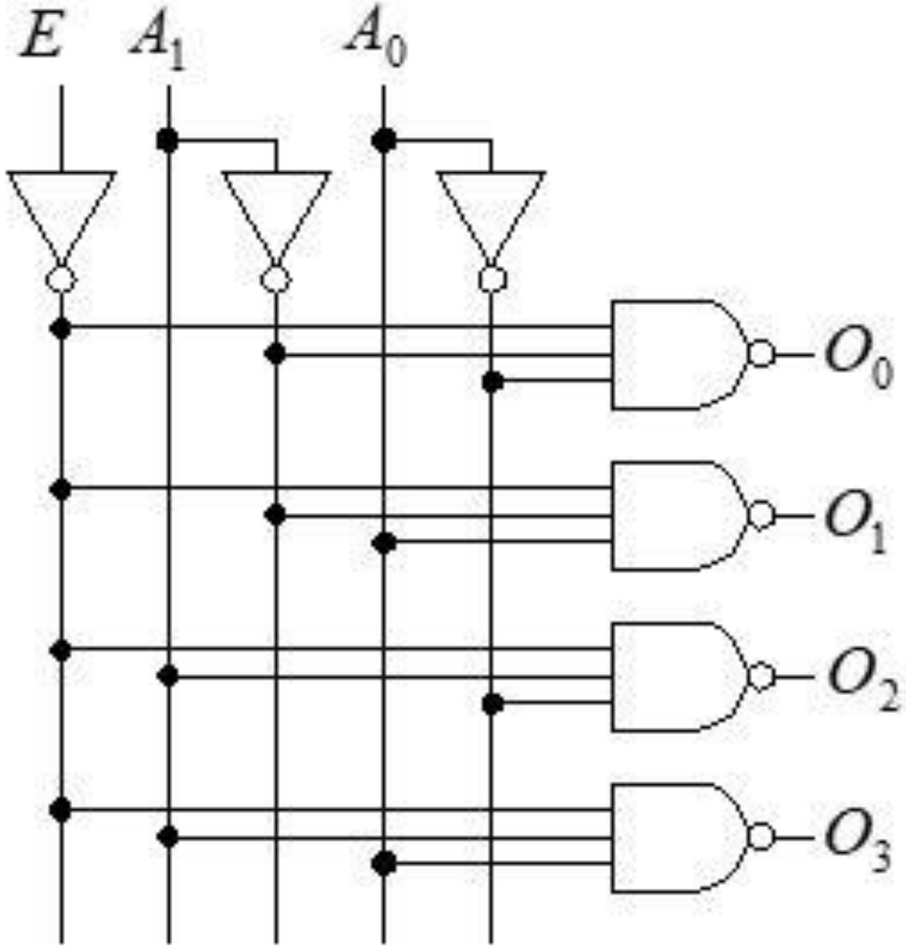
$$O_0 = E + A_1 + A_0 = (E'A_1'A_0')'$$
$$O_1 = E + A_1 + A_0' = (E'A_1'A_0)'$$
$$O_2 = E + A_1' + A_0 = (E'A_1A_0')'$$
$$O_3 = E + A_1' + A_0' = (E'A_1A_0)'$$

$$O_0 = E + A_1 + A_0 = (E'A_1'A_0')'$$

$$O_1 = E + A_1 + A_0' = (E'A_1'A_0)'$$

$$O_2 = E + A_1' + A_0 = (E'A_1A_0')'$$

$$O_3 = E + A_1' + A_0' = (E'A_1A_0)'$$



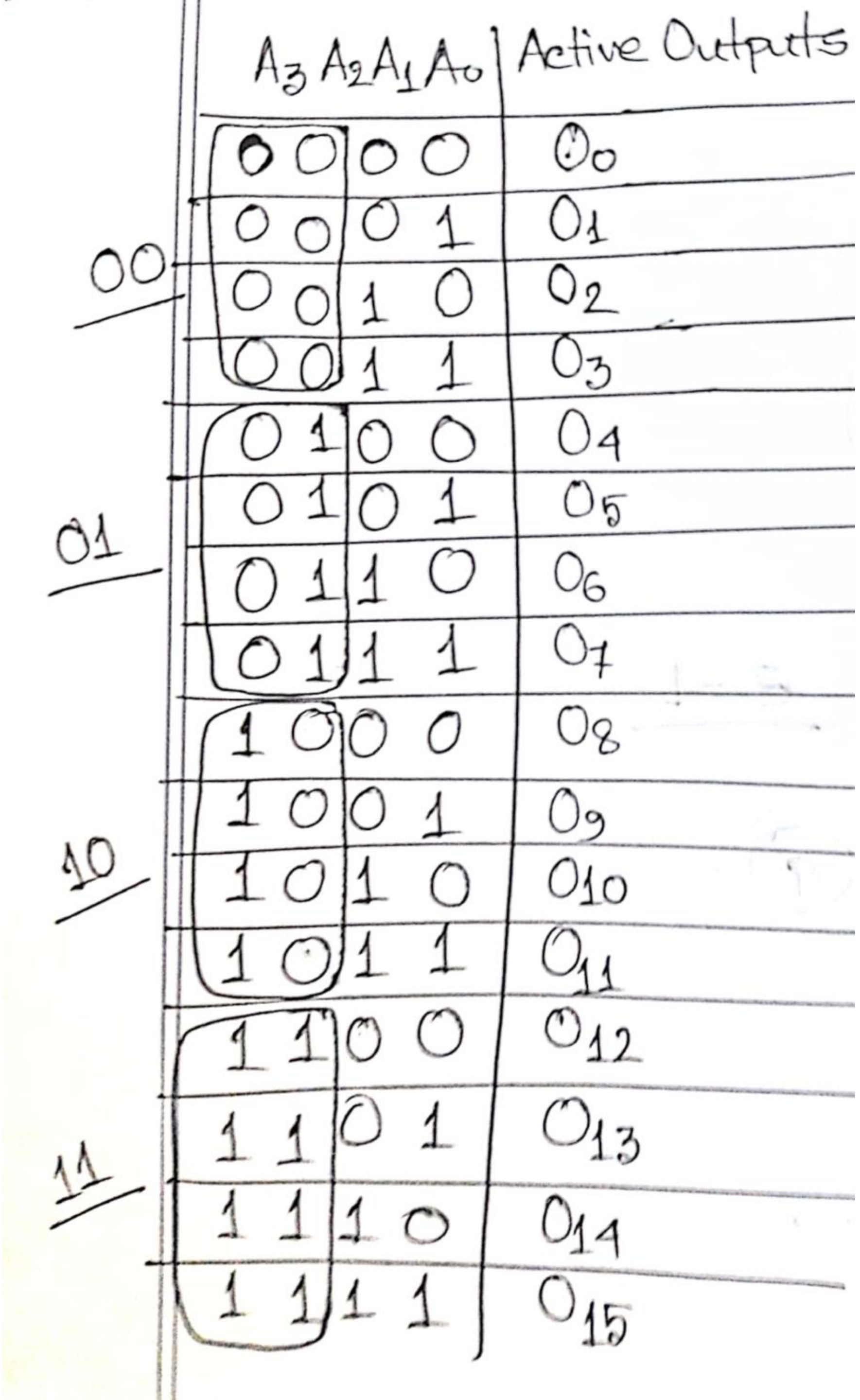
- active-LOW Enable Input and active-HIGH Outputs
- active-HIGH Enable Input and active-HIGH Outputs
- active-HIGH Enable Input and active-LOW Outputs

8.3.3 Expansion of Decoder

Implementation of 4X16 Decoder

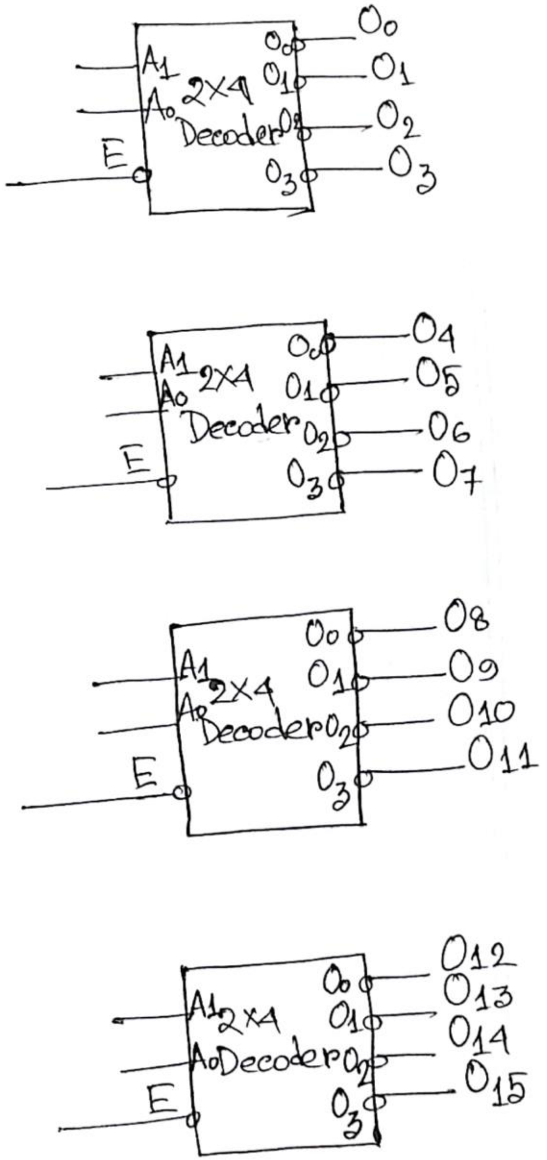
with 2X4 Decoders (active-LOW

Enable Input and active-LOW outputs)

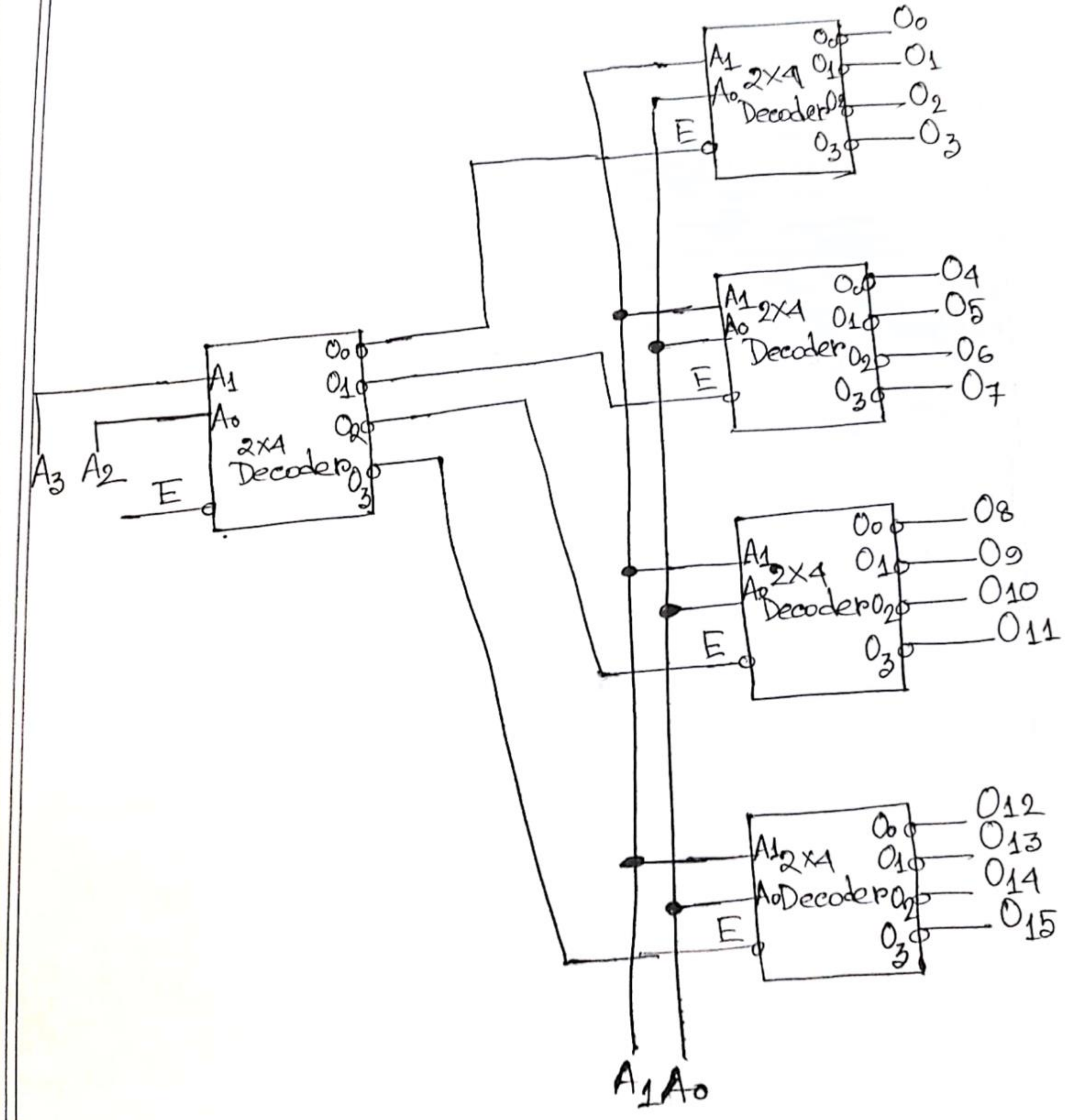


	A_3	A_2	A_1	A_0	Active Outputs
00	0	0	0	0	O_0
	0	0	0	1	O_1
	0	0	1	0	O_2
	0	0	1	1	O_3
01	0	1	0	0	O_4
	0	1	0	1	O_5
	0	1	1	0	O_6
	0	1	1	1	O_7
10	1	0	0	0	O_8
	1	0	0	1	O_9
	1	0	1	0	O_{10}
	1	0	1	1	O_{11}
11	1	1	0	0	O_{12}
	1	1	0	1	O_{13}
	1	1	1	0	O_{14}
	1	1	1	1	O_{15}

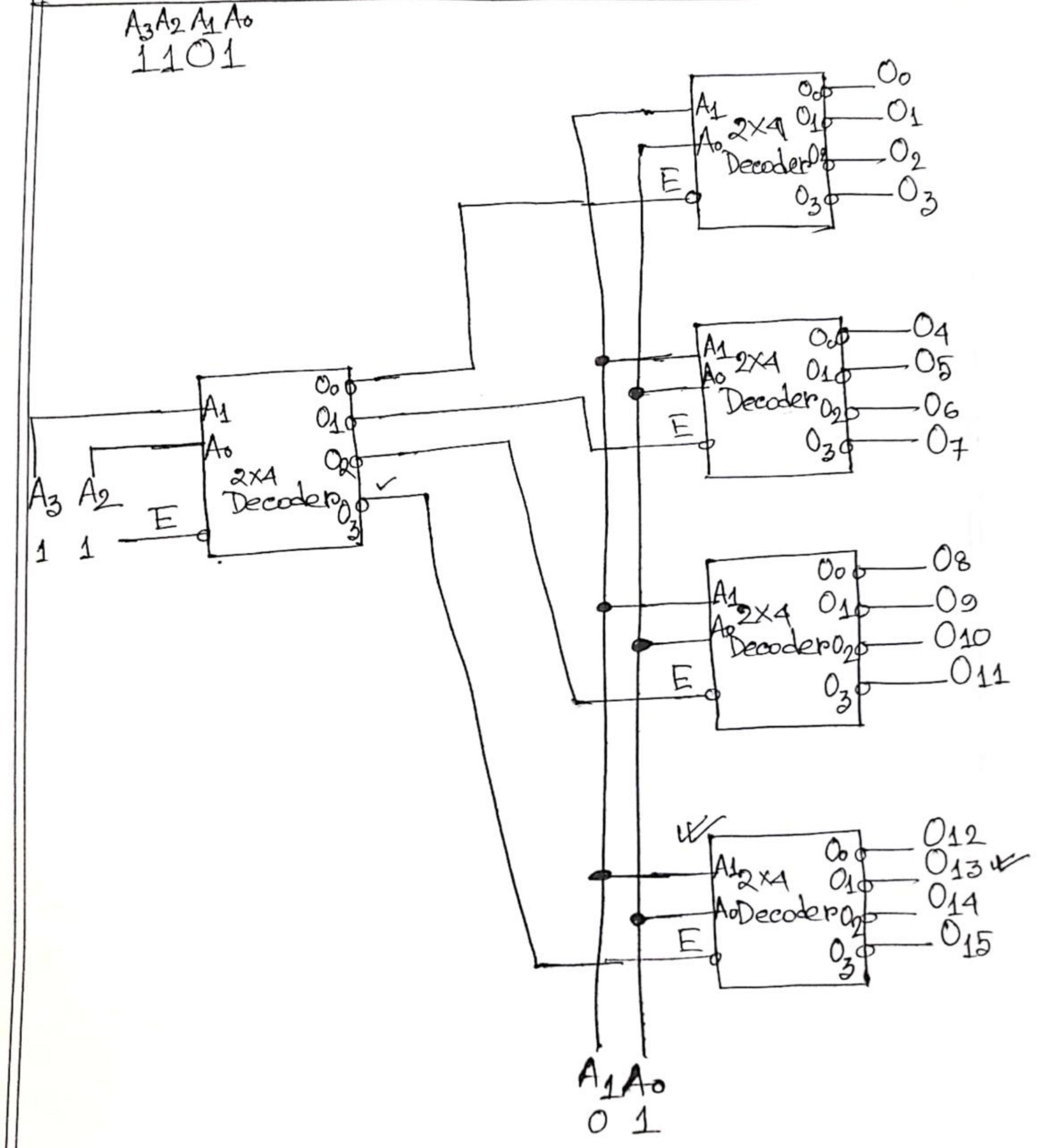
Implementation of 4X16 Decoder with 2X4 Decoders

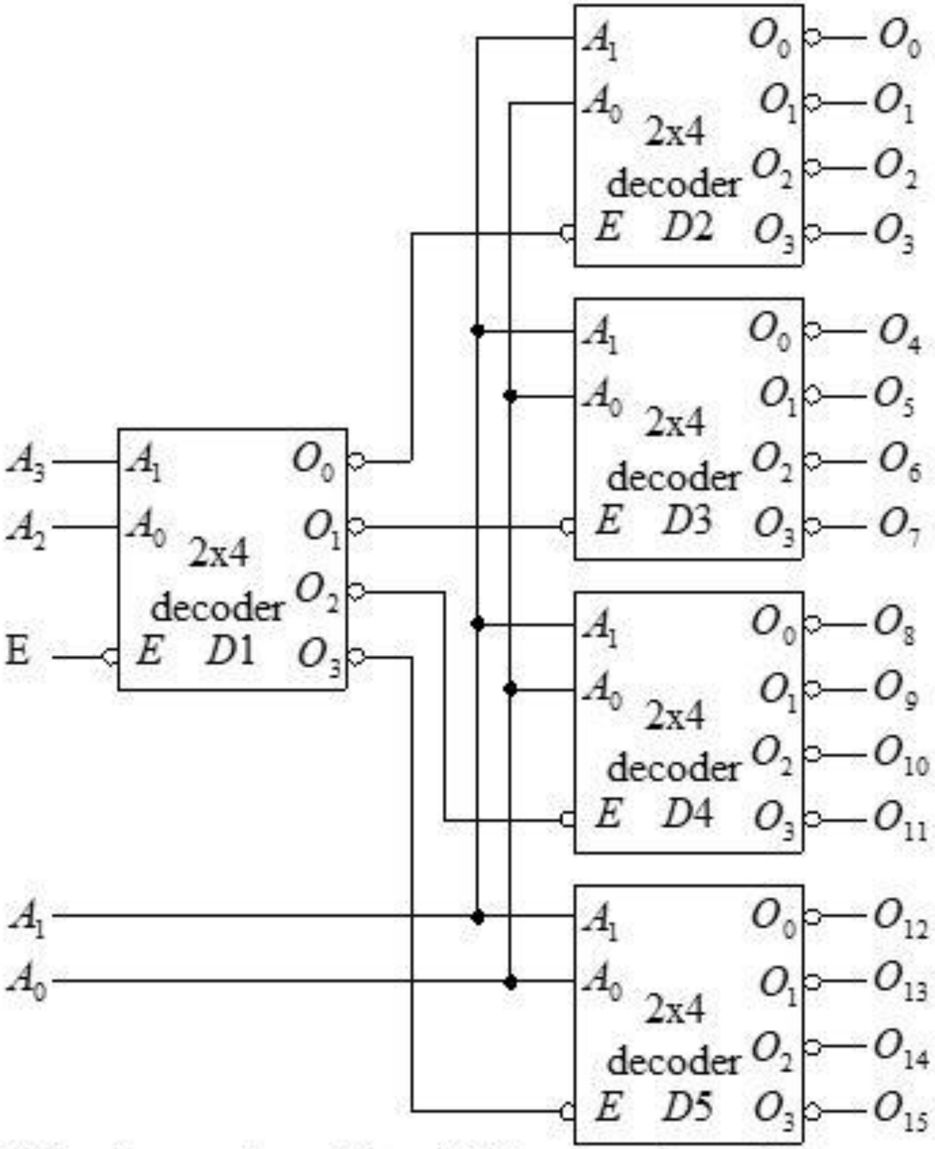


Implementation of 4X16 Decoder with 2X4 Decoders



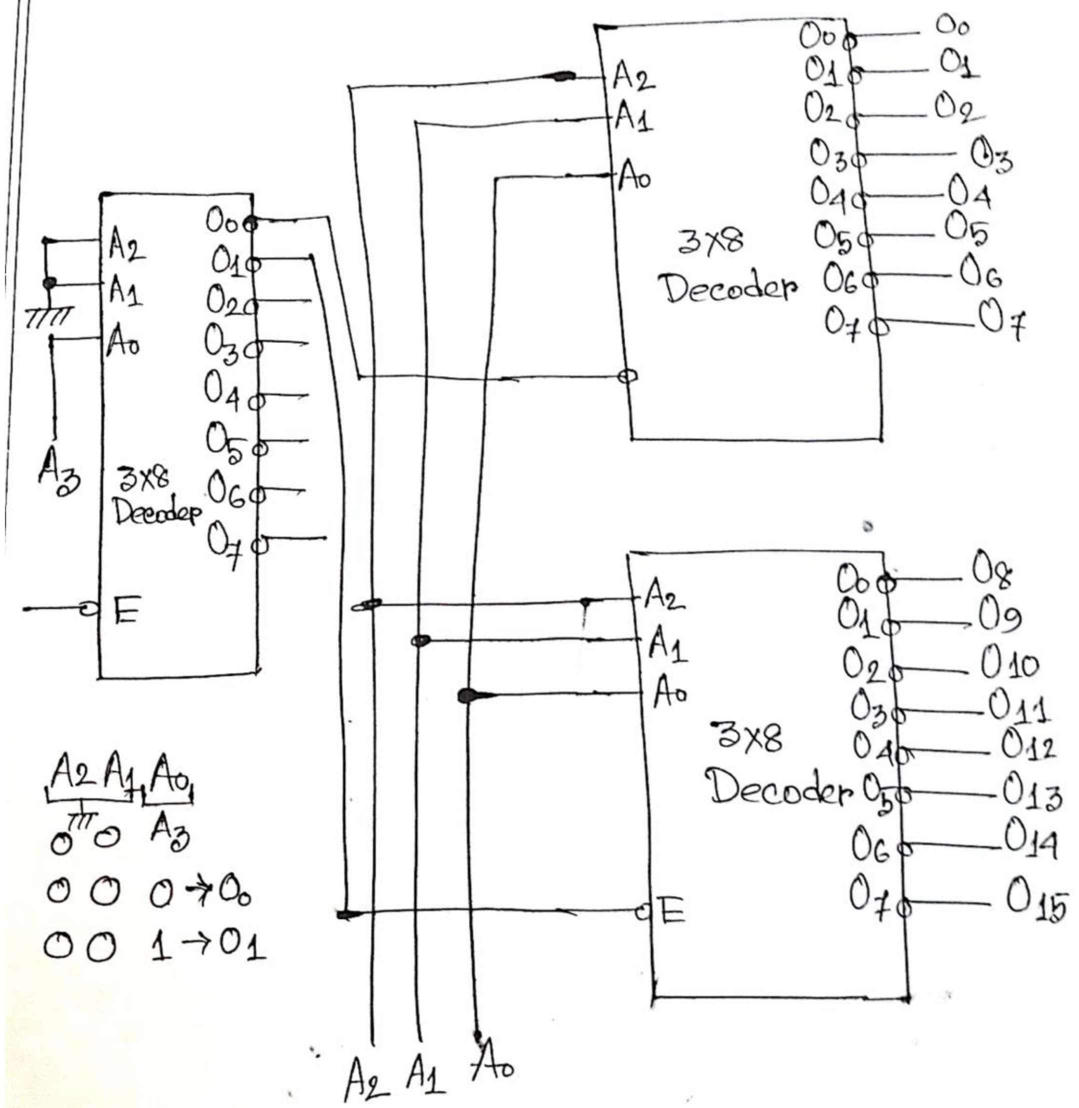
Implementation of 4X16 Decoder with 2X4 Decoders





(b) Implementation of 4-to-16-line decoder with 2-to-4-line decoders

Implementation of 4X16 Decoder with only 3X8 Decoders (active-LOW Enable Input and active-LOW Outputs)

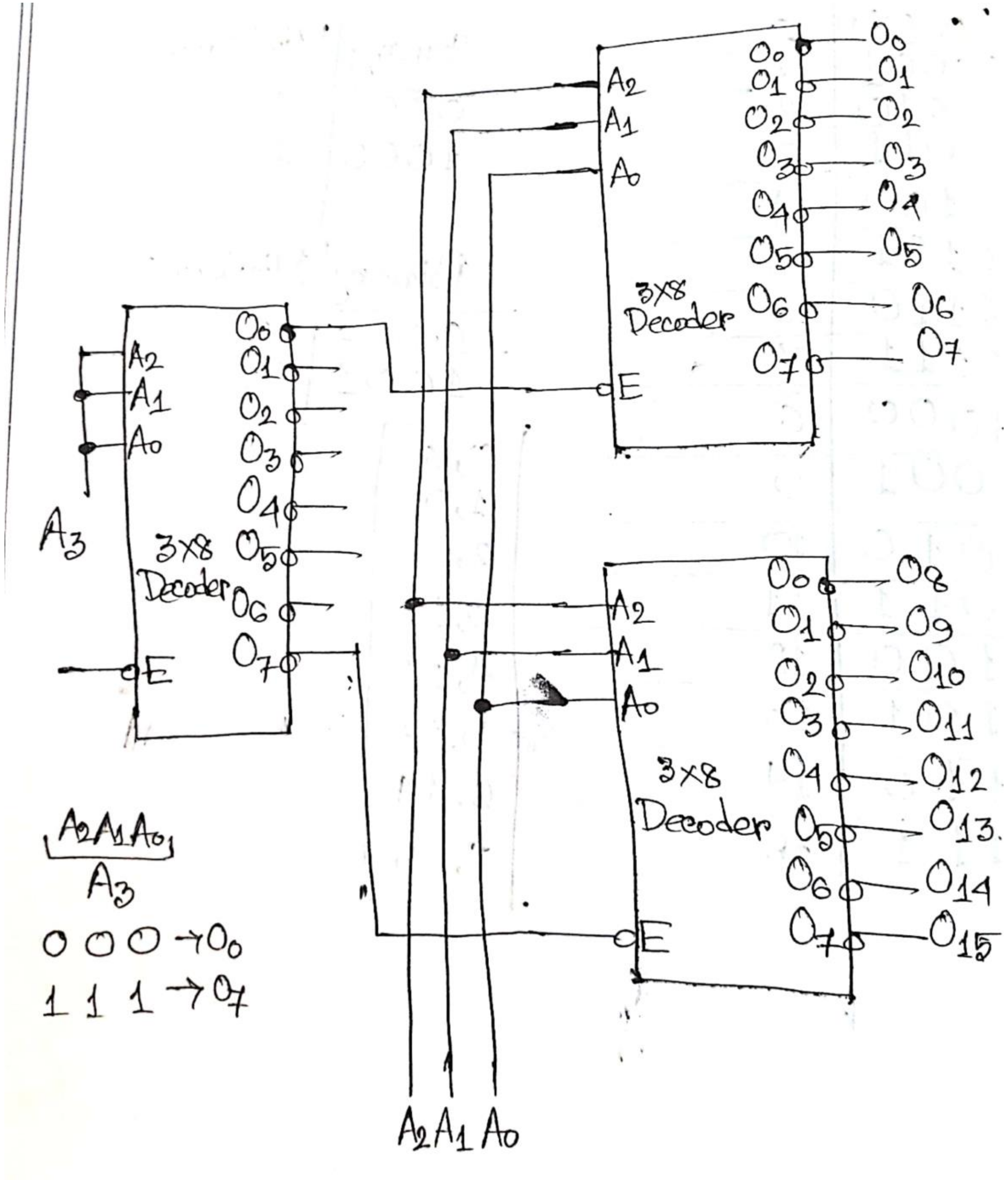


$A_2\ A_1\ A_0$

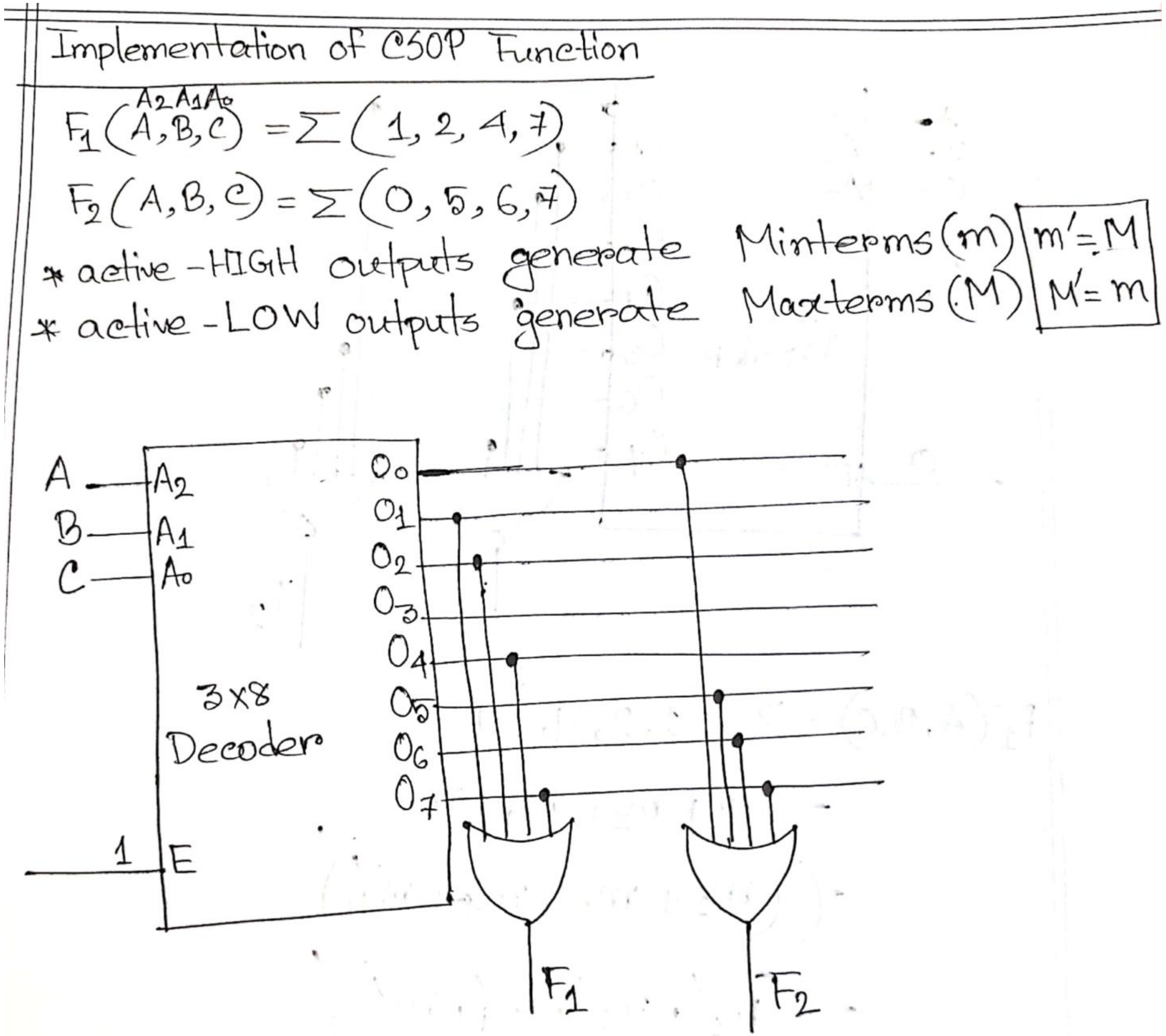
$\underbrace{0\ 0}\ \underbrace{}_{A_3}$

0 0 0 → O_0

0 0 1 → O_1


3X8 Decoder
A2
A1
A0
A3
E
O0
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
O11
O12
O13
O14
O15
A2A1A0
A3
0 0 0 → O0
1 1 1 → O7
A2 A1 A0

8.3.4 Combinational Logic Implementation with Decoder



Implementation of CSOP Function

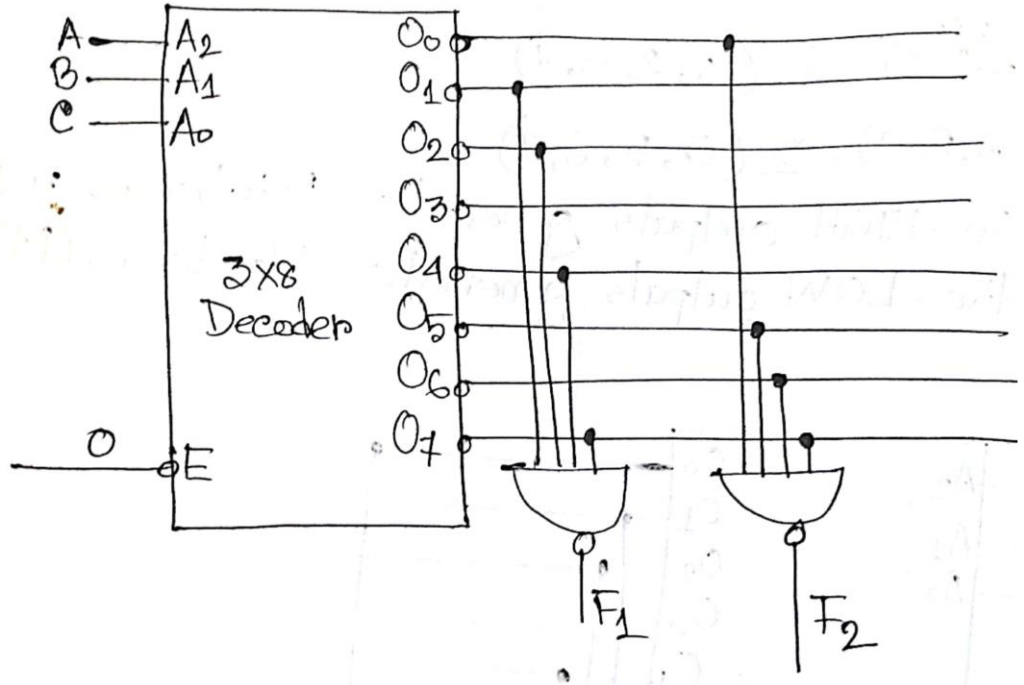
$F_1(\overset{A_2}{A},\overset{A_1}{B},\overset{A_0}{C}) = \Sigma(1, 2, 4, 7)$

$F_2(A, B, C) = \Sigma(0, 5, 6, 7)$

* active-HIGH outputs generate Minterms (m)

* active-LOW outputs generate Maxterms (M)

$\boxed{\begin{array}{l} m' = M \\ M' = m \end{array}}$



$$F_1(A,B,C) = \Sigma(1,2,4,7)$$

$$= m_1 + m_2 + m_4 + m_7$$

$$= \left((m_1 + m_2 + m_4 + m_7)'\right)'$$

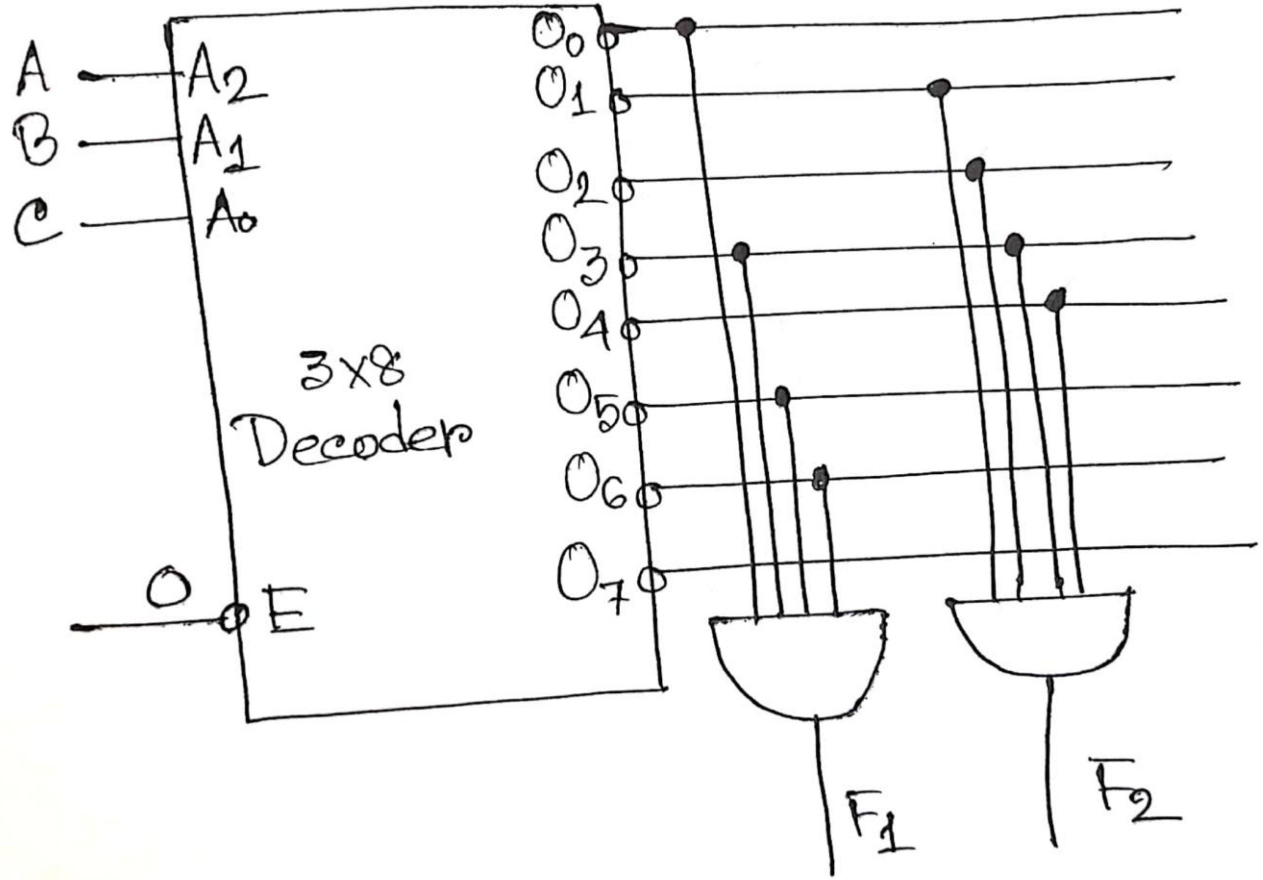
$$= (m_1' \cdot m_2' \cdot m_4' \cdot m_7')'$$

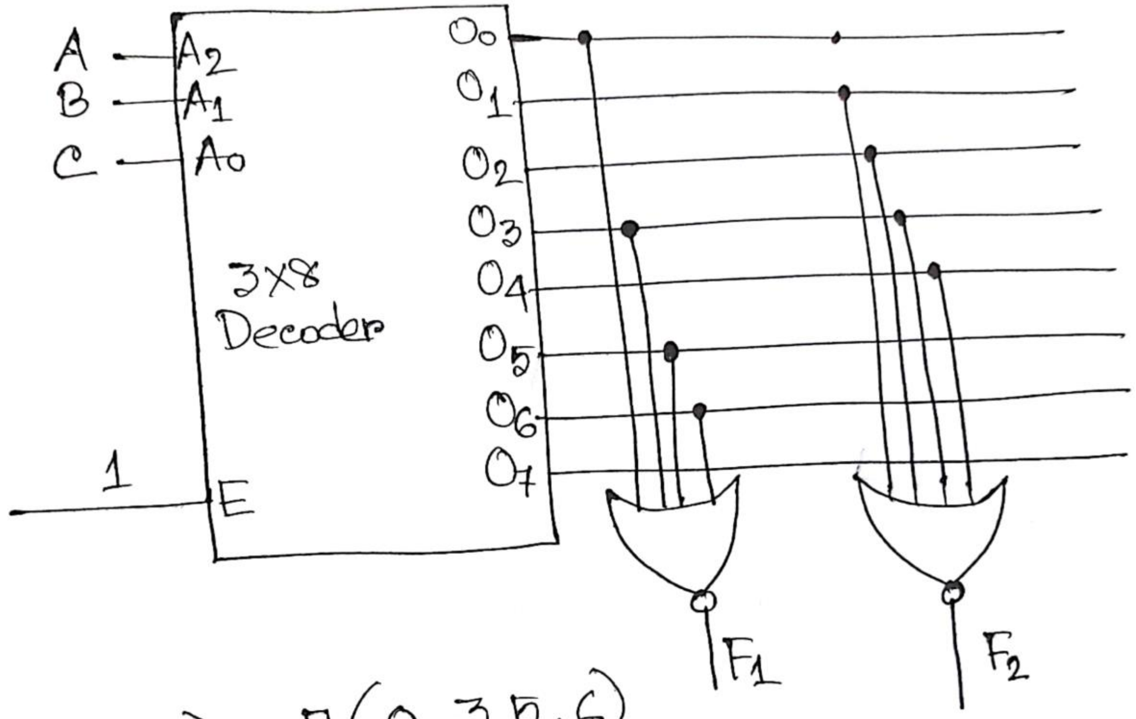
$$= (M_1 \cdot M_2 \cdot M_4 \cdot M_7)'$$

Implementation of CPOS Function

$F_1(A,B,C) = \Pi(0,3,5,6)$

$F_2(A,B,C) = \Pi(1,2,3,4)$





$$F_1(A,B,C) = \Pi(0,3,5,6)$$

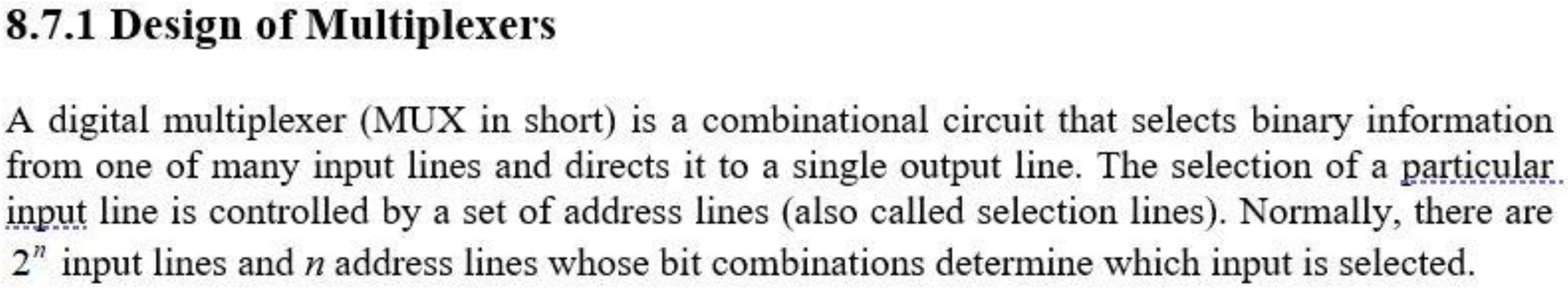
$$= M_0 \cdot M_3 \cdot M_5 \cdot M_6$$

$$= ((M_0 \cdot M_3 \cdot M_5 \cdot M_6)')'$$

$$= (M_0' + M_3' + M_5' + M_6')'$$

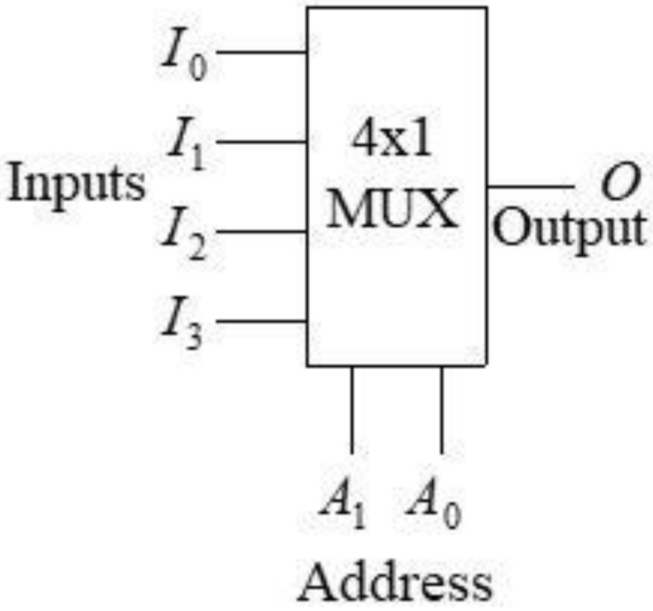
$$= (m_0 + m_3 + m_5 + m_6)'$$

8.7 Multiplexers (MUX)



8.7.1 Design of Multiplexers

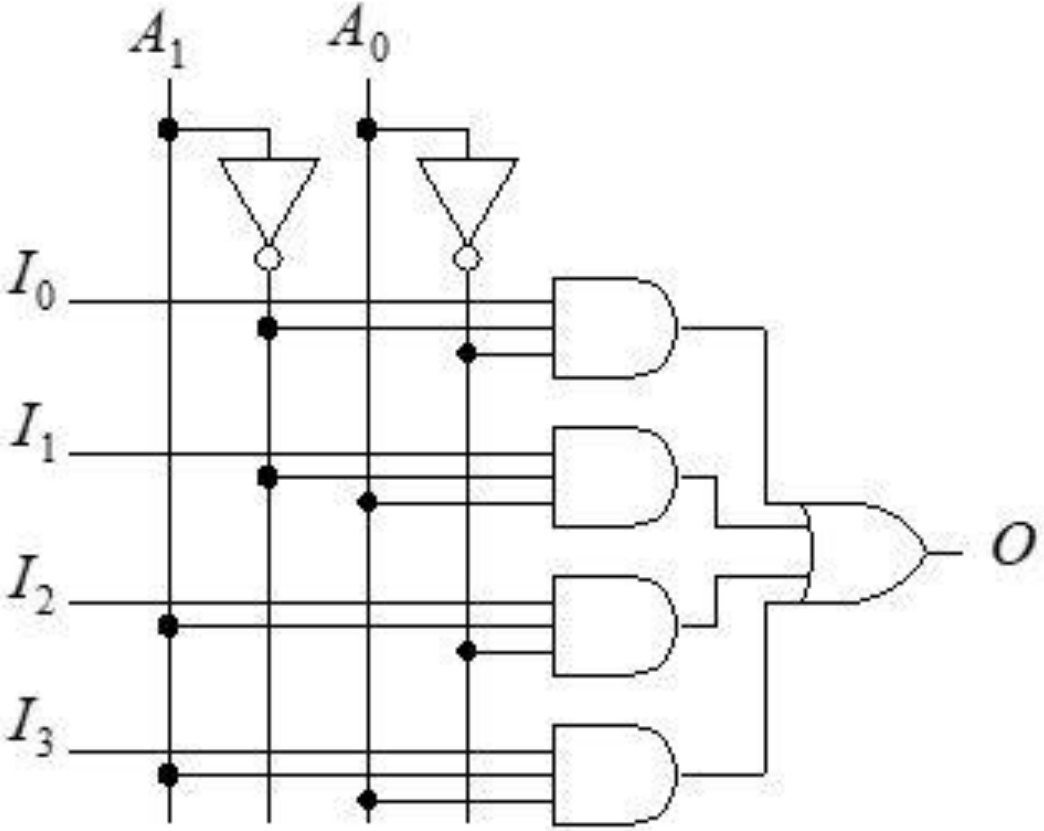
A digital multiplexer (MUX in short) is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of address lines (also called selection lines). Normally, there are 2^n input lines and n address lines whose bit combinations determine which input is selected.



(a) Block diagram

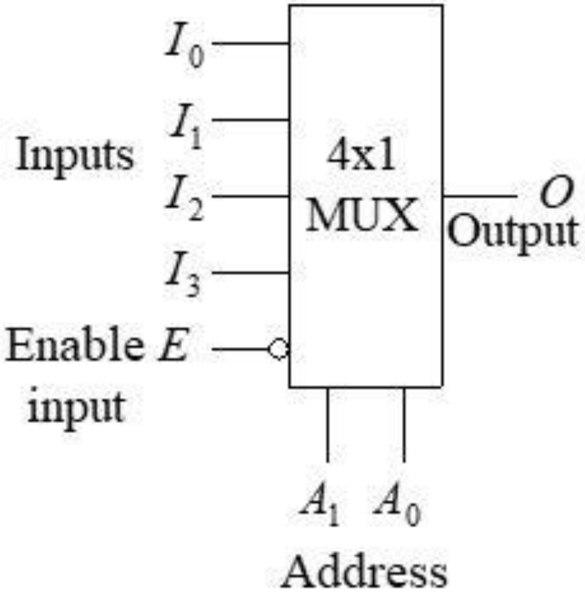
A_1	A_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Truth table

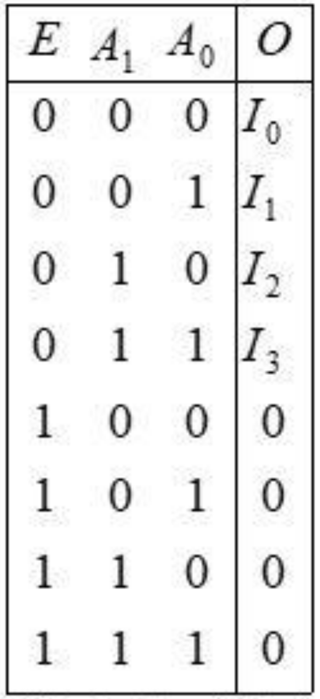


$$O = I_0 A_1' A_0' + I_1 A_1' A_0 + I_2 A_1 A_0' + I_3 A_1 A_0$$

8.7.2 Multiplexer with Enable Input

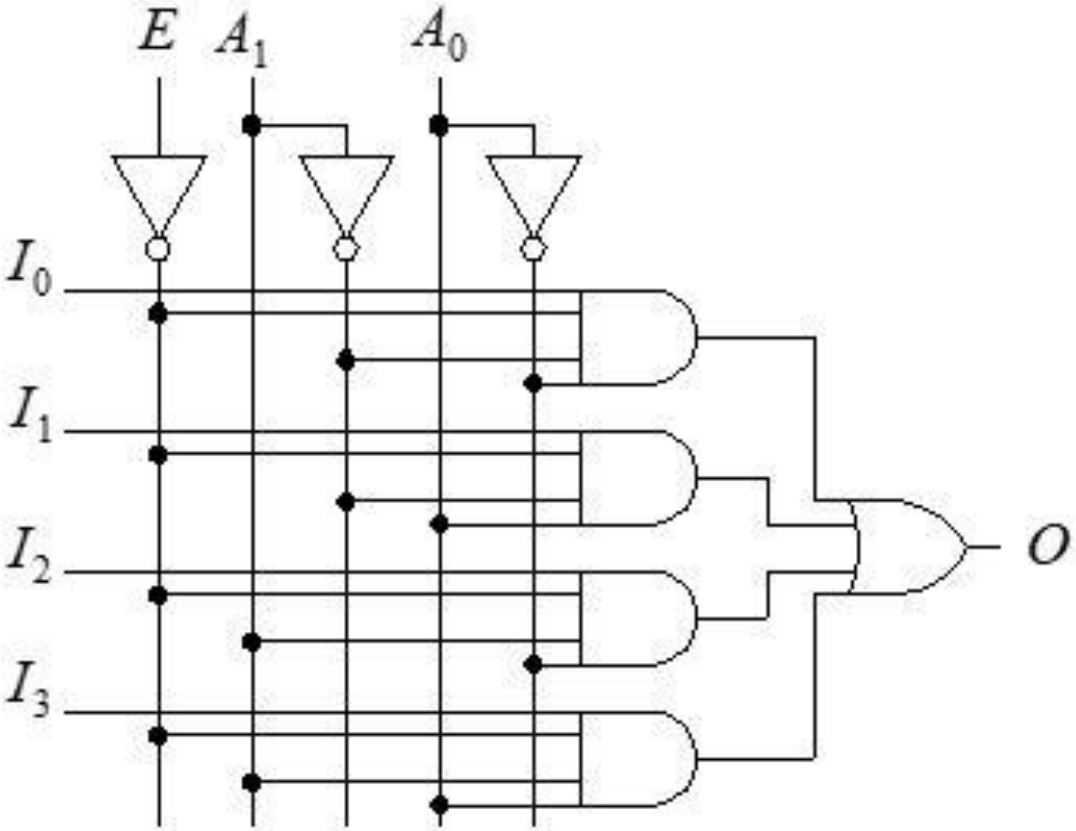


(a) Block diagram

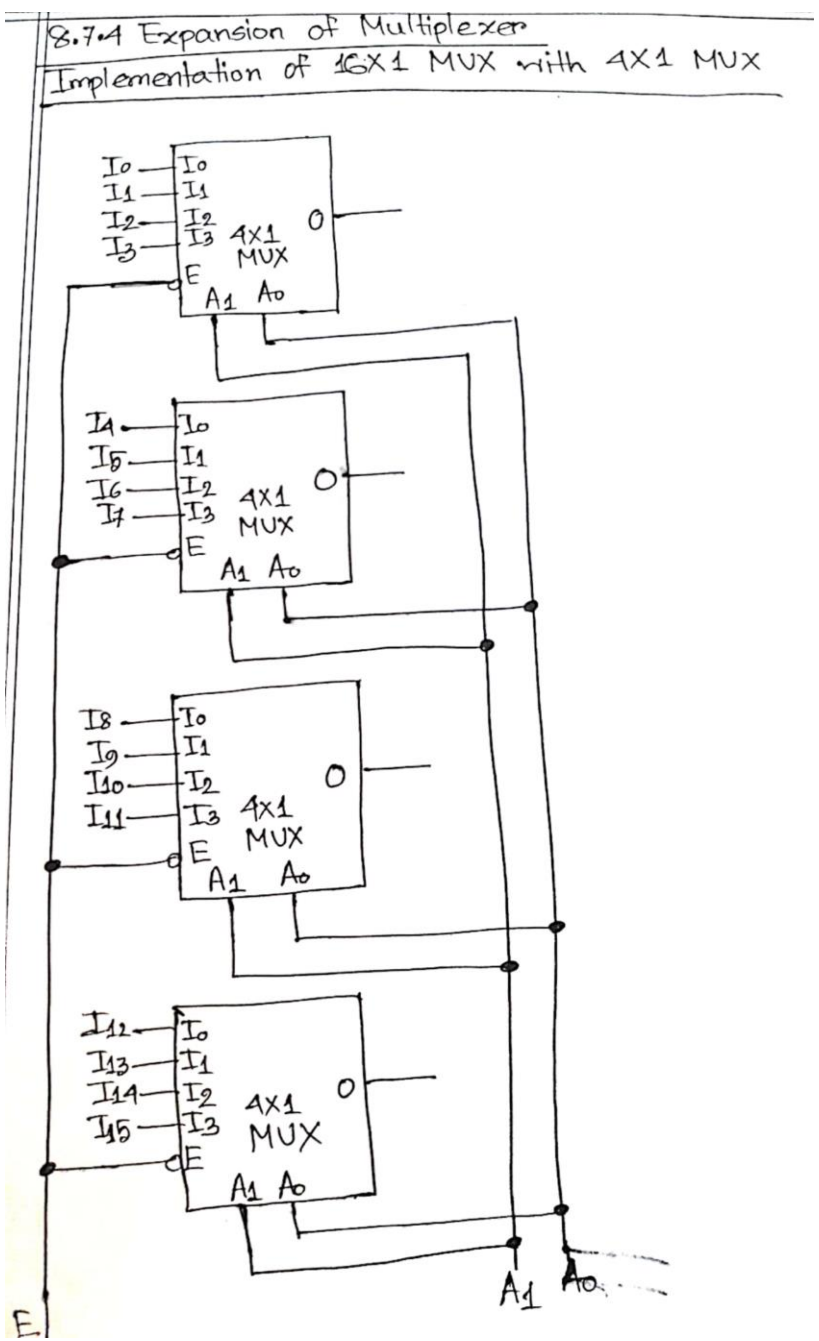


E	A_1	A_0	O
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(b) Truth table

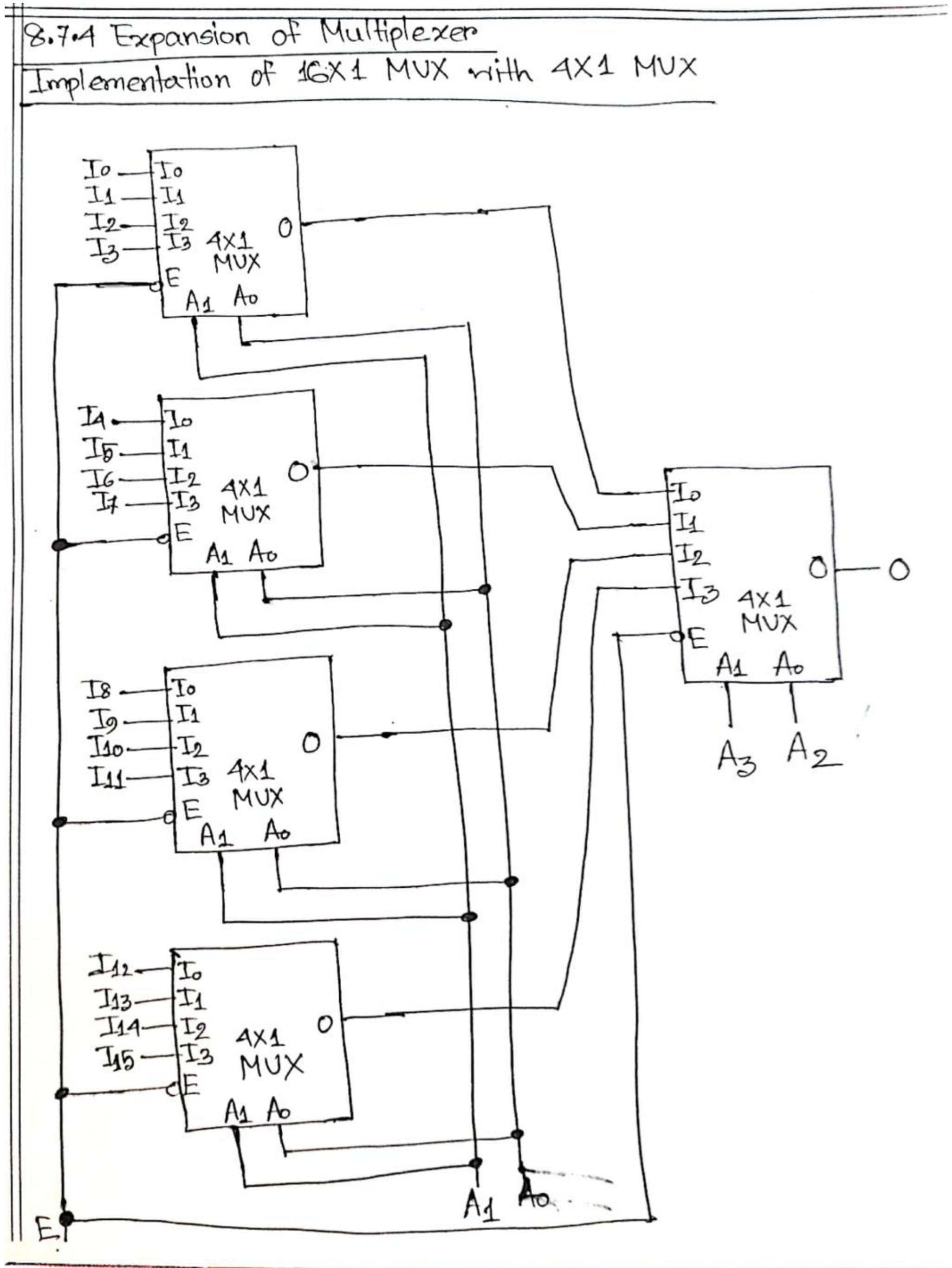


$$O = I_0E'A_1'A_0' + I_1E'A_1'A_0 + I_2E'A_1A_0' + I_3E'A_1A_0$$



8.7.4 Expansion of Multiplexer

Implementation of 16X1 MUX with 4X1 MUX

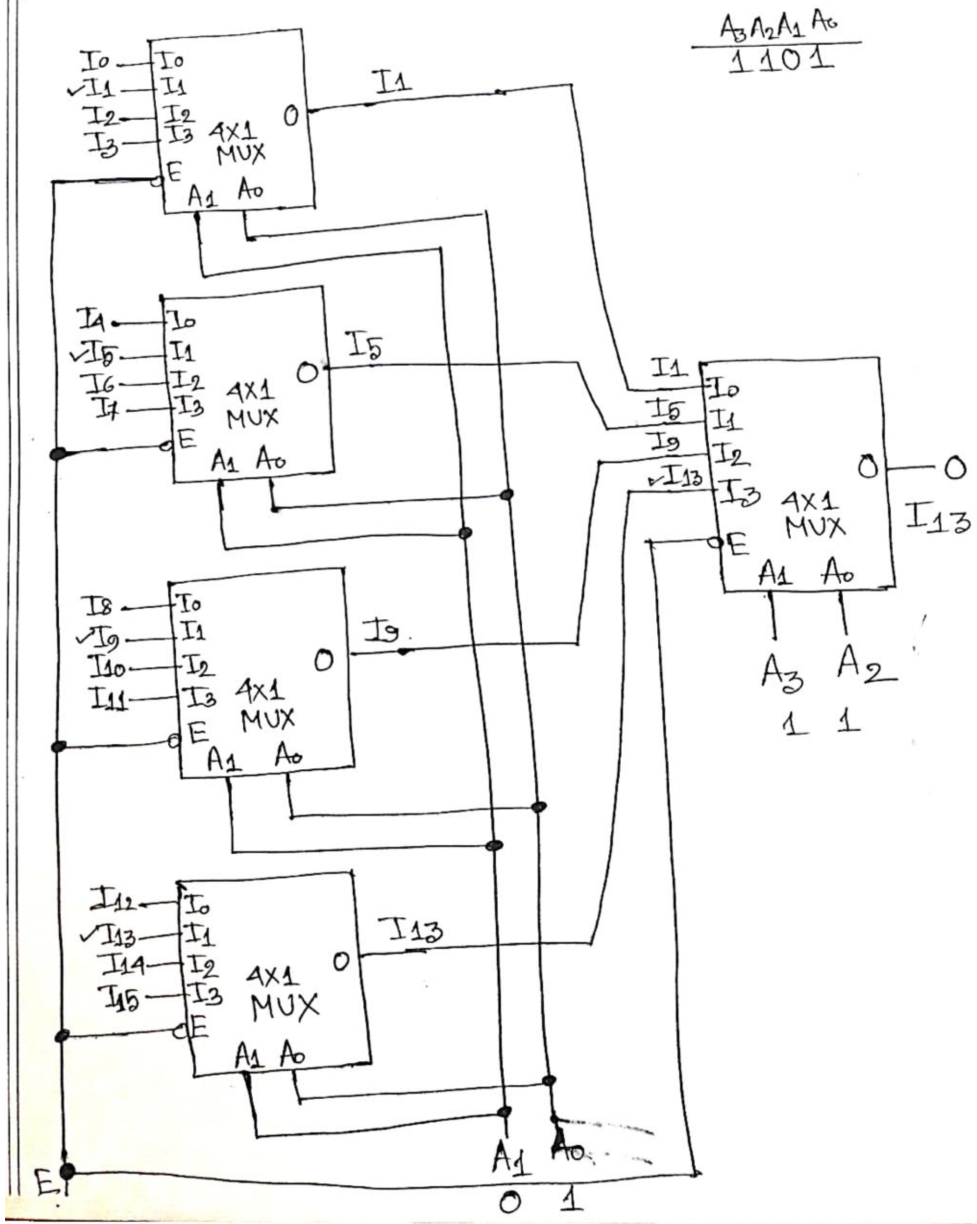


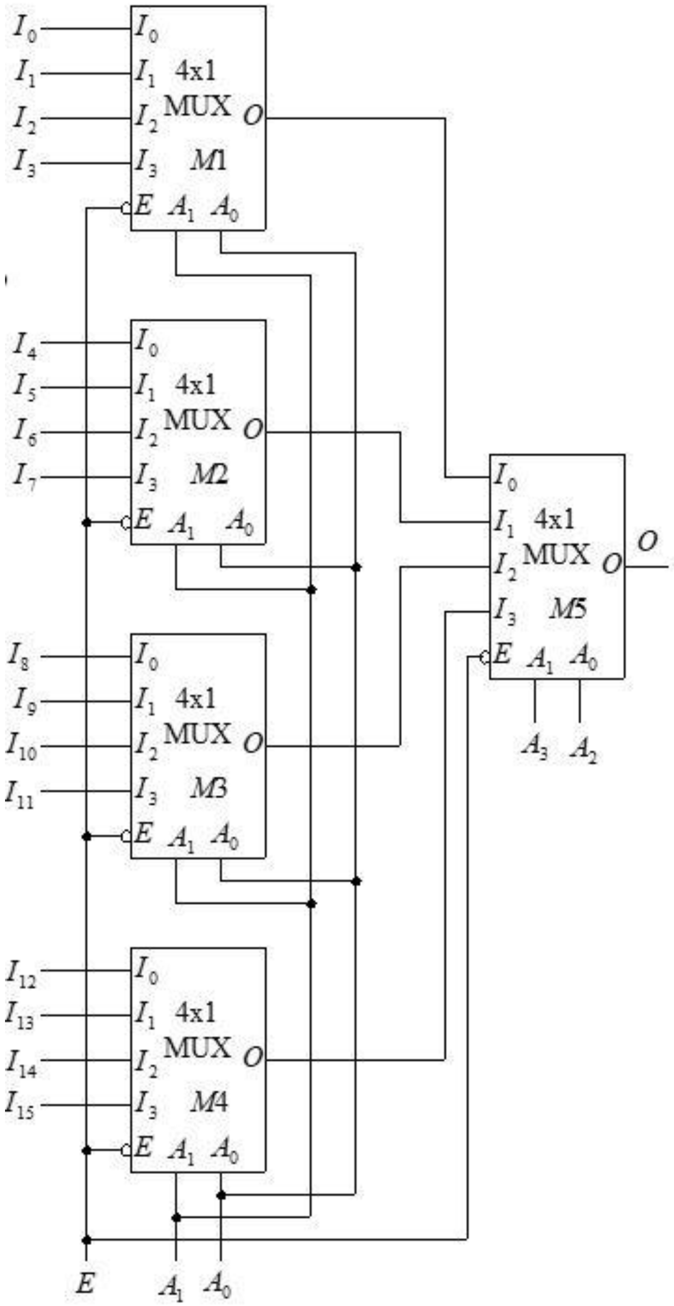
8.7.4 Expansion of Multiplexer

Implementation of 16X1 MUX with 4X1 MUX

8.7.4 Expansion of Multiplexer

Implementation of 16X1 MUX with 4X1 MUX

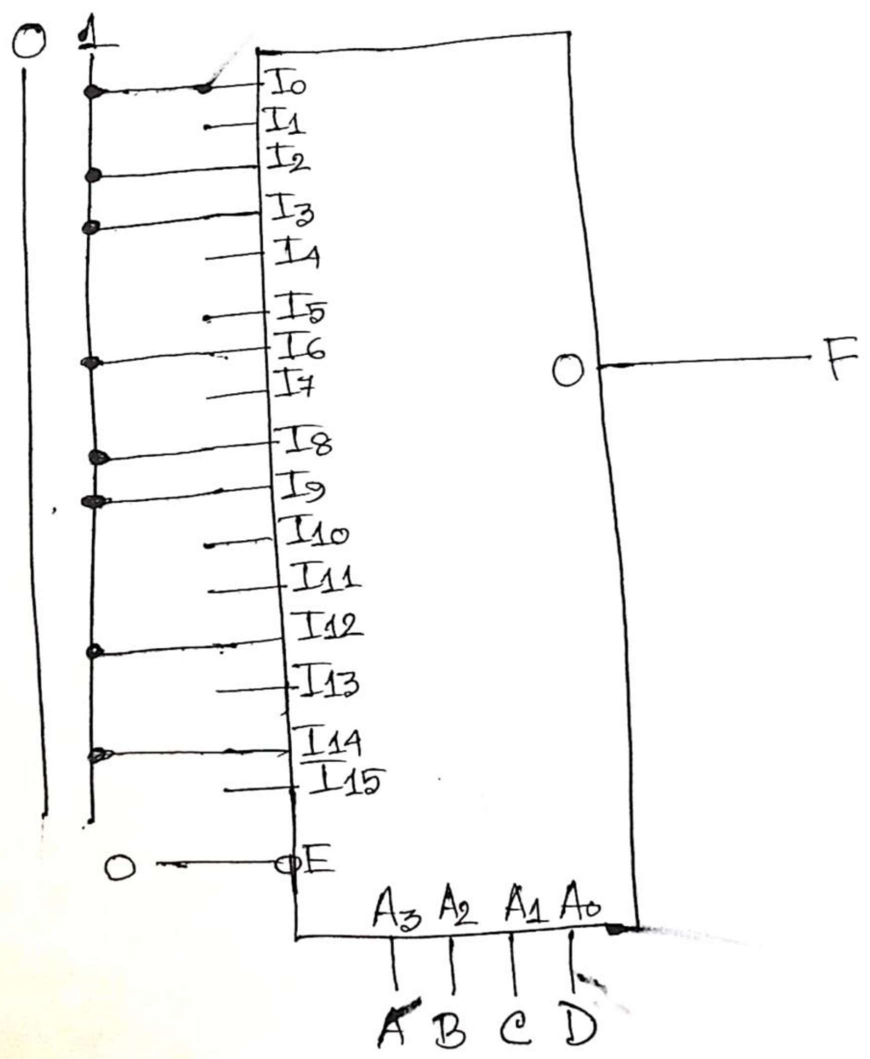




I_0
I_1
I_2
I_3
I_0
I_1
4x1
MUX
I_2
O
I_3
$M1$
E
A_1
A_0
I_4
I_5
I_6
I_7
I_0
I_1
4x1
MUX
I_2
O
I_3
$M2$
E
A_1
A_0
I_8
I_9
I_{10}
I_{11}
I_0
I_1
4x1
MUX
I_2
O
I_3
$M3$
E
A_1
A_0
I_{12}
I_{13}
I_{14}
I_{15}
I_0
I_1
4x1
MUX
I_2
O
I_3
$M4$
E
A_1
A_0
I_0
I_1
4x1
MUX
I_2
O
I_3
$M5$
E
A_1
A_0
O
A_3
A_2
E
A_1
A_0

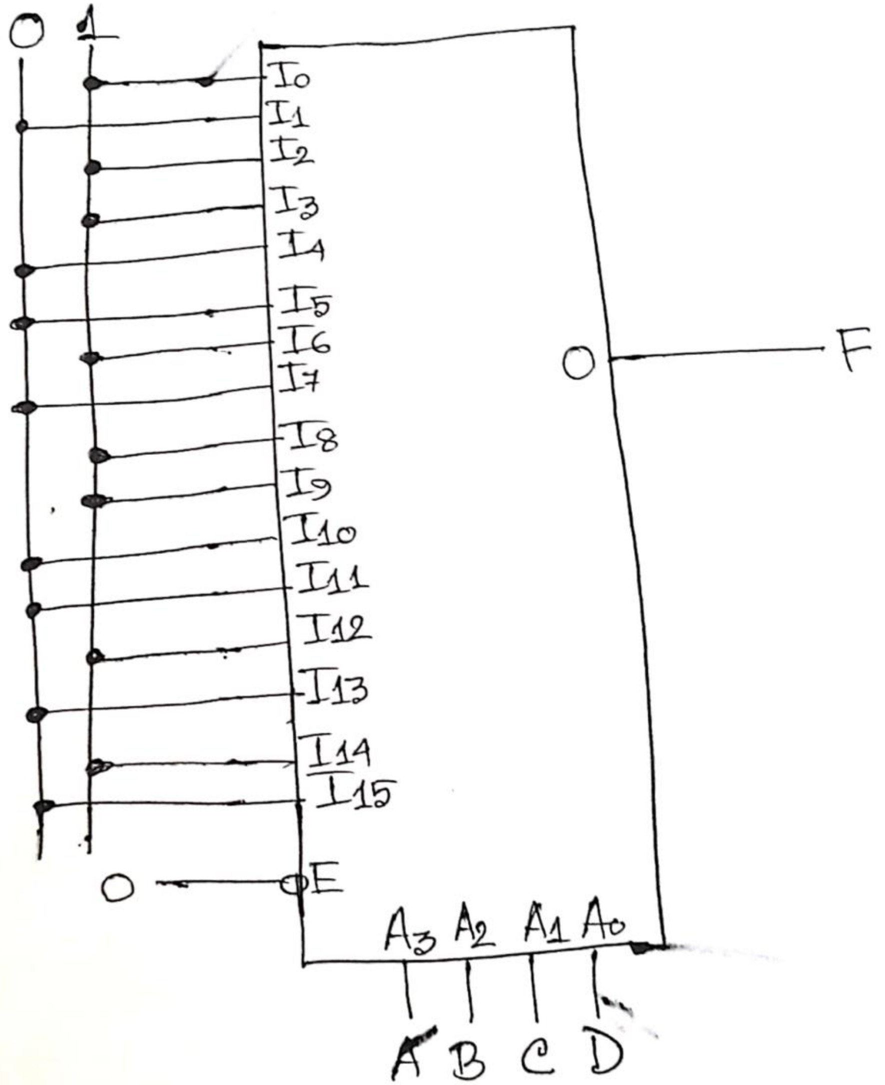
8.7.5 Combinational Logic Implementation with MUX

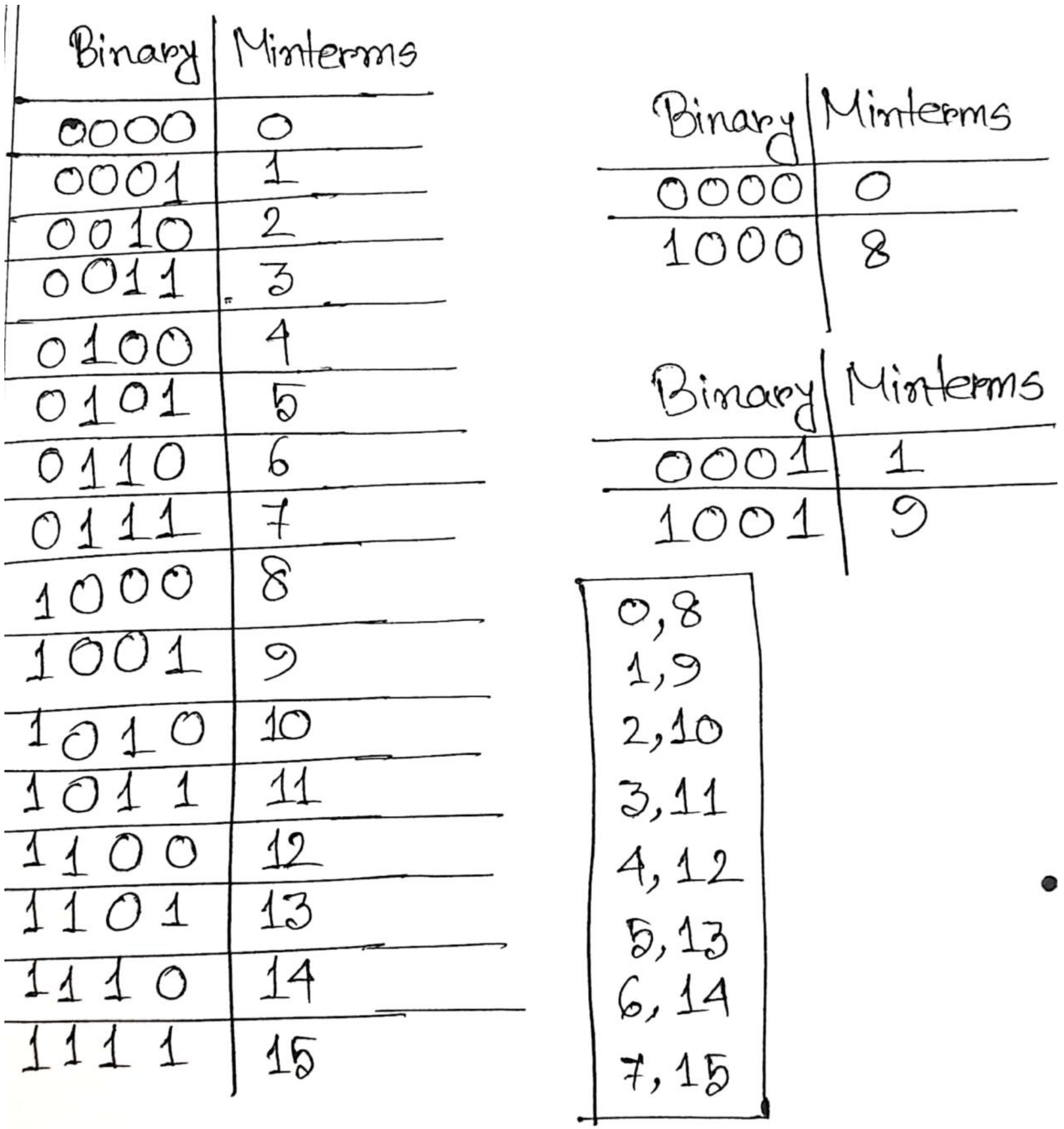
$F(A,B,C,D) = \sum(0, 2, 3, 6, 8, 9, 12, 14)$



8.7.5 Combinational Logic Implementation with MUX

$F(A,B,C,D) = \sum(0, 2, 3, 6, 8, 9, 12, 14)$



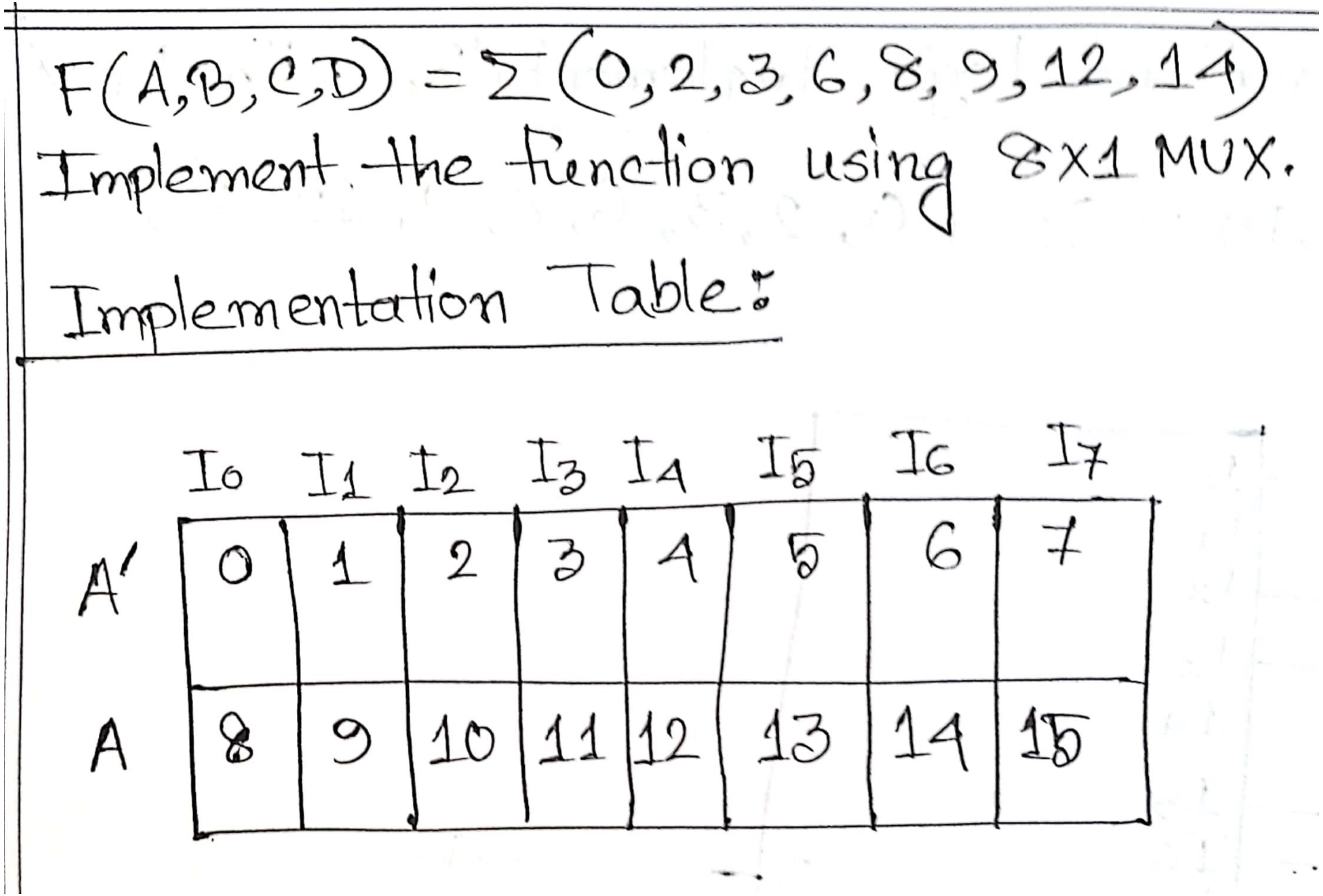


Binary	Minterms
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Binary	Minterms
0000	0
1000	8

Binary	Minterms
0001	1
1001	9

0,8
1,9
2,10
3,11
4,12
5,13
6,14
7,15



$F(A,B,C,D) = \sum(0,2,3,6,8,9,12,14)$

Implement the function using 8X1 MUX.

Implementation Table:

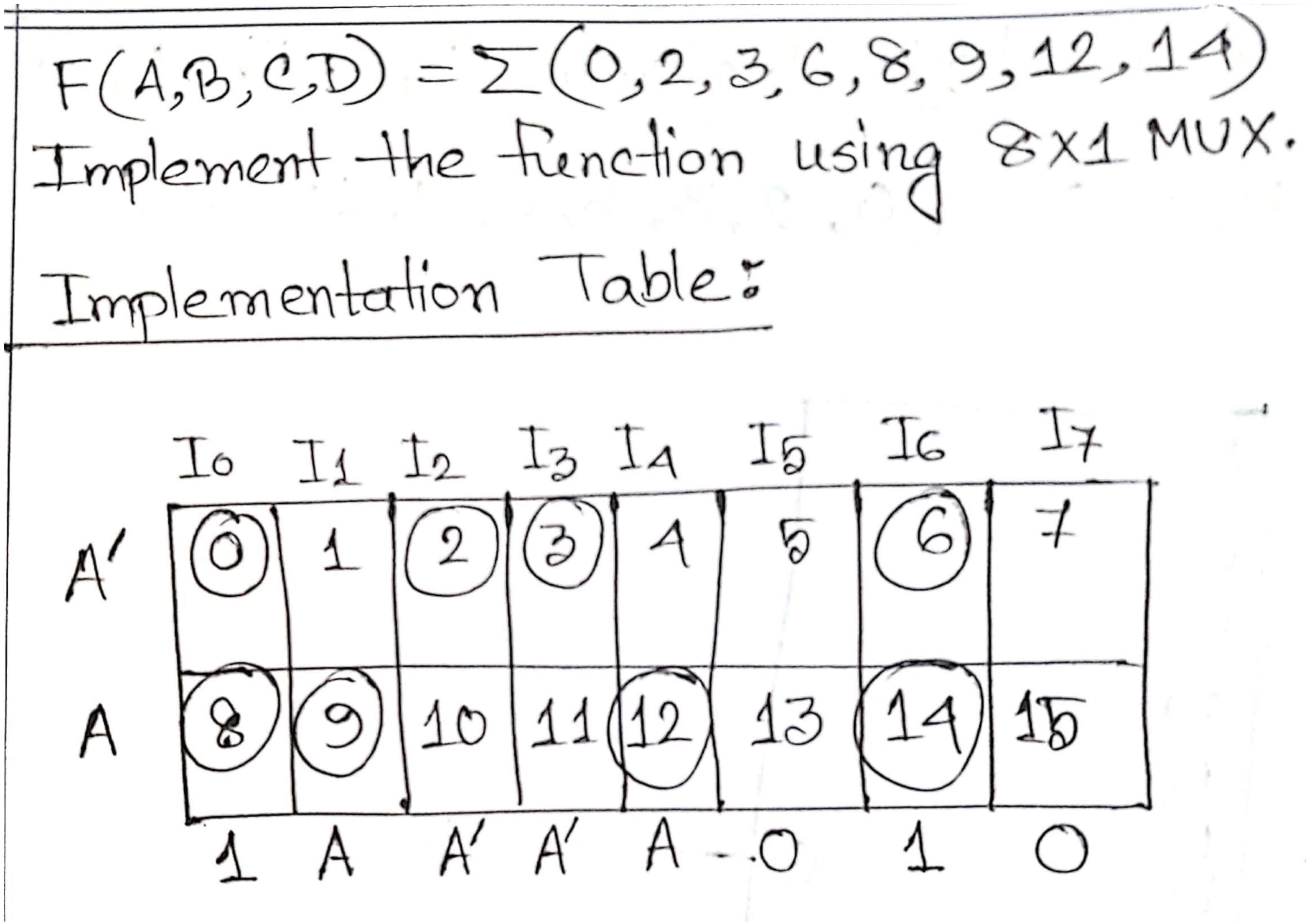
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15

$F(A,B,C,D) = \Sigma(0,2,3,6,8,9,12,14)$

Implement the function using 8X1 MUX.

Implementation Table:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	⓪	1	②	③	4	5	⑥	7
A	⑧	⑨	10	11	⑫	13	⑭	15
	1					0	1	0

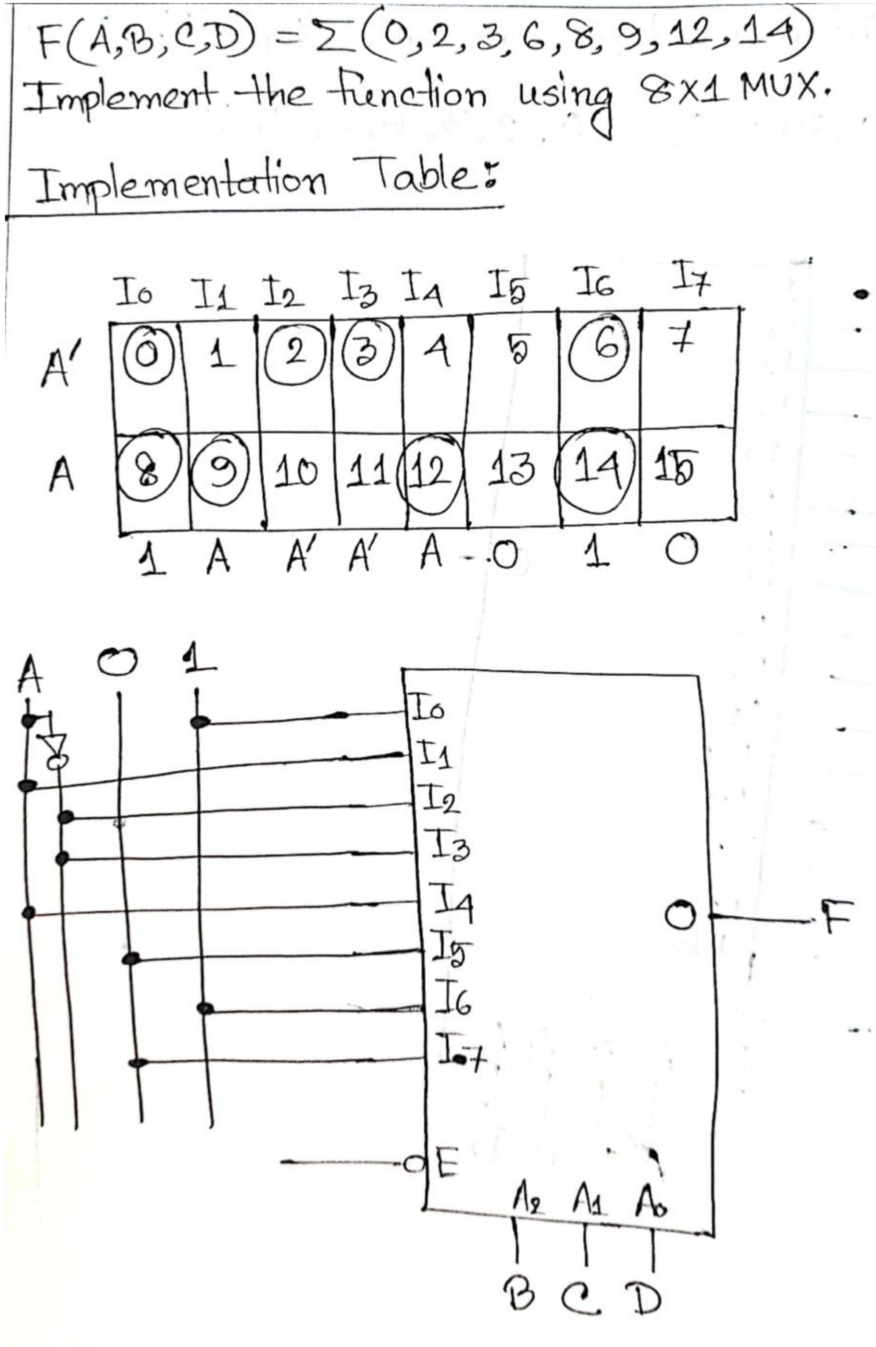


$F(A,B,C,D) = \Sigma(0,2,3,6,8,9,12,14)$

Implement the function using 8X1 MUX.

Implementation Table:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	A	A'	A'	A	0	1	0



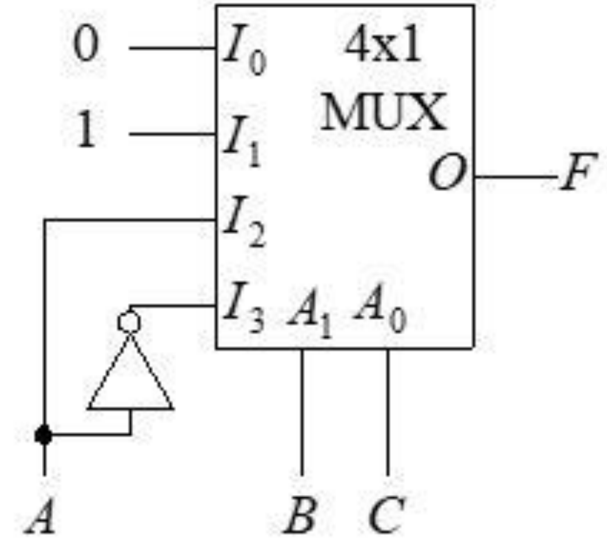
$F(A,B,C,D) = \Sigma(0,2,3,6,8,9,12,14)$

Implement the function using 8X1 MUX.

Implementation Table:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	A	A'	A'	A	0	1	0

$$F(A,B,C)=\sum(1,3,5,6)$$



(a) Implementation with multiplexer

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(b) Truth table

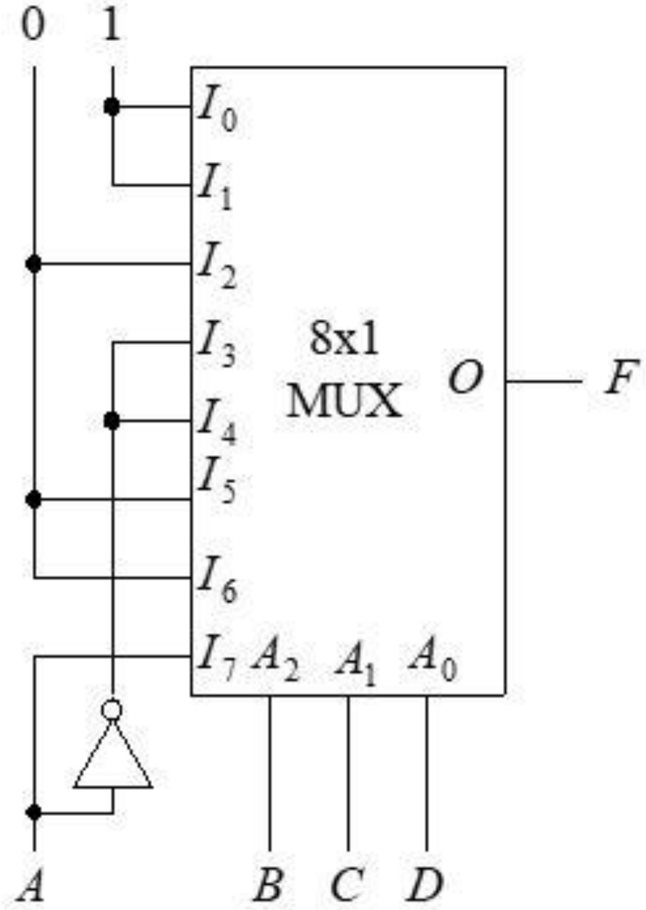
	I_0	I_1	I_2	I_3
A'	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	A'

(c) mplementation table

$$F(A,B,C,D)=\sum(0,1,3,4,8,9,15)$$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	0	A

(a) Implementation table



(b) Implementation with multiplexer