SWITCHING THEORY AND LOGIC DESIGN

UNIT-3

BY
S.SAGAR KRISHNA,
ASSISTANT PROFESSOR,
GAYATRI VIDYA PARISHAD COLLEG OF ENGINEERIN(A)

BCD TO EXCESS-3 CONVERTER

Procedure:

- Write the functional table.
- Using K-MAP get the Boolean expressions for each output.
- Draw the logic diagram.

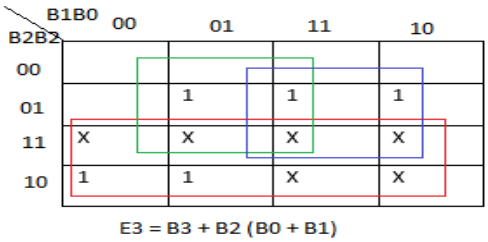
BCD TO EXCESS-3 CONVERTER

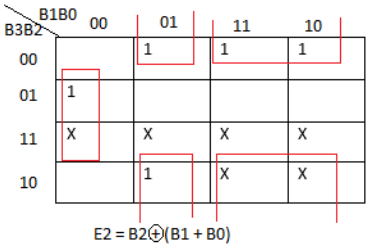
Functional Table:

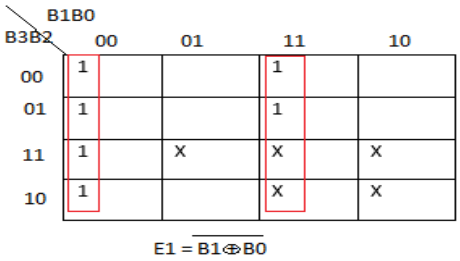
BCD INPUT				EXCESS-3 OUPUT			
B3	B2	B1	B0	E3	E2	E1	E0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

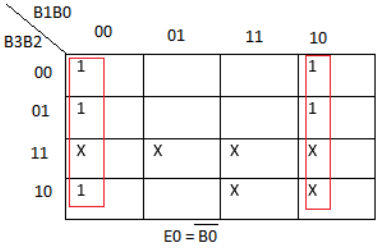
BCD TO EXCESS-3 CONVERTER

Boolean expressions for each output:



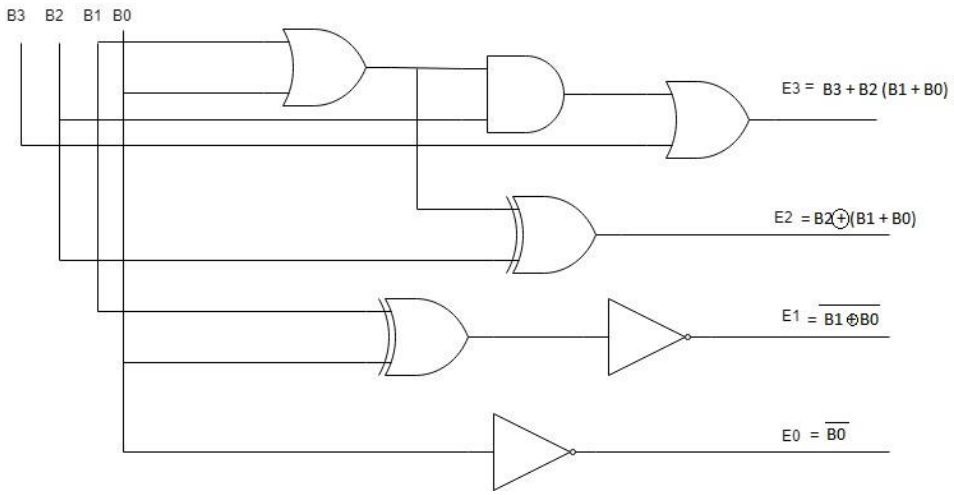






BCD TO EXCESS-3 CONVERTER

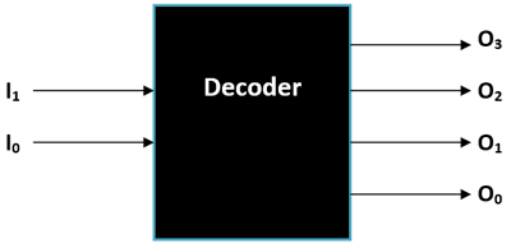
logic diagram :



Decoders

- A decoder has
  - N inputs
  - 2^N outputs
- A decoder selects one of 2^N outputs by decoding the binary value on the N inputs.
- The decoder generates all of the minterms of the N input variables.
  - Exactly one output will be active for each combination of the inputs.

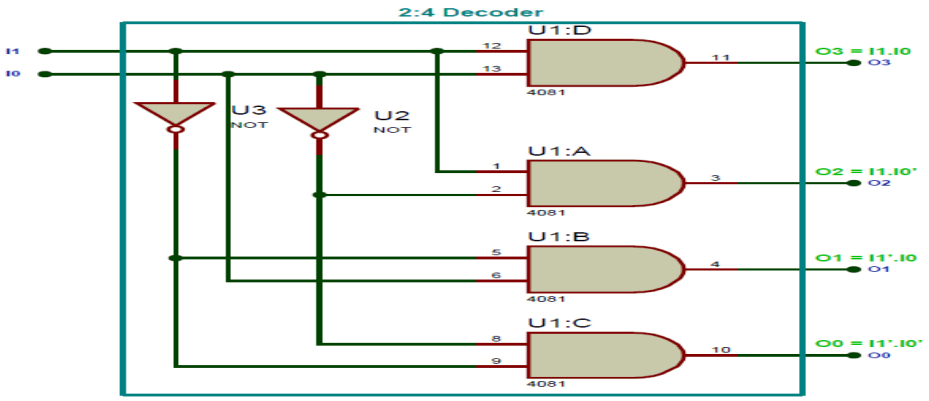
2x4 Decoder:



Functional Table of 2x4 Decoder

Input		Output			
I1	I0	O3	O2	O1	O0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Logic Diagram

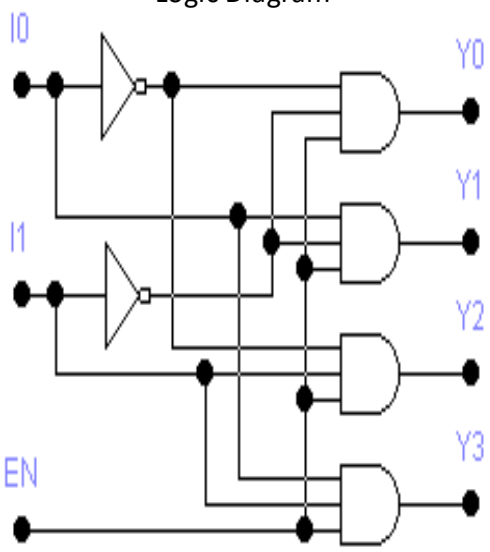


2x4 decoder with enable:

Inputs			Outputs			
EN	I1	I0	Y3	Y2	Y1	Y0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

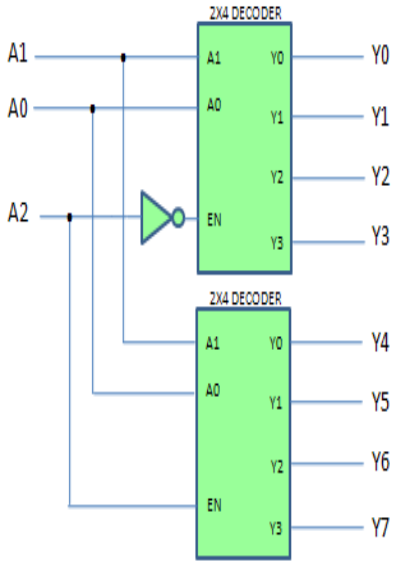
Functional Table

Logic Diagram



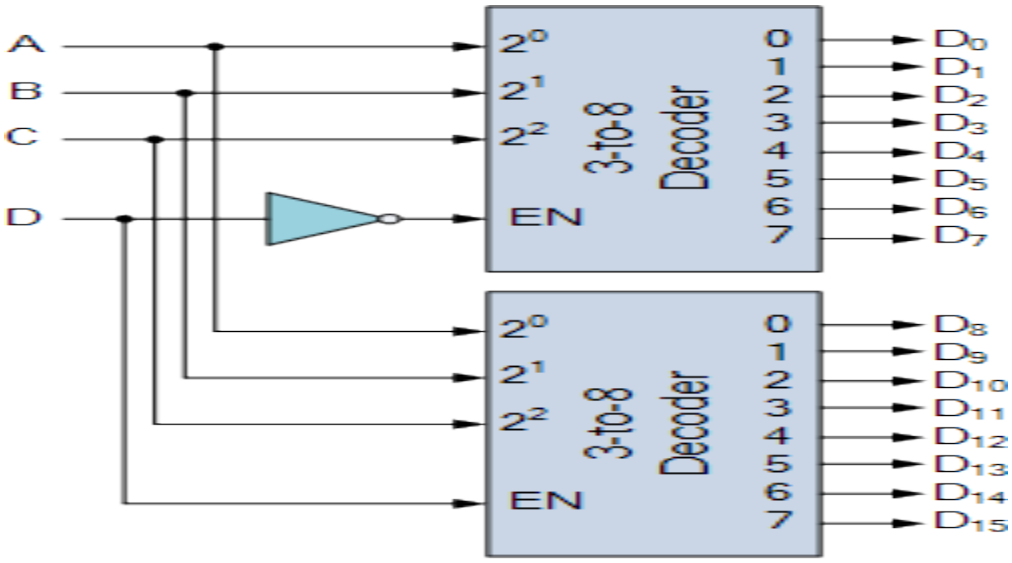
3x8 decoder using 2x4 decoder

- The inputs A0 and A1 is connected as parallel inputs for both the decoders and then the Enable pin of the Second Decoder is made to act as A2 (third input).
- The Inverted signal of A2 is given to the Enable pin of first decoder to get the outputs Y0 to Y3. Here the outputs Y0 to Y3 is referred as Lower four minterms and the outputs Y4 to Y7 is referred as higher four minterms.
- The lower order minterms are obtained from the first decoder and the higher order minterms are obtained from the second decoder.



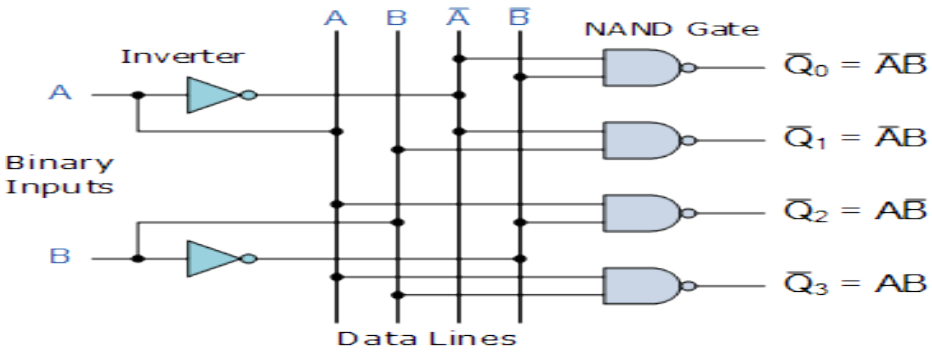
Logic Diagram

4x16 decoder using 3x8 decoder



4-to-16 Line Decoder Implemented with two 3-to-8 Decoders

2x4 decoder with active low output



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Encoders

- An encoder has
  - 2^N inputs
  - N outputs
- An encoder outputs the binary value of the selected (or active) input.
- An encoder performs the inverse operation of a decoder.
- The encoder can be implemented with OR gates whose inputs are determined directly from the truth table. Output *z* is equal to 1 when the input octal digit is 1, 3, 5,or 7. Output *y* is 1 for octal digits 2, 3, 6, or 7, and output *x* is 1 for digits 4, 5, 6, or 7.

Octal to Binary Encoder

- Output *z* is equal to 1 when the input octal digit is 1, 3, 5,or 7. Output *y* is 1 for octal digits 2, 3, 6, or 7, and output *x* is 1 for digits 4, 5, 6, or 7.

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- Issues
  - What if more than one input is active?
  - What if no inputs are active?

Priority Encoders

- If more than one input is active, the higher-order input has priority over the lower-order input.
  - The higher value is encoded on the output
- A valid indicator, v, is included to indicate whether or not the output is valid.
  - Output is invalid when no inputs are active
    - v = 0
  - Output is valid when at least one input is active
    - v = 1

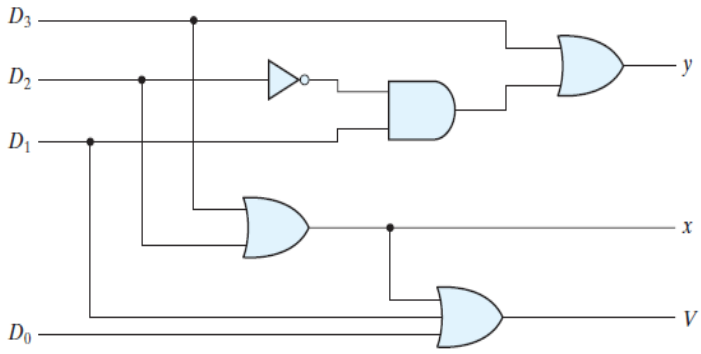
Priority Encoders

Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Boolean Expressions

$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$

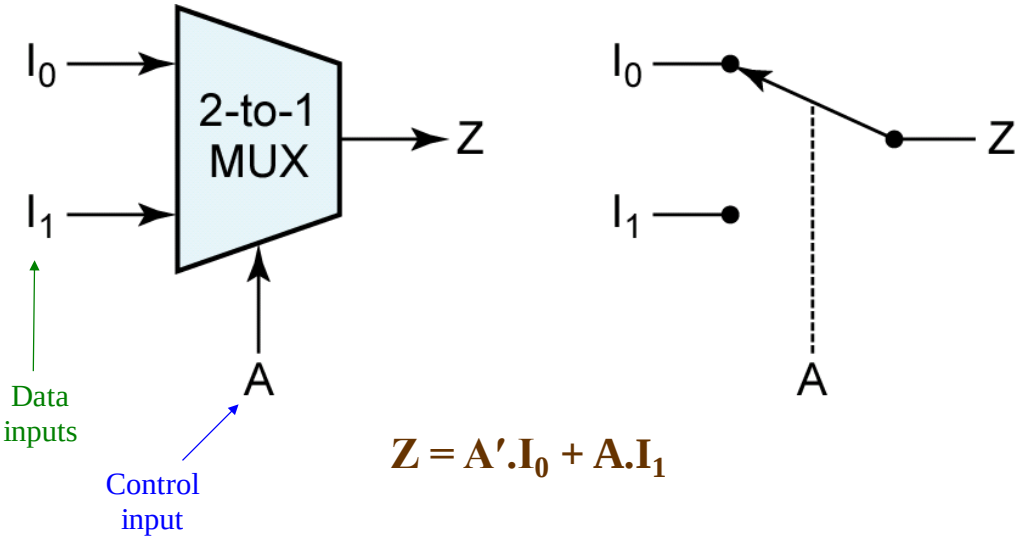


Logic Diagram

Multiplexers

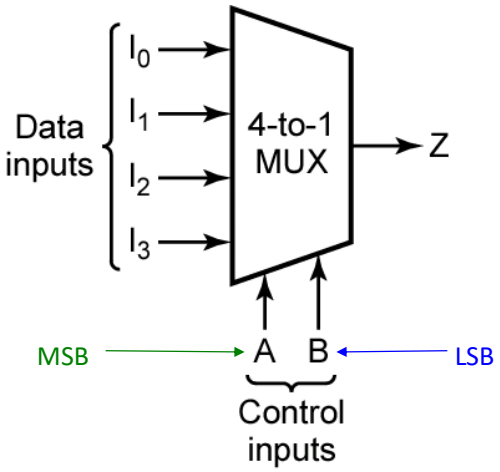
- A multiplexer has
  - N control inputs(Select lines)
  - 2^N data inputs
  - 1 output
- A multiplexer routes (or connects) the selected data input to the output.
  - The value of the control inputs determines the data input that is selected.

Multiplexers



$$Z = A'.I_0 + A.I_1$$

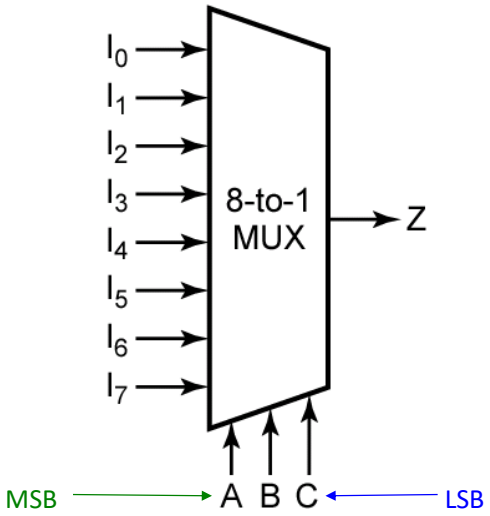
Multiplexers



A	B	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Z = A'.B'.I_0 + A'.B.I_1 + A.B'.I_2 + A.B.I_3$$

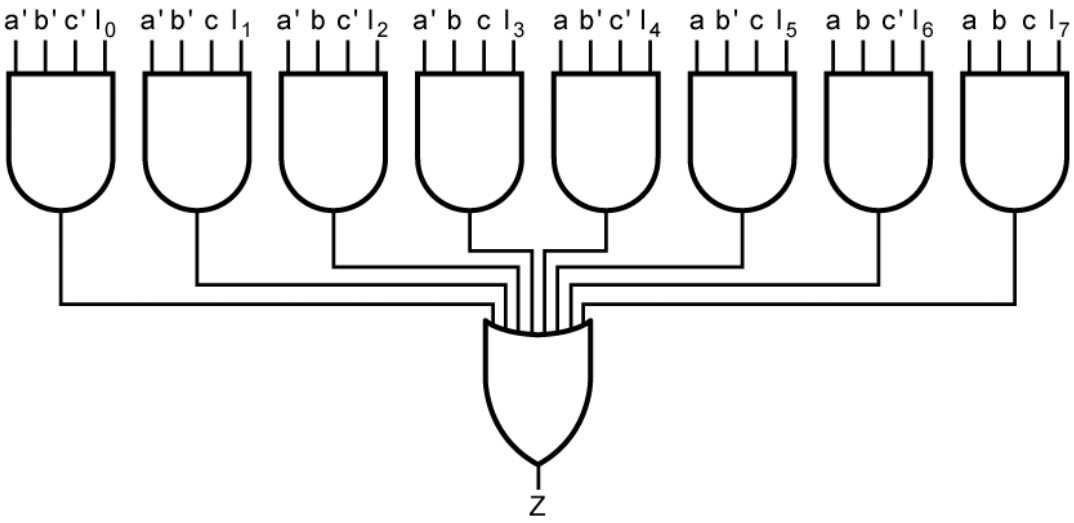
Multiplexers



A	B	C	F
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

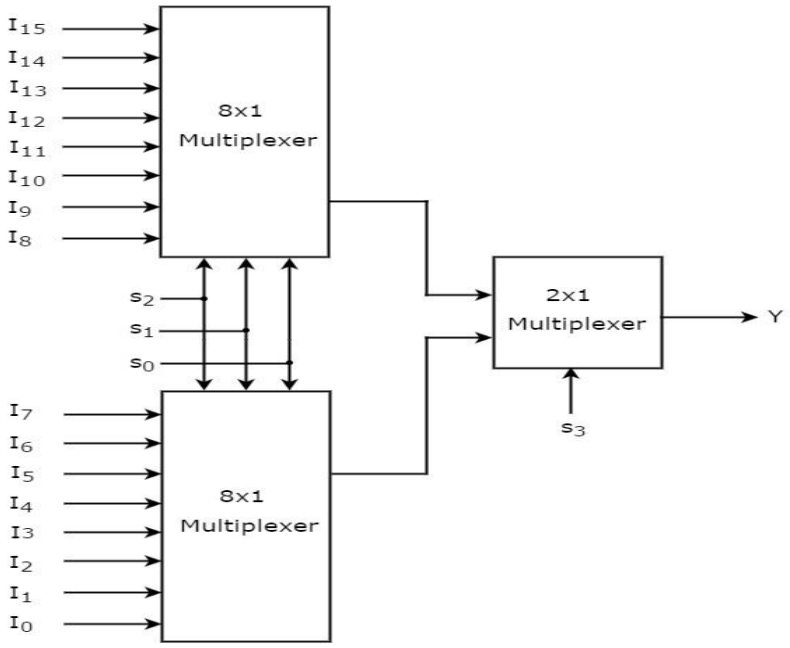
$$Z = A'.B'.C'.I_0 + A'.B'.C.I_1 + A'.B.C'.I_2 + A'.B.C.I_3 + A.B'.C'.I_0 + A.B'.C.I_1 + A'.B.C'.I_2 + A.B.C.I_3$$

Multiplexers



Logic Diagram of 8x1 Multiplexer

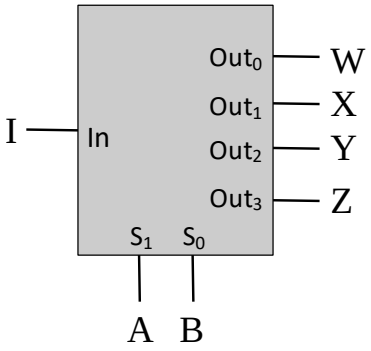
16x1 MULTIPLEXER USING 8X1 MUX



Demultiplexers

- A demultiplexer has
  - N control inputs
  - 1 data input
  - 2^N outputs
- A demultiplexer routes (or connects) the data input to the selected output.
  - The value of the control inputs determines the output that is selected.
- A demultiplexer performs the opposite function of a multiplexer.

Demultiplexers



W = A'.B'.I

X = A.B'.I

Y = A'.B.I

Z = A.B.I

A	B	W	X	Y	Z
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

Designing logic circuits using multiplexers

using an *n*-input Multiplexer

- Use an *n*-input multiplexer to realize a logic circuit for a function with *n* minterms.
  - $m = 2^n$, where m = # of variables in the function
- Each minterm of the function can be mapped to an input of the multiplexer.
- For each row in the truth table, for the function, where the output is 1, set the corresponding input of the multiplexer to 1.
  - That is, for each minterm in the minterm expansion of the function, set the corresponding input of the multiplexer to 1.
- Set the remaining inputs of the multiplexer to 0.

Designing logic circuits using multiplexers

Using an *(n / 2)*-input Multiplexer

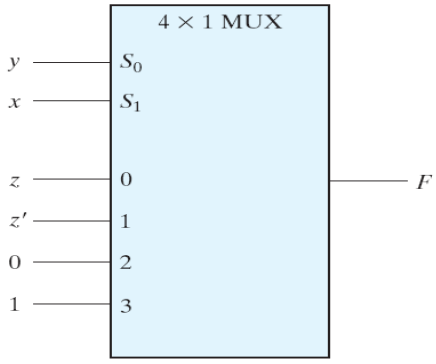
- Use an *(n / 2)*-input multiplexer to realize a logic circuit for a function with *n* minterms.
  - $m = 2^n$, where m = # of variables in the function
- Group the rows of the truth table, for the function, into *(n / 2)* pairs of rows.
  - Each pair of rows represents a product term of *(m – 1)* variables.
  - Each pair of rows can be mapped to a multiplexer input.
- Determine the logical function of each pair of rows in terms of the m^{th} variable.
  - If the m^{th} variable, for example, is x, then the possible values are x, x', 0, and 1.

Using an *(n / 2)*-input Mux

Example: F(x,y,z) = Σm(1, 2, 6, 7)

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

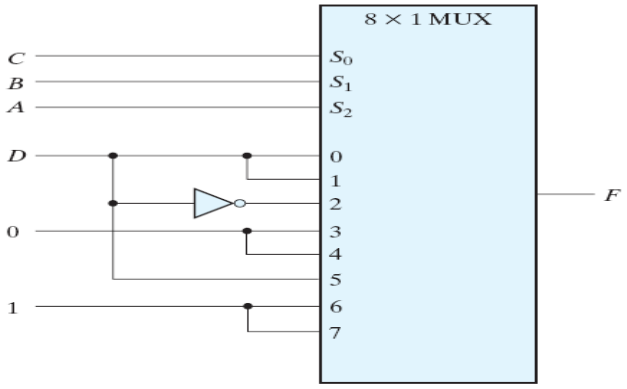


(b) Multiplexer implementation

Using an *(n / 2)*-input Mux

Example: F(A,B,C,D) = Σm(1,3,4,11,12,13,14,15)

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



QUESTIONS:

- Design a 16x1 Multiplexer using 2x1 Multiplexer.
- Using an 8-to-1 multiplexer, design a logic circuit to realize the following Boolean function F(A,B,C) = Σm(2, 3, 5, 6, 7)
- Using an 4-to-1 multiplexer, design a logic circuit to realize the following Boolean function F(A,B,C) = Σm(0,1, 3, 5, 6, 7)
- Design a 4 bit Prime number detector.
- Design a BCD to Gray Code Converter.

THANK YOU