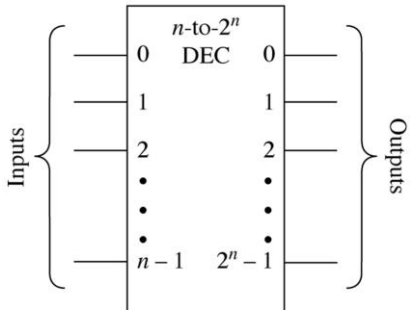
Dept. of Computer Science and Engineering
University of Rajshahi
www.ru.ac.bd

Dr. Shamim Ahmad

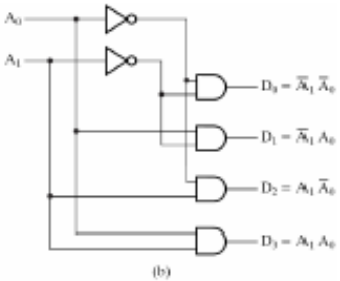
An *n*-to-2*n*-line decoder symbol



2-to-4 Decoder

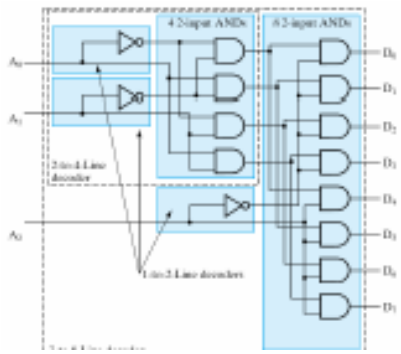
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

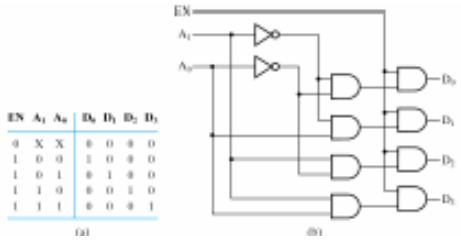


(b)

Decoder Expansion



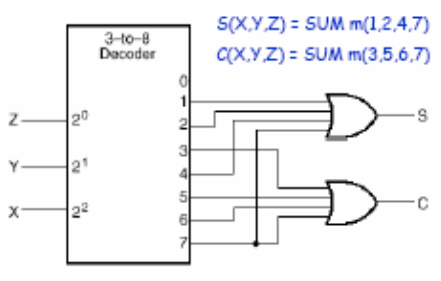
Decoder with enable: 2-to-4



EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a) (b)

Implementing a Binary Adder Using a Decoder



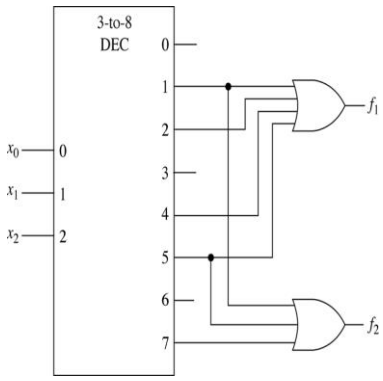
Realization of the Boolean expressions

***f1*(*x*2,*x*1,*x*0) = Σ*m*(1,2,4,5)**

and

***f2*(*x*2,*x*1,*x*0) = Σ*m*(1,5,7)**

with a 3-to-8-line decoder and two or-gates.



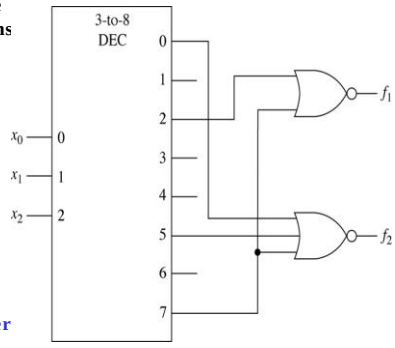
Realization of the Boolean expressions

***f1*(*x*2,*x*1,*x*0) = Σ*m*(0,1,3,4,5,6) = !Σ*m*(2,7)**

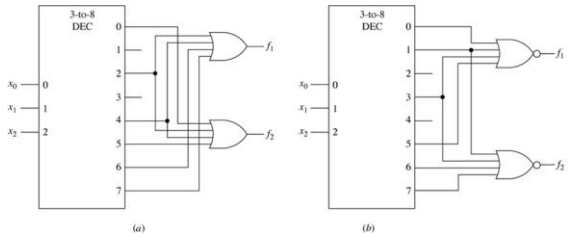
and

***f2*(*x*2,*x*1,*x*0) = Σ*m*(1,2,3,4,6) = !Σ*m*(0,5,7)**

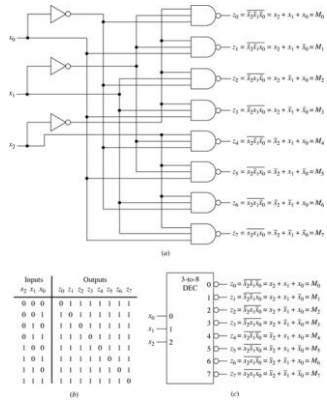
with a 3-to-8-line decoder and two nor-gates.



$f1(x2,x1,x0) = \Pi M(0,1,3,5)$
$f2(x2,x1,x0) = \Pi M(1,3,6,7)$
(*a*) Using output or-gates.
(*b*) Using output nor-gates.



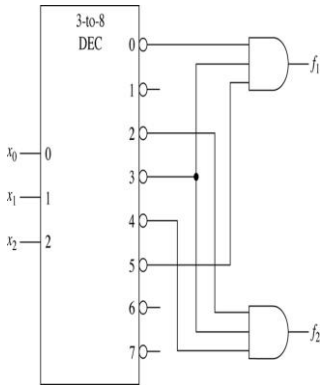
A 3-to-8-line decoder using nand-gates.
(*a*) Logic diagram
(*b*) Truth table
(*c*) Symbol



Realization of the pair of maxterm canonical expressions

$f1(x2,x1,x0) = \Pi M(0,3,5)$
and
$f2(x2,x1,x0) = \Pi M(2,3,4)$

with a 3-to-8-line decoder and two and-gates



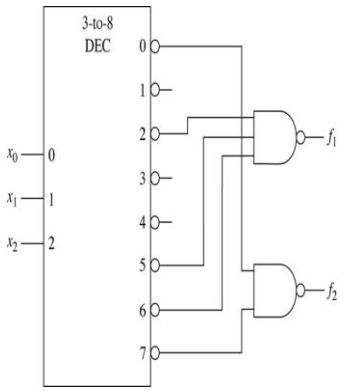
Realization of the Boolean expressions

$f1(x2,x1,x0) = \Pi M(0,1,3,4,7) = \Pi M(2,5,6)$

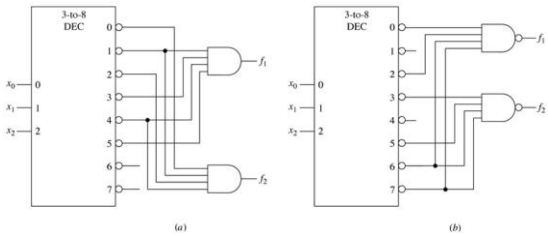
and

$f2(x2,x1,x0) = \Pi M(1,2,3,4,5,6) = \Pi M(0,7)$

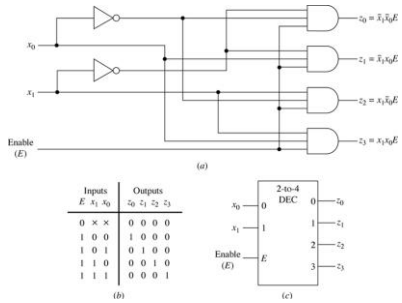
with a 3-to-8-line decoder and two nand-gates.



A decoder realization of $f1(x2,x1,x0) = \Sigma m(0,2,6,7)$ and $f2(x2,x1,x0) = \Sigma m(3,5,6,7)$
(*a*) Using output and-gates.
(*b*) Using output nand-gates.

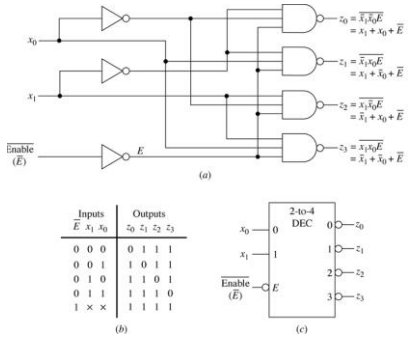


And-gate 2-to-4-line decoder with an enable input.
(*a*) Logic diagram
(*b*) Compressed truth table
(*c*) Symbol.



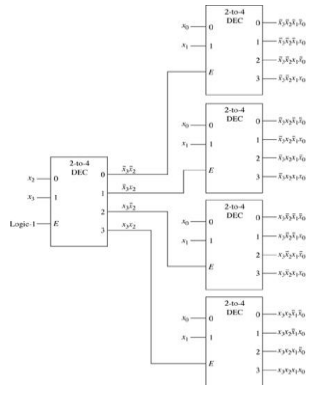
Inputs E x_1 x_0	Outputs z_0 z_1 z_2 z_3
0 × ×	0 0 0 0
1 0 0	1 0 0 0
1 0 1	0 1 0 0
1 1 0	0 0 1 0
1 1 1	0 0 0 1

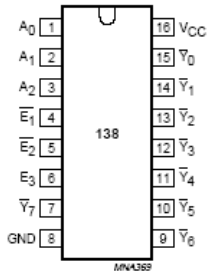
Nand-gate 2-to-4-line decoder with an enable input
(*a*) Logic diagram
(*b*) Compressed truth table. (*c*) Symbol.

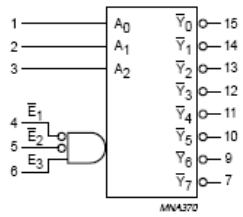


Inputs $\bar{E}$ x_1 x_0	Outputs z_0 z_1 z_2 z_3
0 0 0	0 1 1 1
0 0 1	1 0 1 1
0 1 0	1 1 0 1
0 1 1	1 1 1 0
1 × ×	1 1 1 1

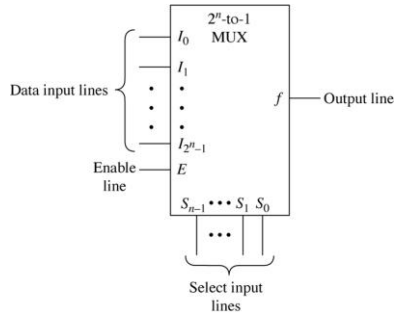
A 4-to-16-line decoder constructed from 2-to-4-line decoder



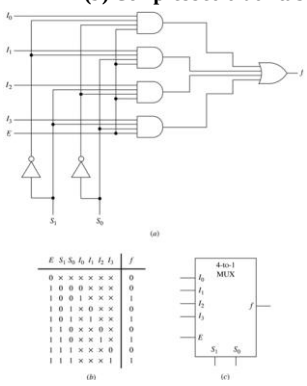




A 2*n*-to-1-line Multiplexer symbol

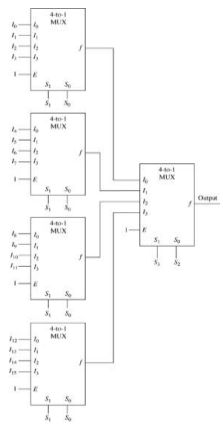


A 4-to-1-line multiplexer. (*a*) Logic diagram (*b*) Compressed truth table. (*c*) Symbol



E	S_1	S_2	f
0	x	x	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

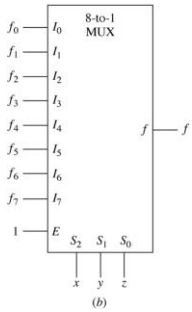
A Multiplexer tree to form a 16-to-1-line Multiplexer



Realization of a three-variable function using a 8-to-1-line Multiplexer.
(*a*) Three-variable truth table
(*b*) General realization.

x	y	z	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

(*a*)

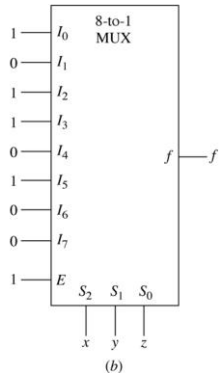


(*b*)

Realization of $f(x,y,z) = \Sigma m(0,2,3,5)$
(*a*) Truth table
(*b*) 8-to-1-line multiplexer realization.

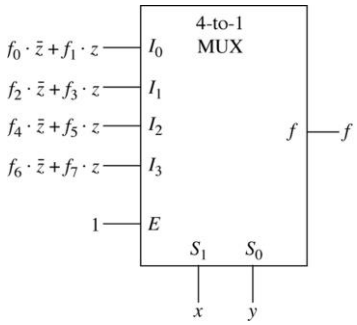
x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(*a*)

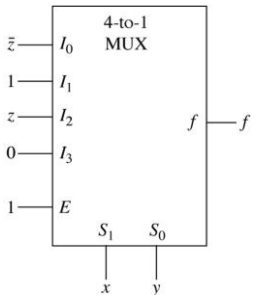


(*b*)

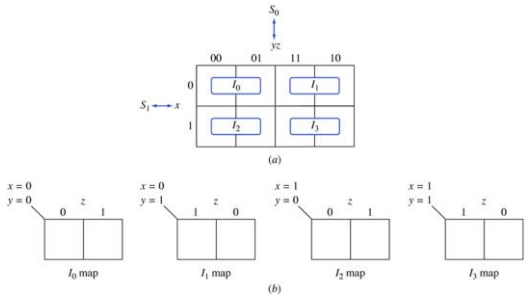
A general realization of a 3-variable Boolean function using a 4-to-1-line multiplexer.



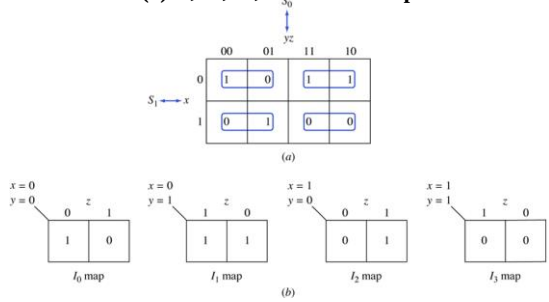
Realization of $f(x,y,z) = \Sigma m(0,2,3,5)$ using a 4-to-1-line multiplexer.



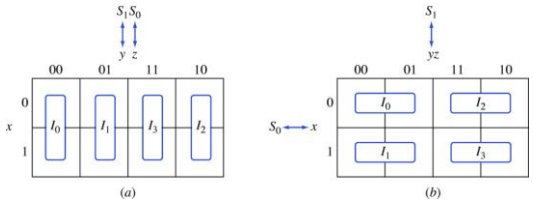
Obtaining multiplexer realizations using Karnaugh maps.
(*a*) Cell groupings corresponding to the data line functions.
(*b*) Karnaugh maps for the *Ii* subfunctions



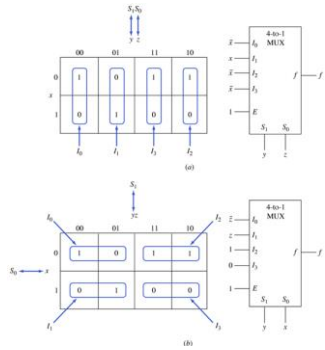
Realization of $f(x,y,z) = \Sigma m(0,2,3,5)$
(*a*) Karnaugh map.
(*b*) *I*0, *I*1, *I*2, and *I*3 submaps.



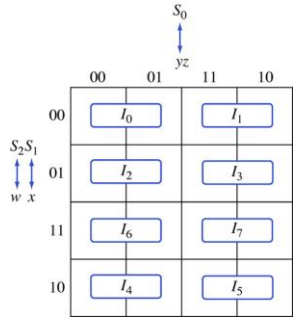
Using Karnaugh maps to obtain multiplexer realizations under various assignments to the select inputs.
(*a*) Applying input variables *y* and *z* to the *S*1 and *S*0 select lines.
(*b*) Applying input variables *x* and *y* to the *S*0 and *S*1 select lines.



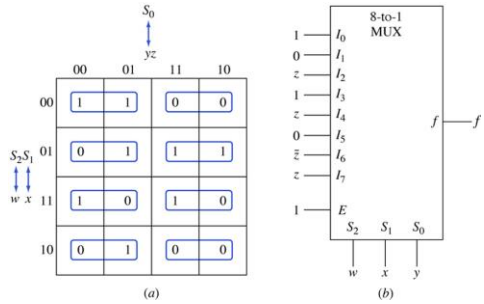
Alternative realizations of $f(x,y,z) = \Sigma m(0,2,3,5)$.
(*a*) Applying input variables *y* and *z* to the *S*1 and *S*0 select lines.
(*b*) Applying input variables *x* and *y* to the *S*0 and *S*1 select lines.



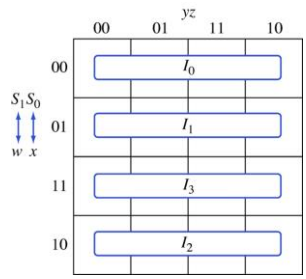
A select line assignment and corresponding data line functions for a multiplexer realization of a four-variable function.

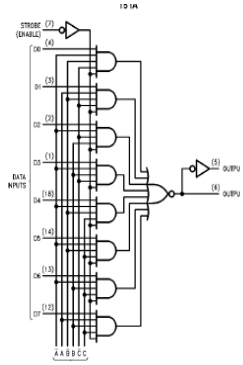


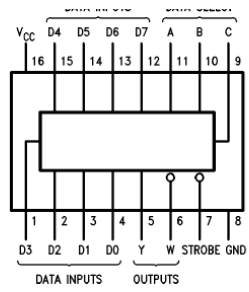
Realizations of $f(w,x,y,z) = \Sigma m(0,1,5,6,7,9,12,15)$ (*a*) Karnaugh map. (*b*) Multiplexer realization.



Using a four-variable Karnaugh map to obtain a Boolean function realization with a 4-to-1-line multiplexer.



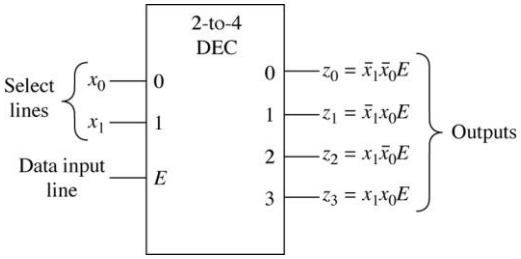




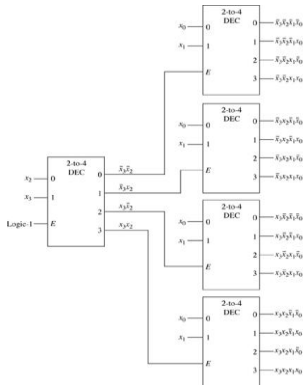
TL/F/6541

Order Number 54151ADMQB, 54151AFMQB, DM54151AJ, DM54151AW or DM74151AN
See NS Package Number J16A, N16E or W16A

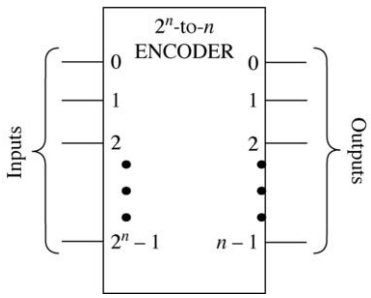
Demultiplexer.



A 4-to-16-line decoder constructed from 2-to-4-line decoder



Encoders

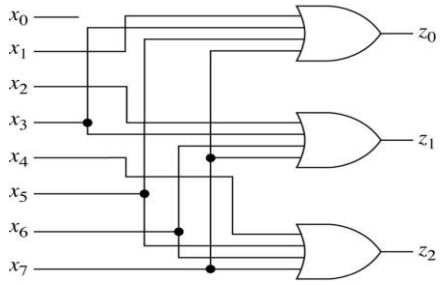


Encoder Example

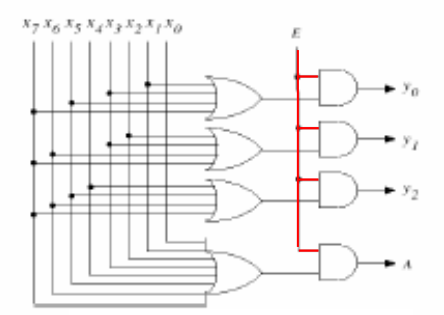
- Example: 8-to-3 binary encoder (octal-to-binary)

$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

An 8-to-3-line encoder.



Encoder Example (cont.)



4-to-2 Priority Encoder (cont.)

• The operation of the priority encoder is such that:

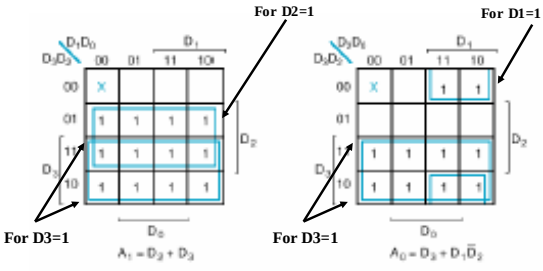
• If two or more **inputs are equal to 1 at the same time**, the input in the **highest-numbered** position will take precedence.

• A ***valid output indicator***, designated by **V,** is set to 1 only when **inputs one or more** are equal

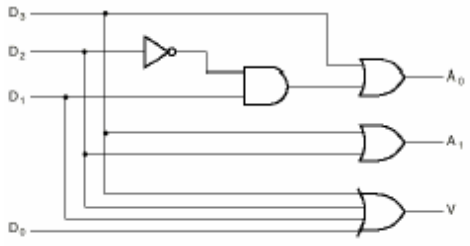
Example: 4-to-2 Priority Encoder Truth Table

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Example: 4-to-2 Priority Encoder K-Maps

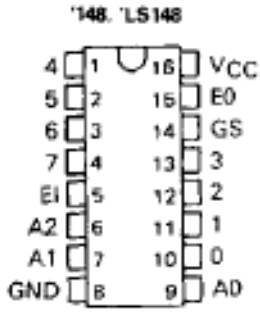


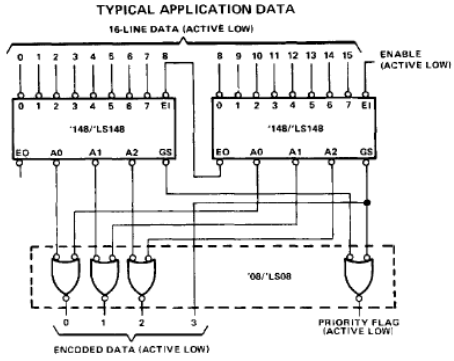
Example: 4-to-2 Priority Encoder Logic Diagram



Priority Encoders

x0	x1	x2	x3	x4	x5	x6	x7	z2	z1	z0	V
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1





A multiplexer/demultiplexer arrangement for information transmission