Digital Systems

18B11EC213

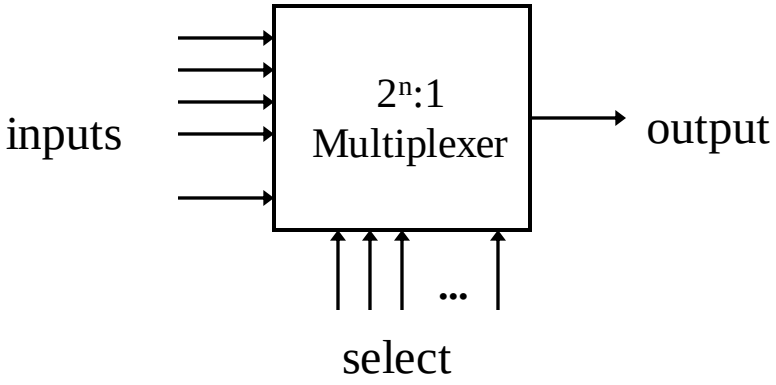
Combinational Circuits

Contents

- Multiplexer
- Demultiplexer
- Comparator
- Code Converter
  - 4 Bit Binary to Gray
  - BCD to Excess-3

Multiplexer

- A multiplexer is a device which has
  (i) a number of *input* lines
  (ii) a number of *selection* lines
  (iii) one *output* line
- It steers one of 2^n inputs to a single output line, using n selection lines. Also known as a *data selector*.

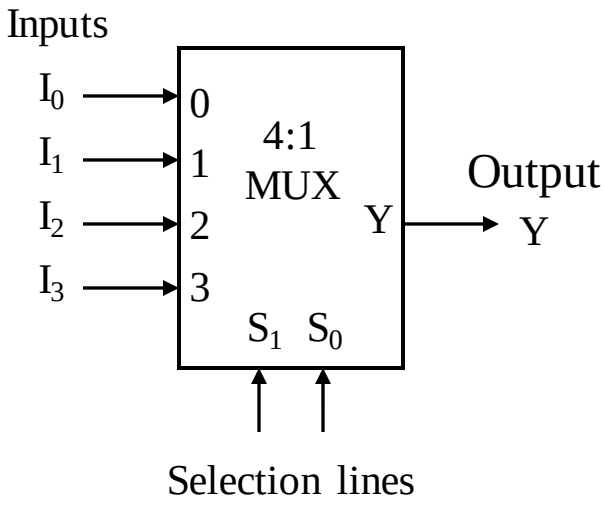


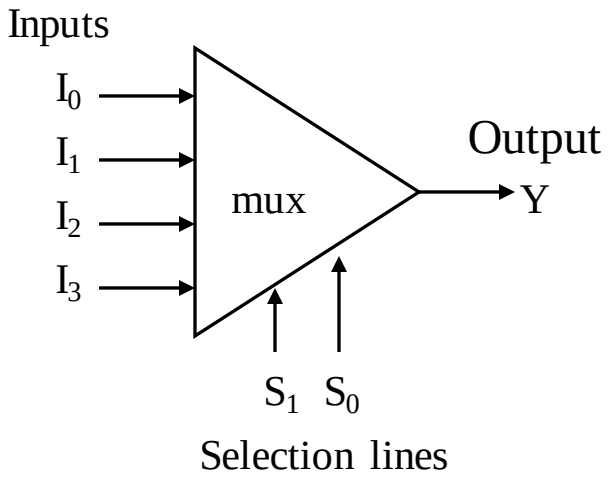
Multiplexer

- Truth table for a 4-to-1 multiplexer:

I_0	I_1	I_2	I_3	S_1	S_0	Y
d_0	d_1	d_2	d_3	0	0	d_0
d_0	d_1	d_2	d_3	0	1	d_1
d_0	d_1	d_2	d_3	1	0	d_2
d_0	d_1	d_2	d_3	1	1	d_3

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3





Multiplexer

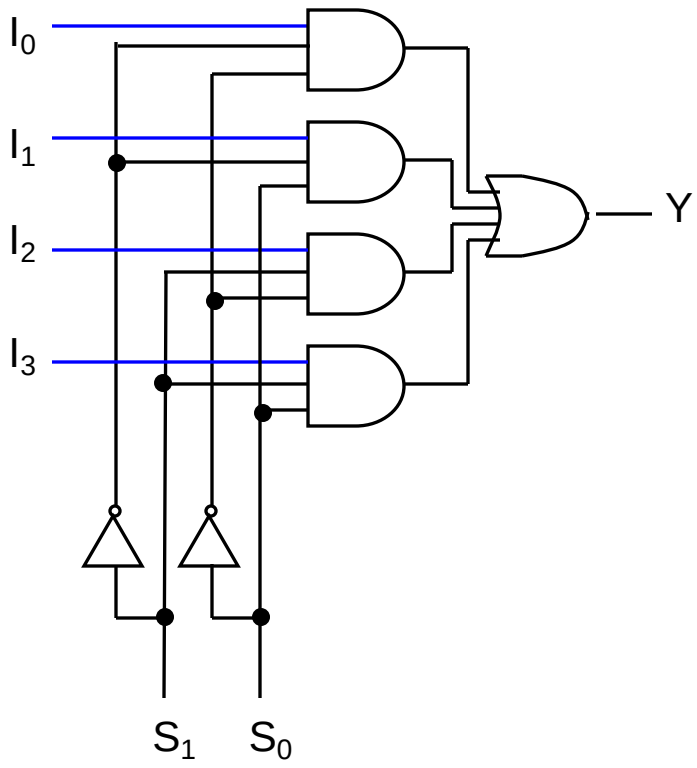
- Output of multiplexer is

  "sum of the (product of *data lines* and *selection lines*)"

- Example: the output of a 4-to-1 multiplexer is:

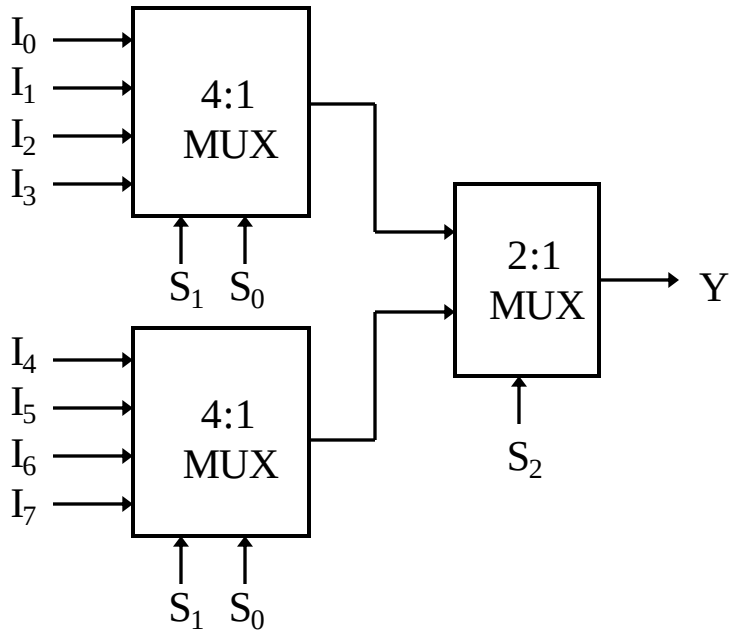
$$Y = I_0.(S_1'.S_0') + I_1.(S_1'.S_0) + I_2.(S_1.S_0') + I_3.(S_1.S_0)$$

$$= m_oI_o + m_1I_1 + m_2I_2 + m_3I_3$$



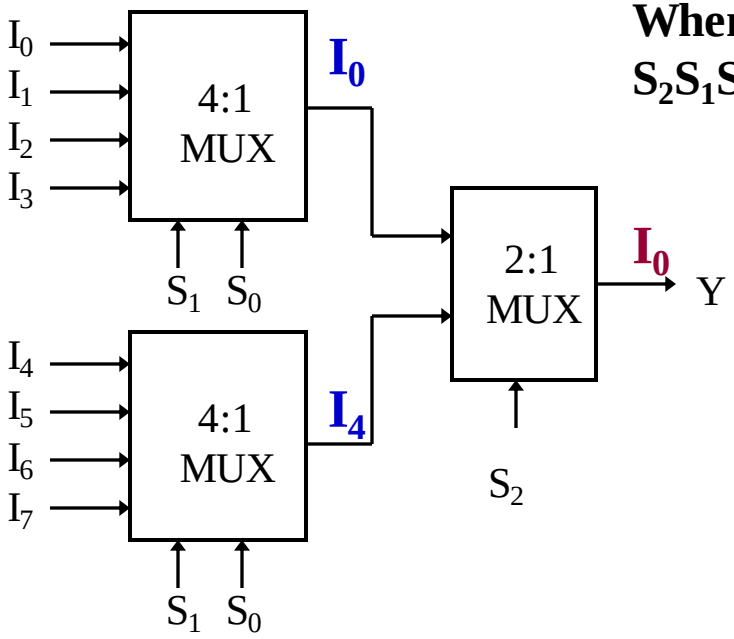
Larger Multiplexers

- Larger multiplexers can be constructed from smaller ones.
- An 8-to-1 multiplexer can be constructed from smaller multiplexers like this (note placement of selector lines):



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Larger Multiplexers

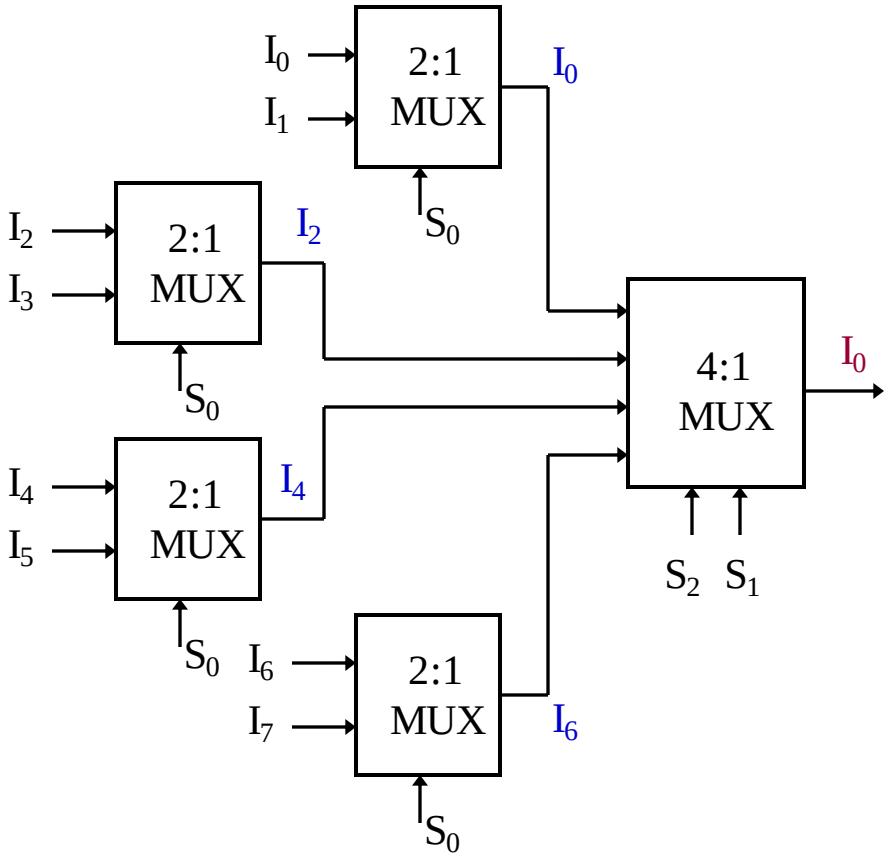


When
$S_2S_1S_0$ = 000

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Larger Multiplexers

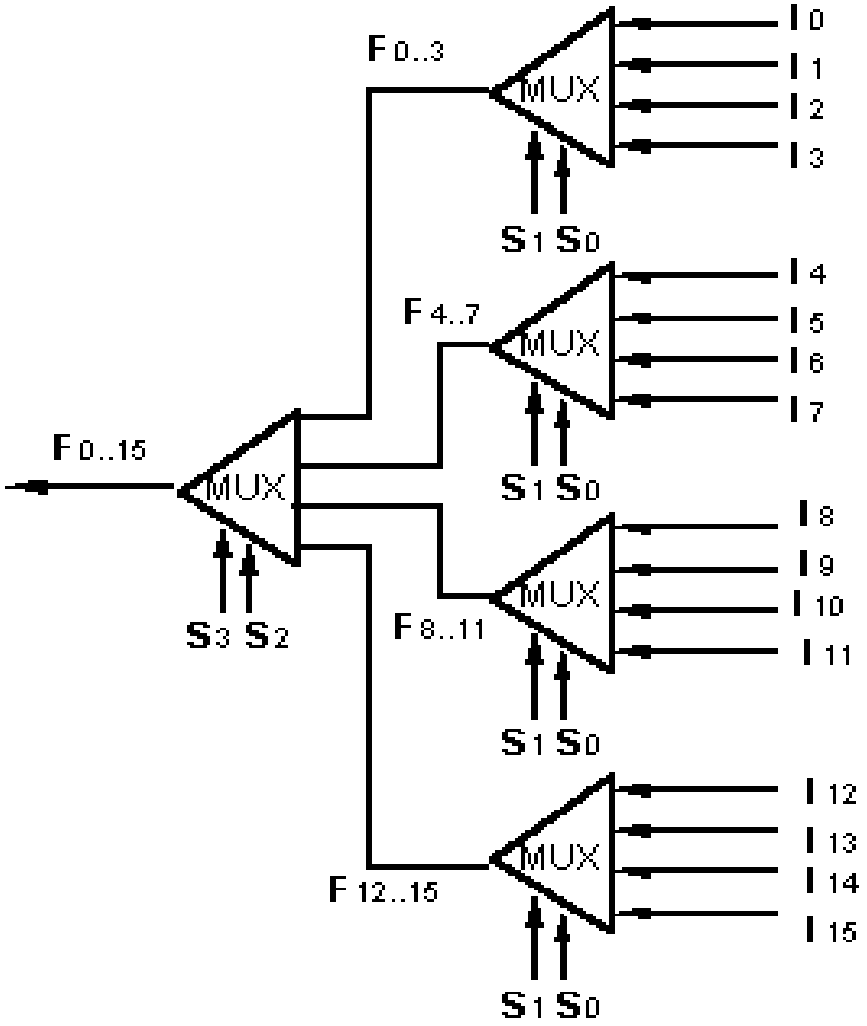
- Another implementation of an 8-to-1 multiplexer using smaller multiplexers:



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Larger Multiplexers

- A 16:1 multiplexer can be constructed from five 4:1 multiplexers:



Multiplexers: Implementing Functions

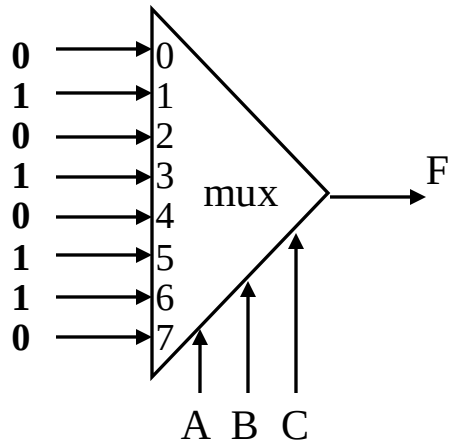
- A Boolean function can be implemented using multiplexers.
- A 2^n-to-1 multiplexer can implement a Boolean function of *n* input variables, as follows:
  - (i) Express in sum-of-minterms form.

    Example: F(A,B,C) = A'B'C + A'BC + AB'C + ABC'

    = Σm(1,3,5,6)
  - (ii) Connect *n* variables to the *n* selection lines.
  - (iii) Put a '1' on a data line if it is a minterm of the function, '0' otherwise.

Multiplexers: Implementing Functions

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



This method works because:

Output = $m_0.I_0 + m_1.I_1 + m_2.I_2 + m_3.I_3 + m_4.I_4 + m_5.I_5 + m_6.I_6 + m_7.I_7$

Supplying '1' to I_1, I_3, I_5, I_6 , and '0' to the rest:

Output = $m_1 + m_3 + m_5 + m_6$

Using Smaller Multiplexers

- Procedure

  1) Express boolean function in "sum-of-minterms" form.

     e.g. $F(A,B,C)= \Sigma m(0,1,3,6)$

  2) Reserve one variable (in our example, we take the least significant one) for input lines of multiplexer, and use the rest for selection lines.

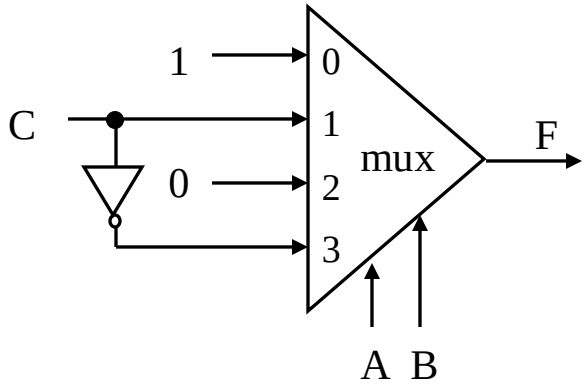
     e.g. C is for input lines, A and B for selection lines.

  3) Draw the truth table for function, do grouping of selection line values, and then determine multiplexer inputs by comparing input line (C) and function (F) for corresponding selection line values.

Using Smaller Multiplexers

A	B	C	F	Mux Input
0	0	0	1	1
0	0	1	1	
0	1	0	0	C
0	1	1	1	
1	0	0	0	0
1	0	1	0	
1	1	0	1	C’
1	1	1	0	

$F(A,B,C)= \Sigma m(0,1,3,6)$

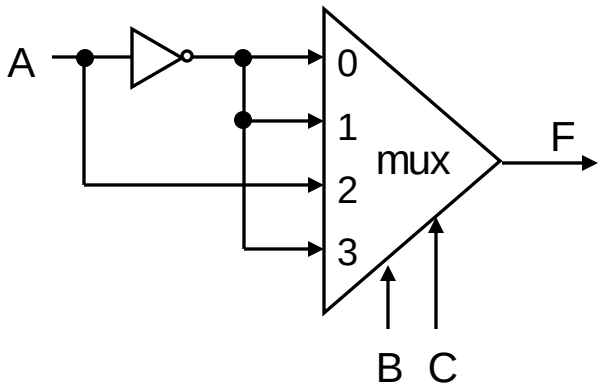


Using Smaller Multiplexers

- Alternative: What if we use A for input lines, and B, C for selector lines?

$F(A,B,C) = \Sigma m(0,1,3,6)$

A	B	C	F	Mux Input
0	0	0	1	1
0	0	1	1	
0	1	0	0	C
0	1	1	1	
1	0	0	0	0
1	0	1	0	
1	1	0	1	C’
1	1	1	0	



A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

A’ (when BC = 00)

A’ (when BC = 01)

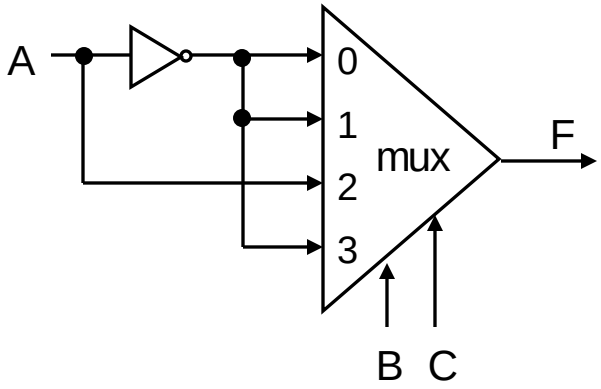
A (when BC = 10)

A’ (when BC = 11)

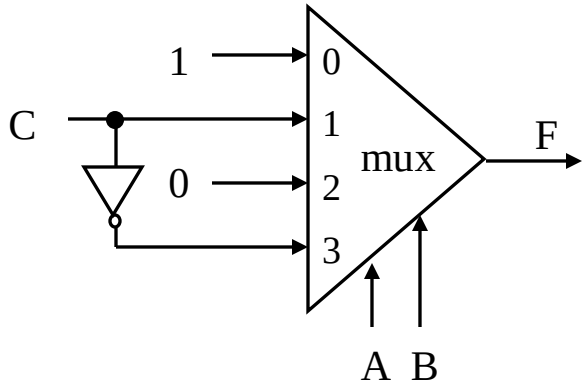
Using Smaller Multiplexers

$F(A,B,C)= \Sigma m(0,1,3,6)$

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



	I_0	I_1	I_2	I_3
(C')0	0	2	4	6
(C)1	1	3	5	7
	1	C	0	C'



Implement the Boolean function by using 8x1 MUX –
$F(A,B,C,D)=\sum m(0,1,3,4,8,9,14,15)$

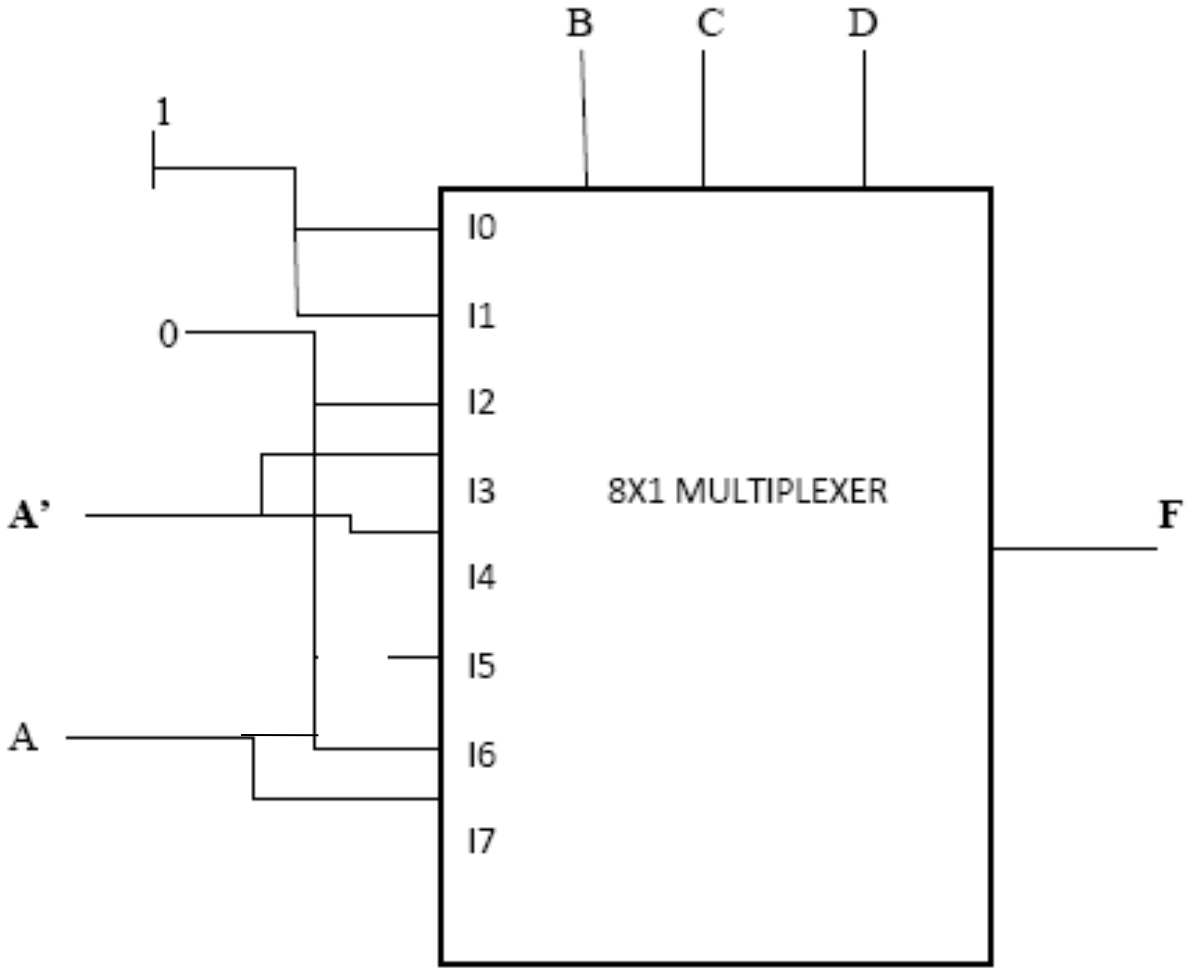
Total number of variables , n= 4 (A,B,C,D)
Select line variable (n-1) = 3 (B,C,D) Input variable = 1 (A)

Implementation table –

	I0	I1	I2	I3	I4	I5	I6	I7
(A')0	0	1	2	3	4	5	6	7
(A)1	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	A	A

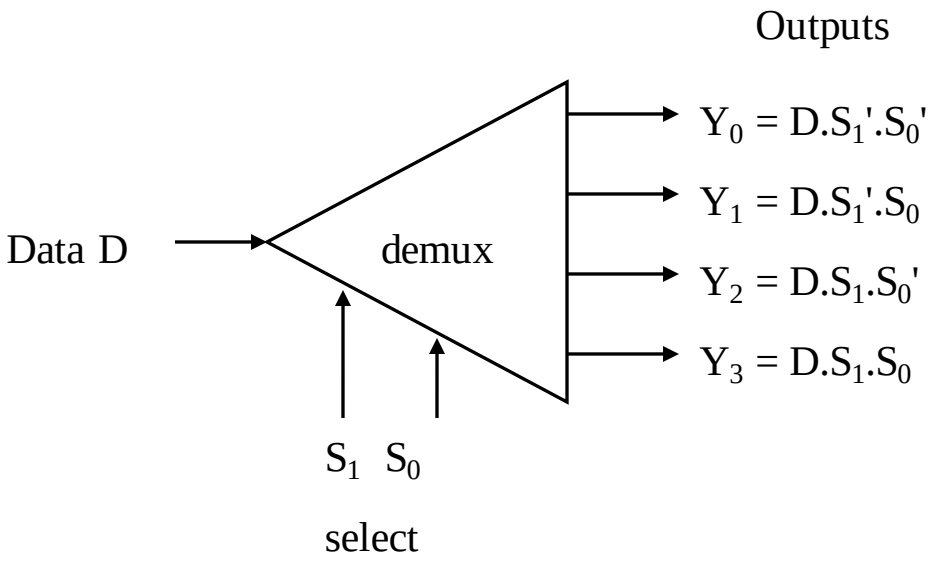
$F(A,B,C,D)=\sum m(0,1,3,4,8,9,14,15)$

Diagram corresponding to implementation table.



Demultiplexer

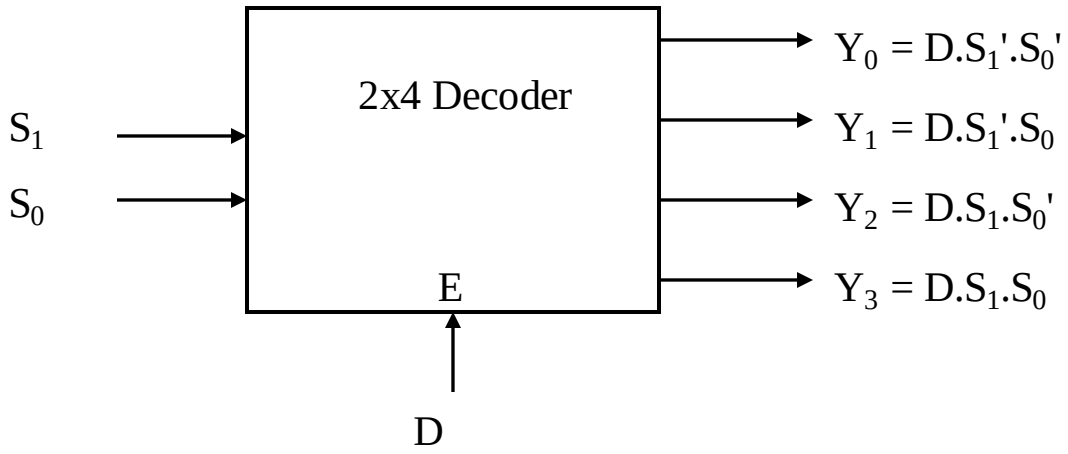
- Given an input line and a set of selection lines, the demultiplexer will direct data from input to a selected output line.
- An example of a 1-to-4 demultiplexer:



S_1	S_o	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

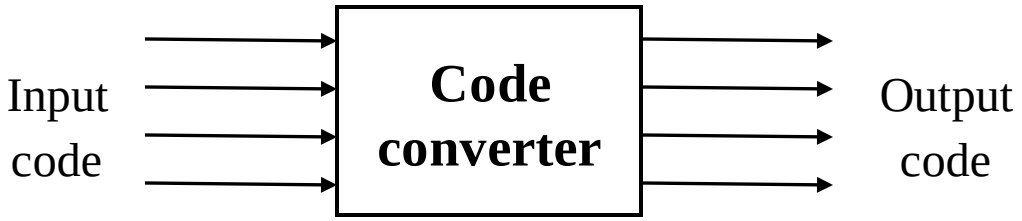
Demultiplexer

- The demultiplexer is actually identical to a decoder with enable, as illustrated below:



Code Converters

- Code converters – take an input code, translate to its equivalent output code.

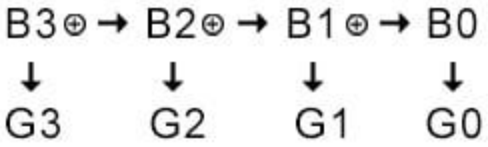


Example: BCD to Excess-3 Code Converter.

Input: BCD digit

Output: Excess-3 digit

4 Bit Binary to Gray Code Converter



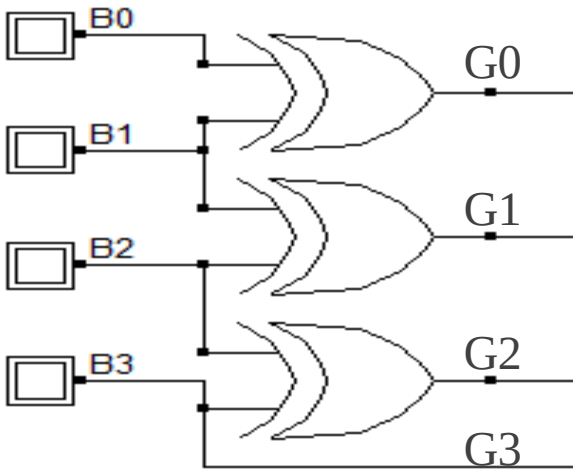
Based on this

G3 = B3

G2 = B2 ⊕ B3

G1 = B1 ⊕ B2

G0 = B0 ⊕ B1

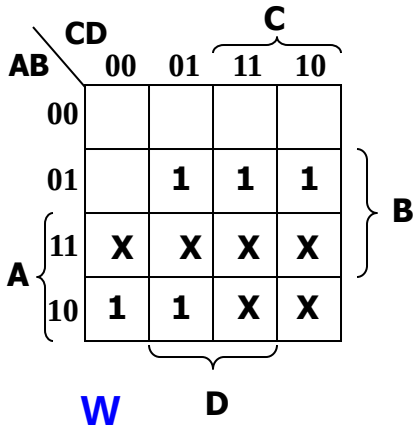


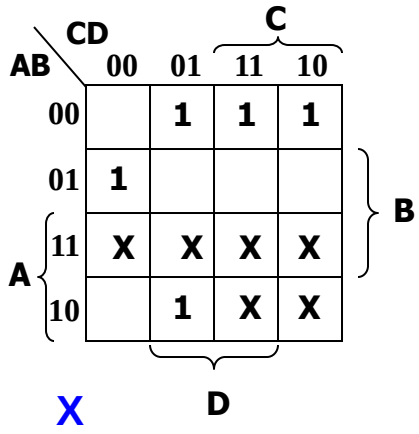
BCD-to-Excess-3 Code Converter

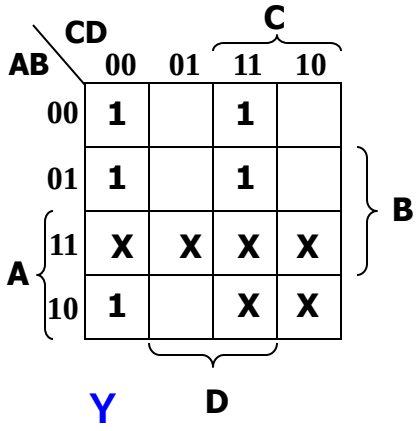
- Truth table:

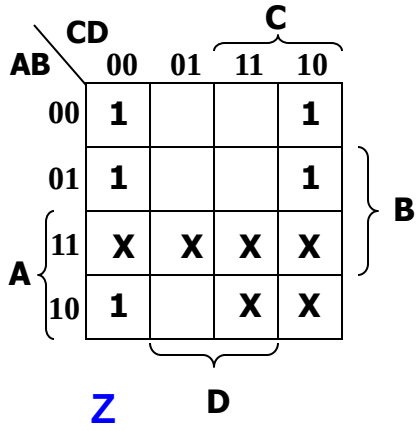
	BCD				Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

K-maps:

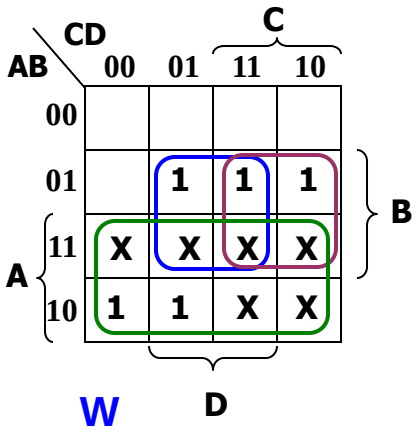


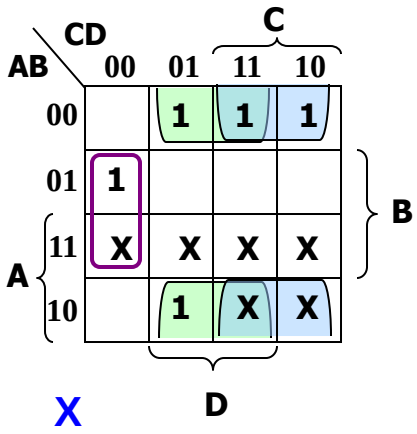


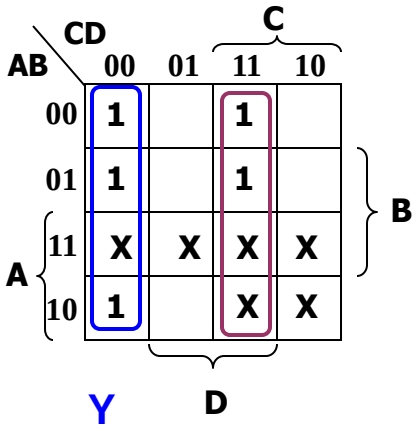


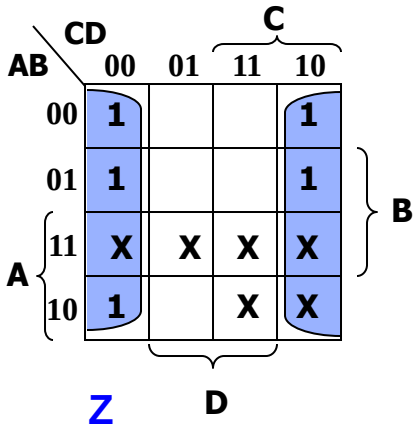


BCD-to-Excess-3 Code Converter









W = A + B.C + B.D

X = B'.C + B‘.D + B.C'.D'

Y = C.D + C'.D'

Z = D'

Comparator

A comparator compares two n-bit values to determine which is greater, or if they are equal.

1 bit comparator: $A = A_0$, $B = B_0$

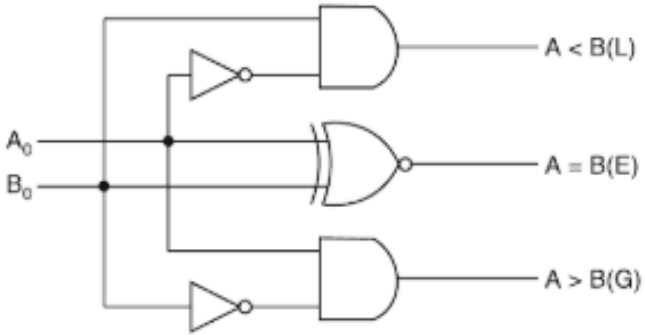
$A0 < B0: L = \overline{A0}\ B0$

$A0 = B0: E = \overline{A0}\ \overline{B0} + A0\ B0$

$A0 > B0: G = A0\ \overline{B0}$

It is to be noted that E can be realized as $\overline{(L + G)}$.

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



(b) Logic diagram

2-bit Magnitude Comparator

The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be $A = A_1A_0$ and $B = B_1B_0$.

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is

$$A > B : G = A_1\overline{B}_1 + (A_1 \odot B_1)A_0\overline{B}_0$$

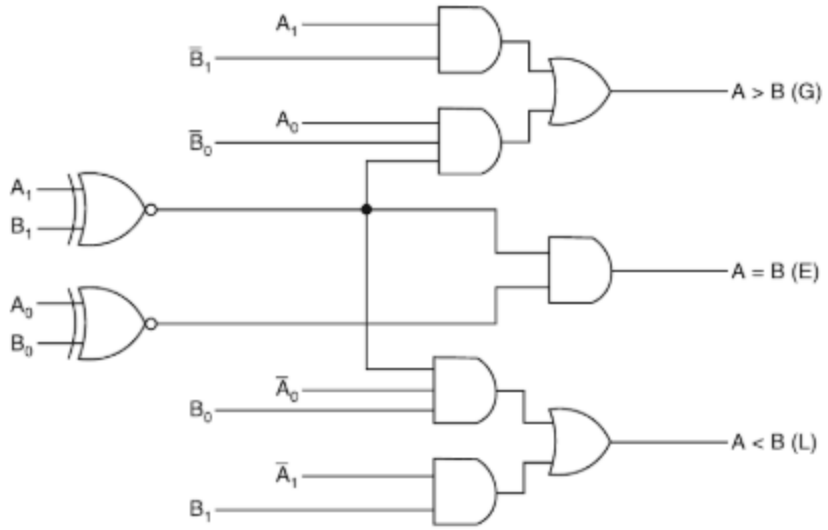
1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If A_1 and B_1 coincide and $A_0 = 0$ and $B_0 = 1$, then $A < B$. So the expression for $A < B$ is

$$A < B : L = \overline{A}_1B_1 + (A_1 \odot B_1)\overline{A}_0B_0$$

If A_1 and B_1 coincide and if A_0 and B_0 coincide then $A = B$. So the expression for $A = B$ is

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$

The logic diagram for a 2-bit comparator is as shown in Figure



Extending to Multibit Numbers

- Compare the most significant bits.
  - If they are not equal, no need to compare the other bits.
  - If they are equal, we must check the next bit.
- Continue until one number if found to be greater than the other, or all bits are checked and the numbers are found equal.

Arithmetic Circuits: Comparator

- Magnitude comparator: compares 2 values A and B, to see if A>B, A=B or A<B.
- Classical method requires 2^{2n} rows in truth table!
- How do we compare two 4-bit values A ($a_3a_2a_1a_0$) and B ($b_3b_2b_1b_0$)?

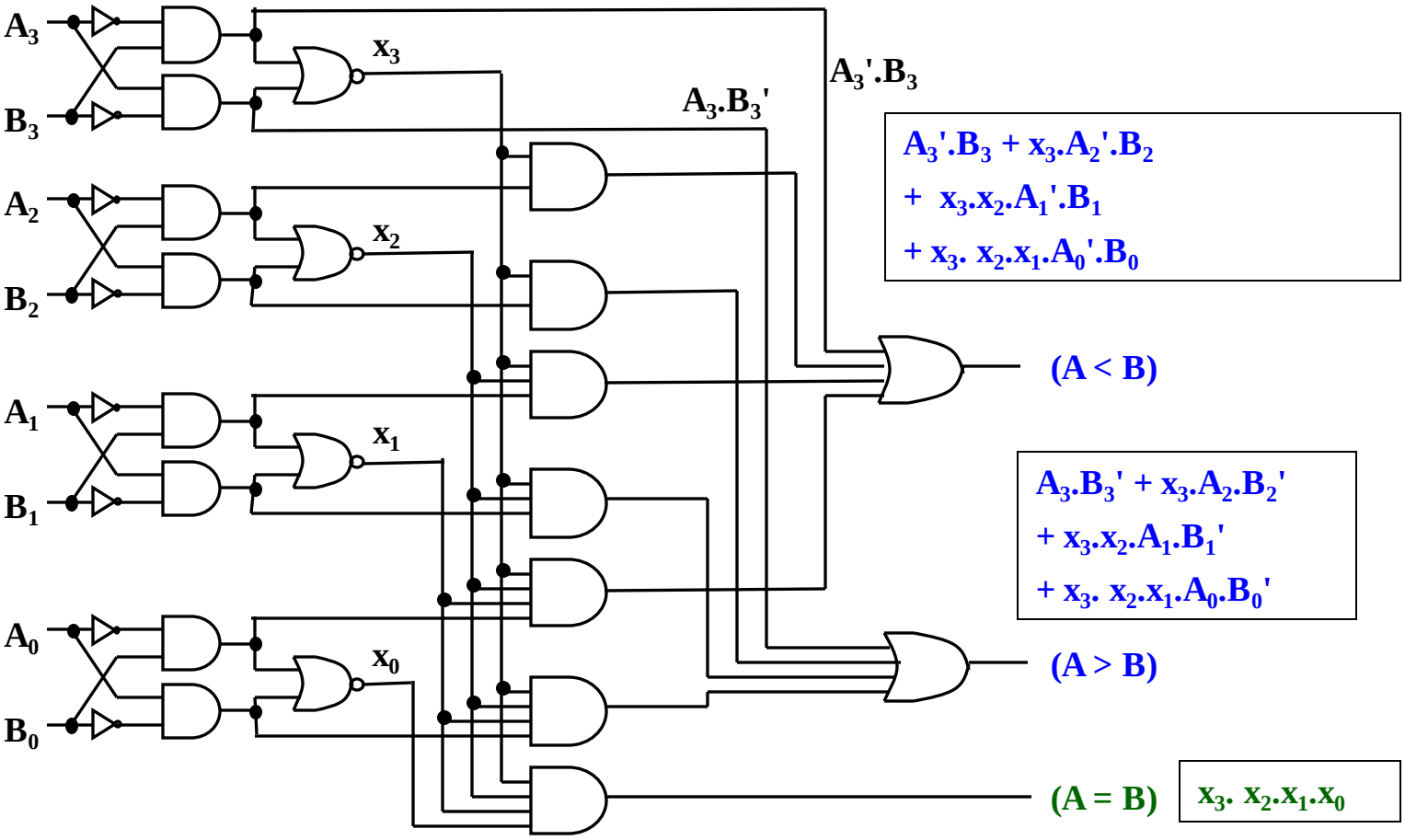
  If ($a_3 > b_3$) then A > B

  If ($a_3 < b_3$) then A < B

  If ($a_3 = b_3$) then if ($a_2 > b_2$) ….

Arithmetic Circuits: Comparator

Let $A = A_3A_2A_1A_0$, $B = B_3B_2B_1B_0$; $x_i = A_i.B_i + A_i'.B_i'$



References

1. A. Anand Kumar, “Fundamentals of Digital Circuits”, PHI, Fourth Edition.

2. S. Salivahanan and S. Arivazhagan, “Digital circuits and design”, Vikas Publishing House PVT Limited, Fifth edition.