Multiplexers

MULTIPLEXERS (Data Selectors)

A multiplexers (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.

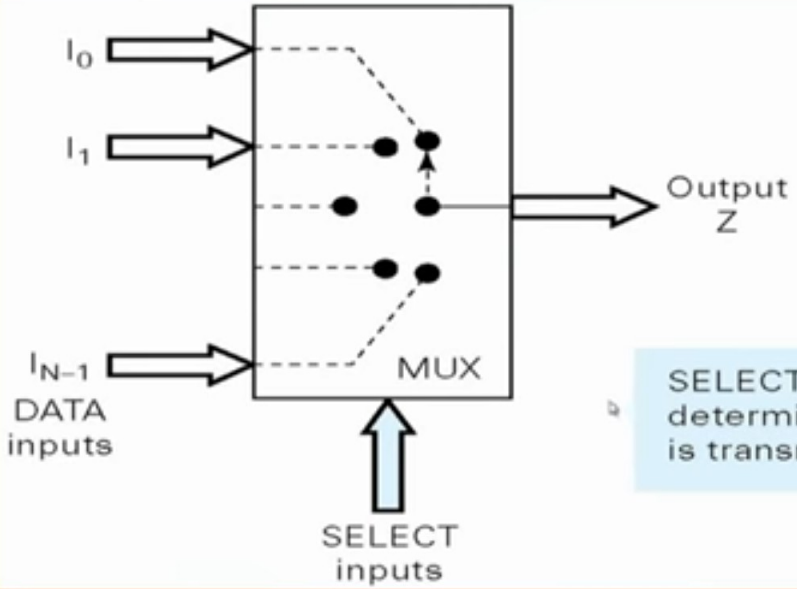
The basic multiplexers has several data input lines and a single output line. It also has data- select inputs, which permit digital data on any one of the inputs to be switched to the output line.

MUX-continued...

A modern stereo system may have a switch that selects music from one of four sources: a cassette tape, CD, a radio tuner , or an auxiliary input such as audio from a VCR or DVD. The switch selects one of the electronic signals from one of these four sources and sends it to the power amplifier and speakers.

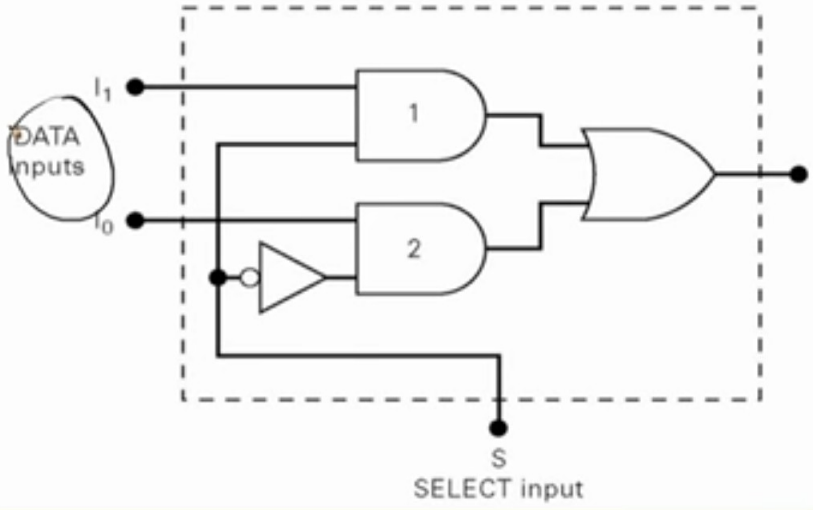
In simple terms, this is what a multiplexer (MUX) does; it selects one of several input signals and passes it on to the output.

Functional diagram of MUX



SELECT input code determines which input is transmitted to output Z.

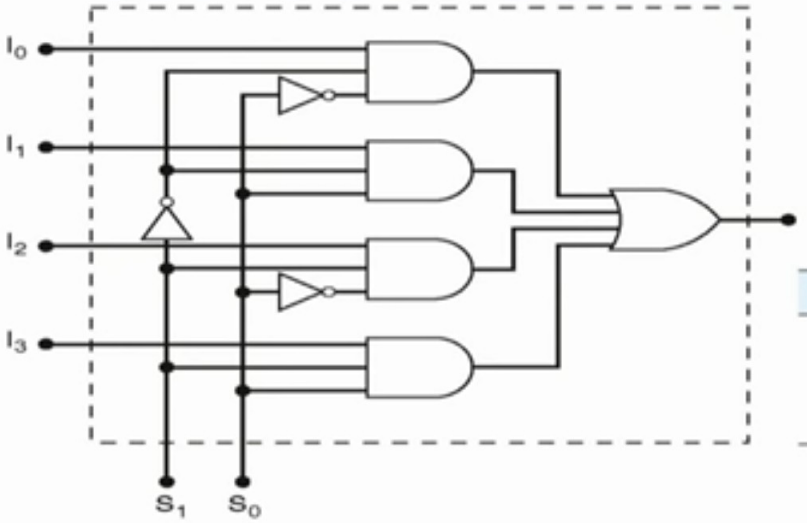
Two-input multiplexer



$$Z = I_0 \cdot \bar{S} + I_1 \cdot S$$

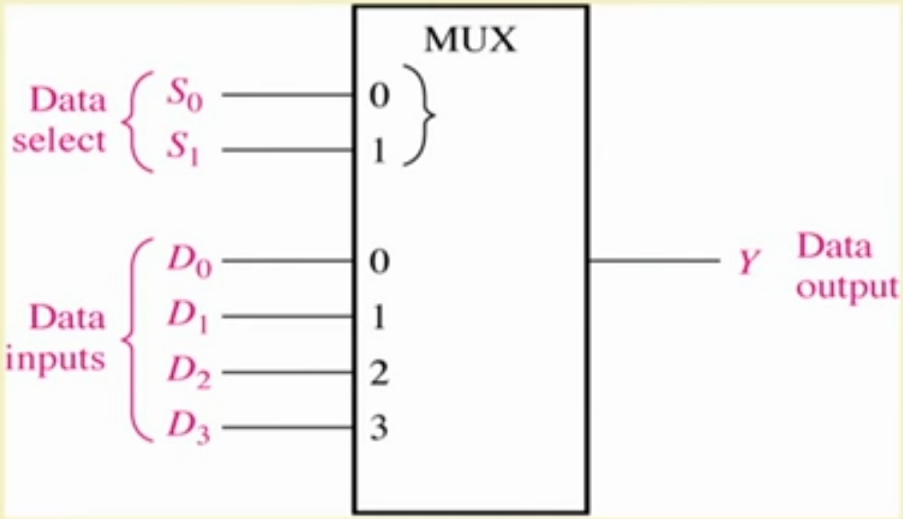
S	Output
0	$Z = I_0$
1	$Z = I_1$

Four-input multiplexer



S_1	S_0	Output
0	0	$Z = I_0$
0	1	$Z = I_1$
1	0	$Z = I_2$
1	1	$Z = I_3$

Logic symbol for a 1-of-4 data selector/multiplexer.



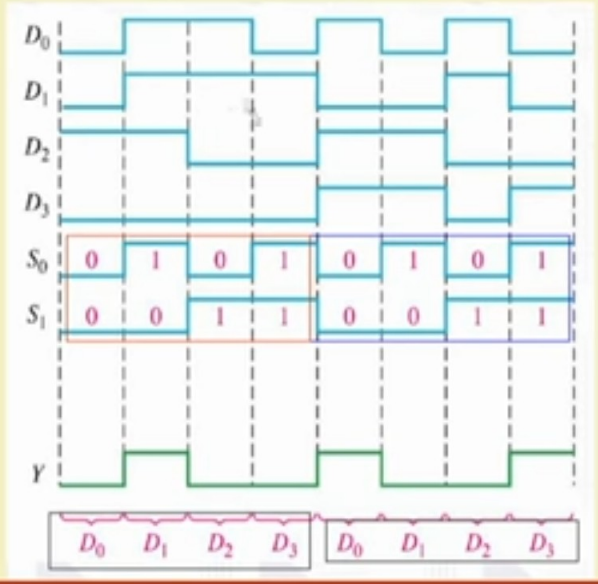
Logic symbol for a 1-of-4 data selector/multiplexer.

FUNCTION TABLE

INPUTS						STROBE $\overline{G}$	OUTPUT Y
SELECT		DATA					
B	A	C0	C1	C2	C3		
S1	S0	X	X	X	X	H	Z
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

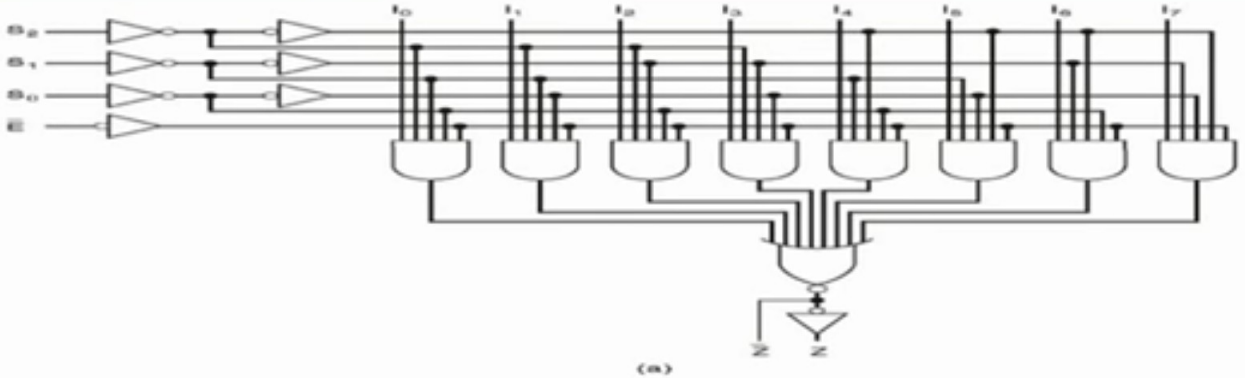
Select inputs A and B are common to both sections.

Output Waveforms in relation with the Data-Input and Data-Select waveforms - 4-input MUX



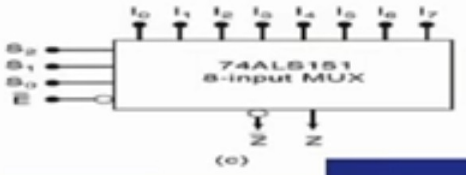
The binary state of the data-select inputs during each interval determines which data input is selected. Here the data-select inputs go through a repetitive binary sequence 00,01,10,11,00, and so on. The resulting output waveform is shown.

8-input multiplexer



Inputs				Outputs	
$\overline{E}$	S_2	S_1	S_0	Z	$\overline{Z}$
H	X	X	X	L	H
L	L	L	L	I_0	$\overline{I_0}$
L	L	L	H	I_1	$\overline{I_1}$
L	L	H	L	I_2	$\overline{I_2}$
L	L	H	H	I_3	$\overline{I_3}$
L	H	L	L	I_4	$\overline{I_4}$
L	H	L	H	I_5	$\overline{I_5}$
L	H	H	L	I_6	$\overline{I_6}$
L	H	H	H	I_7	$\overline{I_7}$

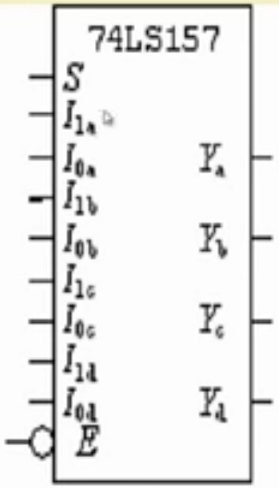
(b)



(c)

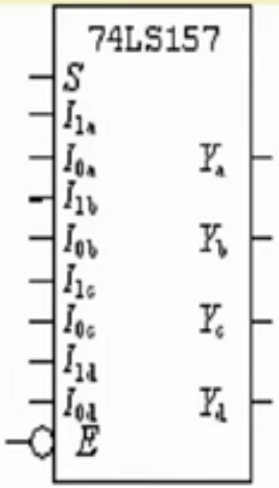
MSI Quad Two–input Multiplexer

- The **74LS157** contain of quad two–input multiplexers, $I_{0a}\, I_{0b}\, I_{0c}\, I_{0d}$ and $I_{1a}\, I_{1b}\, I_{1c}\, I_{1d}$.
- The logic symbol and truth table is shown in Figure.
- Notice that each of the four multiplexer shares a common data select line and a common *Enable.*
- Each multiplexer has only one data select input because there are only two groups of inputs to be selected.



Inputs		Outputs			
$\overline{E}$	S	Y_a	Y_b	Y_c	Y_d
H	X	L	L	L	L
L	L	I_{0a}	I_{0b}	I_{0c}	I_{0d}
L	H	I_{1a}	I_{1b}	I_{1c}	I_{1d}

Logic symbol and truth table for 74LS157 multiplexer.



Inputs		Outputs			
$\overline{E}$	S	Y_a	Y_b	Y_c	Y_d
H	X	L	L	L	L
L	L	I_{0a}	I_{0b}	I_{0c}	I_{0d}
L	H	I_{1a}	I_{1b}	I_{1c}	I_{1d}

Logic symbol and truth table for 74LS157 multiplexer.

MSI Quad Two–input Multiplexer

- $\overline{E}$ Input is *LOW* – allows the selected input data to pass through to the output.
- $\overline{E}$ Input is *HIGH* – will disable the multiplexers, all of the outputs will be *LOW*.
- When $\overline{E}$ = 0 and S = 1, the Y outputs will follow the set of I_1 inputs, that is $Y_a = I_{1a}$, $Y_b = I_{1b}$, $Y_c = I_{1c,}$ and $Y_d = I_{1d}$.

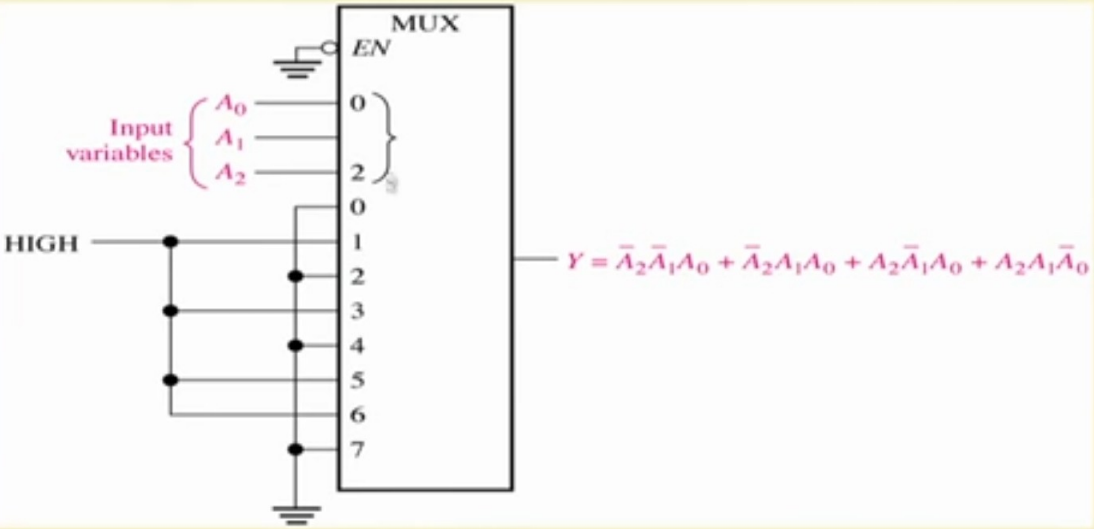
LOGIC FUNCTION GENERATION USING MUX

Exercise 1:

Implement the logic circuit function specified in the table given below by using 74LS151 8-input data selector/multiplexer.

Input			Output	
A2	A1	A0	Y	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	1	3
1	0	0	0	4
1	0	1	1	5
1	1	0	1	6
1	1	1	0	7

Solution :

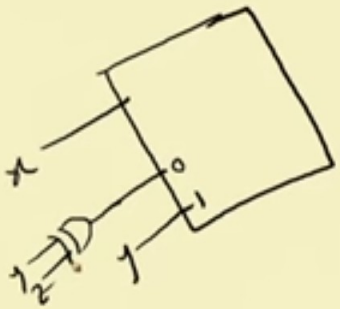

MUX
EN
Input variables
A_0
A_1
A_2
0
2
0
HIGH
1
2
3
4
5
6
7
$Y = \bar{A}_2\bar{A}_1A_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + A_2A_1\bar{A}_0$

LOGIC FUNCTION GENERATION USING MUX-Method

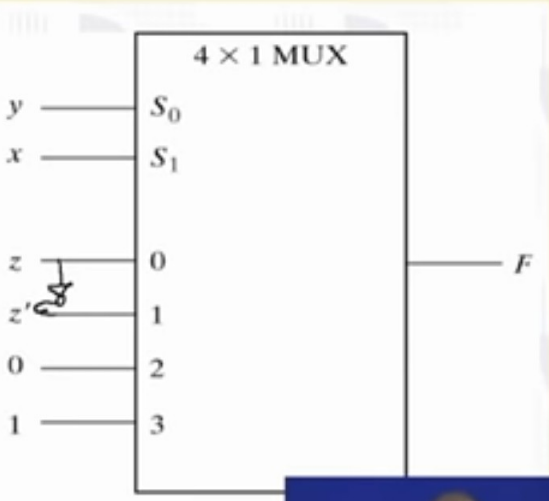
- An efficient method for implementing a Boolean function of n variables with a MUX that has n-1 selection inputs and 2^{n-1} data inputs is given below:
  - List the Boolean function in a truth table
  - Apply the first n-1 variables in the table to the selection inputs of the MUX.
  - For each combination of the selection variables, evaluate the output as a function of the last variable. This function can be 0,1, the variable, or the complement of the variable. Apply these values to the data inputs in the proper order.

LOGIC FUNCTION GENERATION USING MUX-Example 2

Implement the Boolean function F=x'y'z +x'yz' + xyz' + xyz
using a suitable MUX

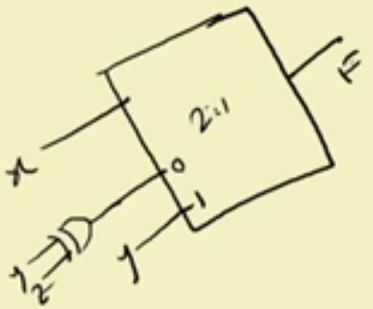


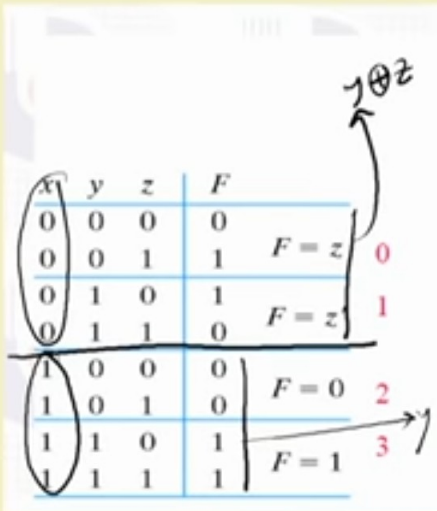
x	y	z	F		
0	0	0	0	$F = z$	0
0	0	1	1		
0	1	0	1	$F = z'$	1
0	1	1	0		
1	0	0	0	$F = 0$	2
1	0	1	0		
1	1	0	1	$F = 1$	3
1	1	1	1		

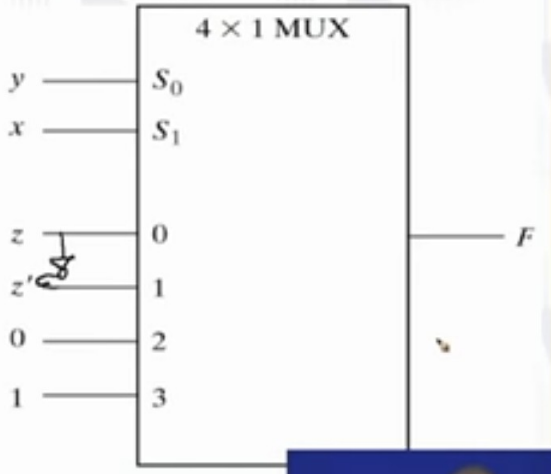


LOGIC FUNCTION GENERATION USING MUX-Example 2

Implement the Boolean function F=x'y'z +x'yz' + xyz' + xyz
using a suitable MUX







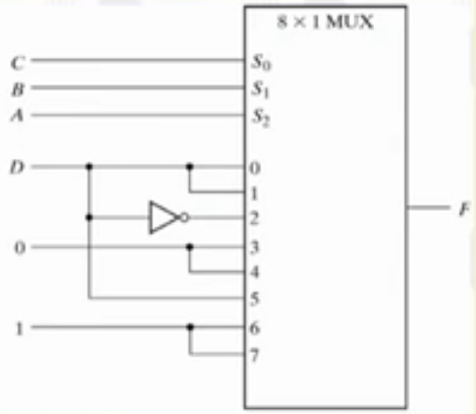
LOGIC FUNCTION GENERATION USING MUX-Example 2

- The two variables x and y are applied to the selection lines in that order; x is connected to the S_1 input and y to the S_0 input.
- The values for the data input lines are determined from the truth table of the function
  - For ex., when xy=00, output F is equal to z because F=0 when z=0 and F=1 when z=1. This requires that variable z is applied to the data input 0

LOGIC FUNCTION GENERATION USING MUX-Example 3

Implement the Boolean function F=A'B'C'D+A'B'CD+A'BC'D'
+AB'CD+ABC'D'+ABC'D+ABCD'+ABCD using a suitable MUX

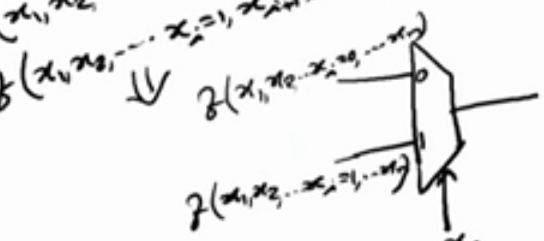
A	B	C	D	F		
0	0	0	0	0	$F = D$	0
0	0	0	1	1		
0	0	1	0	0	$F = D$	1
0	0	1	1	1		
0	1	0	0	1	$F = D'$	2
0	1	0	1	0		
0	1	1	0	0	$F = 0$	3
0	1	1	1	0		
1	0	0	0	0	$F = 0$	4
1	0	0	1	0		
1	0	1	0	0	$F = D$	5
1	0	1	1	1		
1	1	0	0	1	$F = 1$	6
1	1	0	1	1		
1	1	1	0	1	$F = 1$	7
1	1	1	1	1		



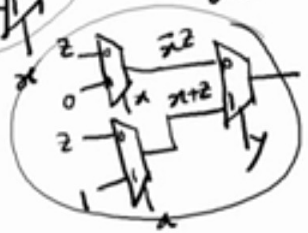
$f(x,y,z) = xy + \bar{x}z$

Shanon Decomposition

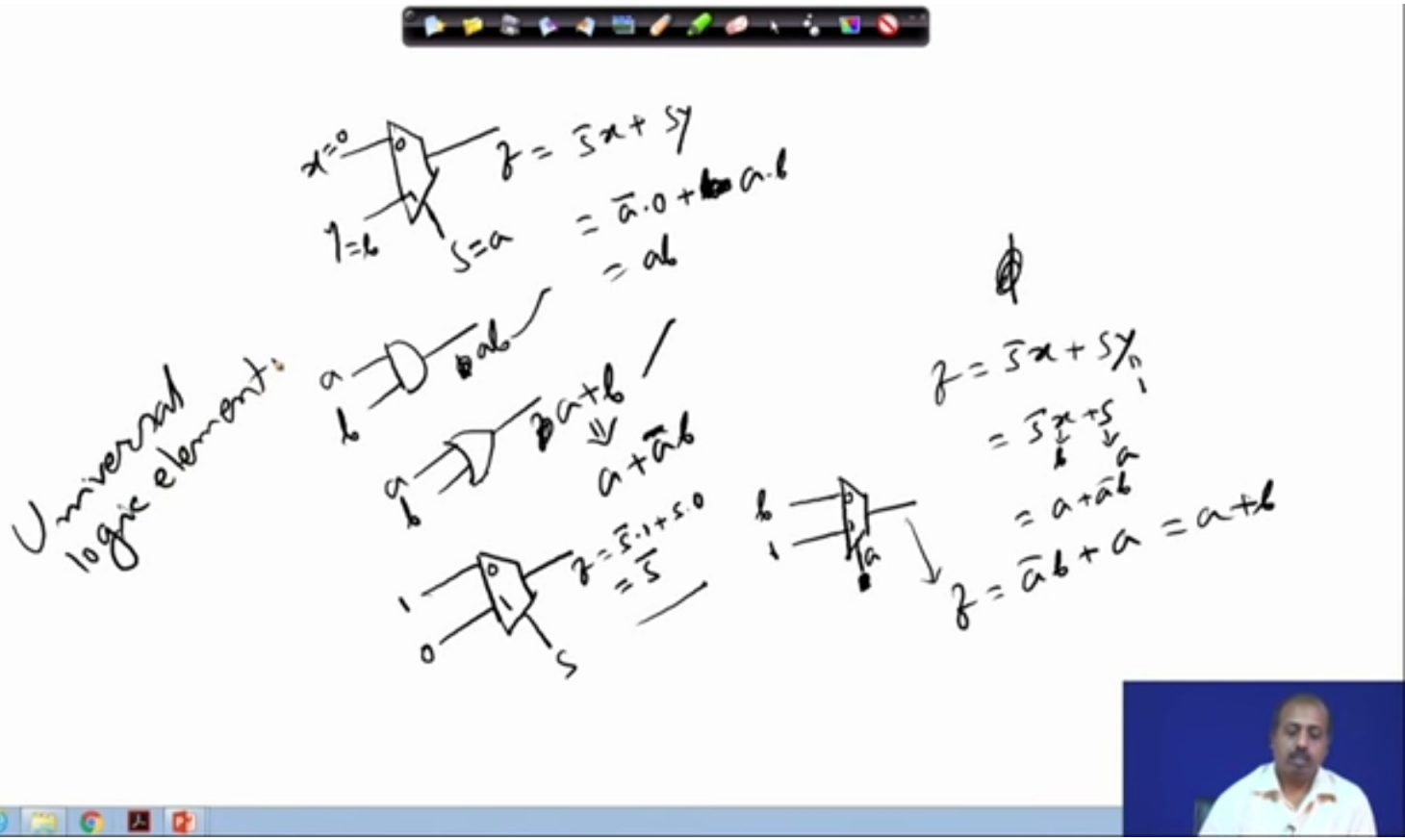
$f(x_1, x_2, \ldots x_n) = \bar{x_i} \cdot f(x_1, x_2, \ldots x_i=0, x_{i+1}, \ldots x_n) +$
$x_i \cdot f(x_1, x_2, \ldots x_i=1, x_{i+1} \ldots x_n)$



$f(x,y,z)$
$= \bar{x} \cdot f(x=0, y, z) +$
$x \cdot f(x=1, y, z)$
$= \bar{x} \cdot z + x \cdot y$

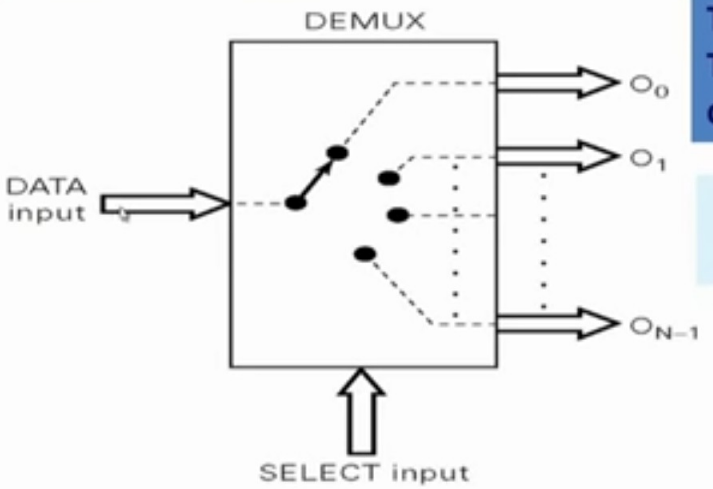


$f(x,y,z) = \bar{y} \cdot f(x, y=0, z) +$
$y \cdot f(x, y=1, z)$
$= \bar{y} \cdot (\bar{x}z) + y(x + \bar{x}z)$
$= \bar{y}(\bar{x}z) + y(x+z)$


Universal logic element:
x=0
y=b
S=a
z = S̄x + Sy
= ā.0 + a.b
= ab
a
b
ab
a
b
a+b
a+āb
1
0
S
z = S̄.1 + S.0
= S̄
b
1
a
z = S̄x + Sy
= S̄x + Sā
= a + āb
z = āb + a = a + b

General demultiplexer

Functional diagram:-

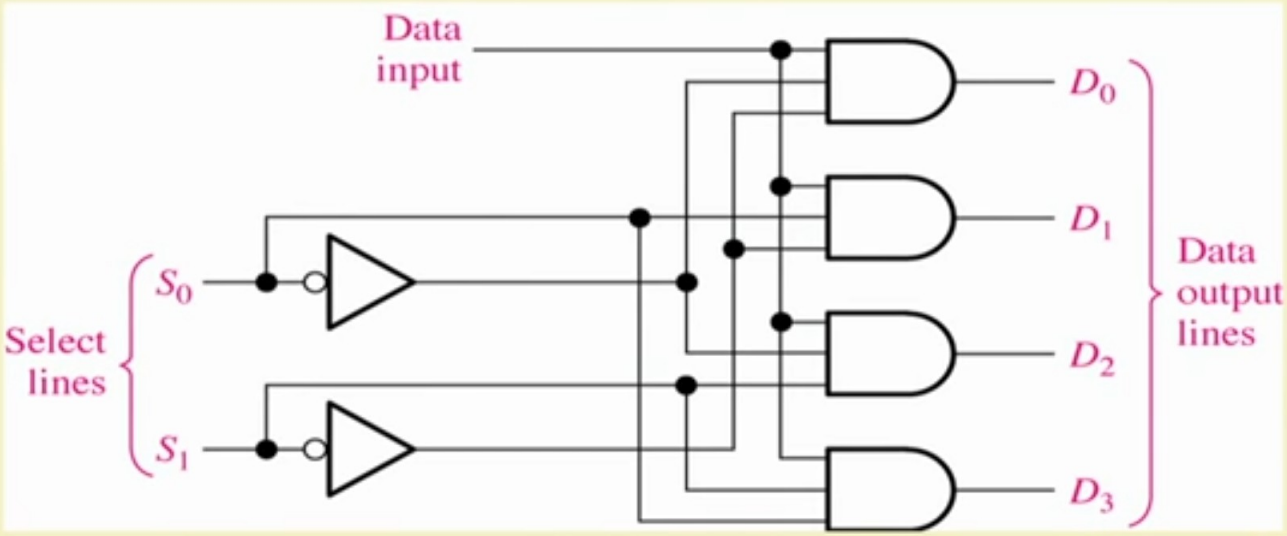


The large arrow indicates one or more lines. The select i/p code determines to which output the DATA input will be transmitted

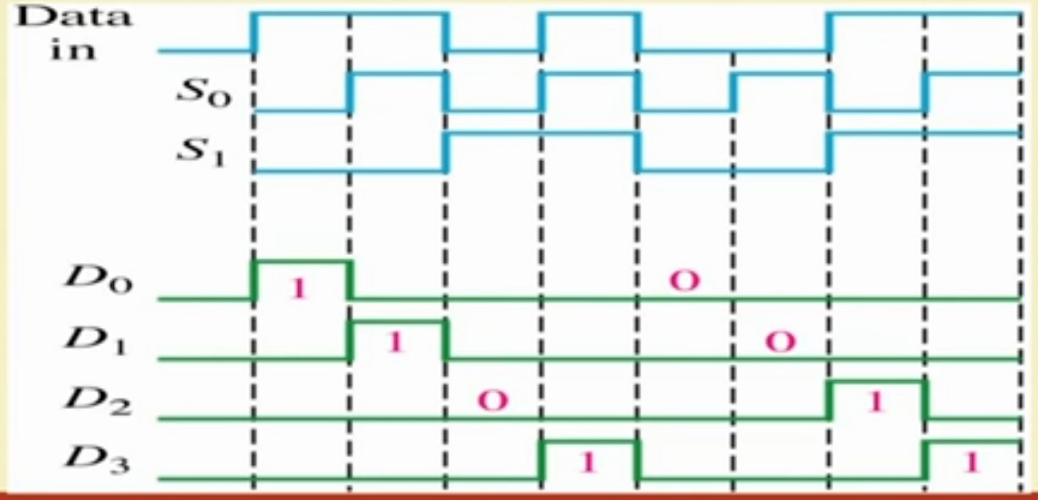
DATA input is transmitted to only one of the outputs as determined by select input code

In other words, the demultiplexer takes one data input source and selectively distributes it to 1 of N output channels just like multiposition switch

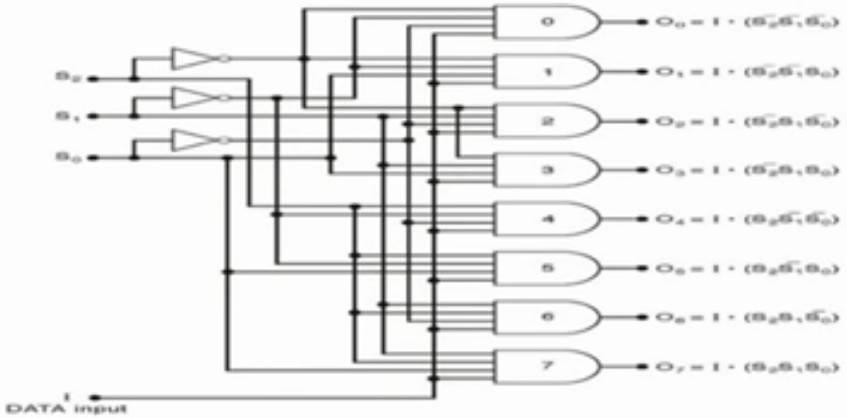
A 1-line-to-4-line demultiplexer.



The serial data input waveform (Data in) and data select inputs (S_o and S_1) and the corresponding data output waveforms (D_0 through D_3) are shown below



1-line- to-8 line demultiplexer

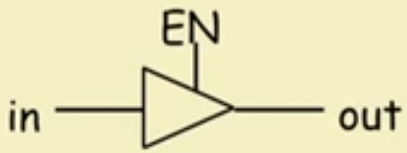


SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Note: I is the data input

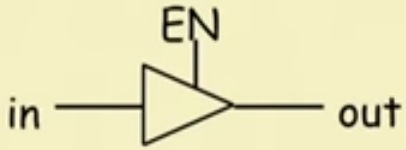
Buffers

- Buffer:
  - Doesn't change the input.
  - Only amplifies.



Three-State Buffers

- Buffer output has 3 states: 0, 1, Z
- Z stands for High-Impedance ≅ Open circuit

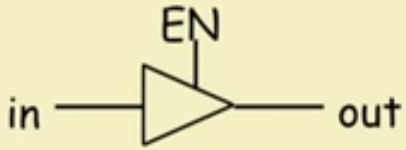


EN	in	out
0	X	Z
1	0	0
1	1	1

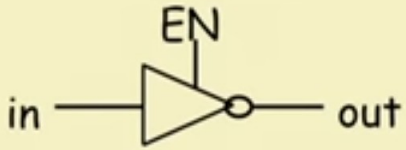
EN = 0 → out = Z (open circuit)

EN = 1 → out = in (regular buffer)

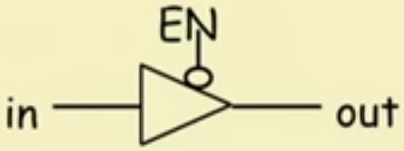
Three-state buffer(BUF)/inverter(INV) symbols



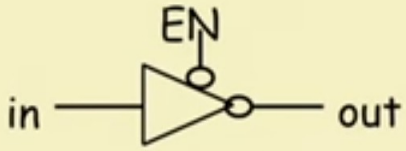
3-state BUF, EN high



3-state INV, EN high



3-state BUF, EN low



3-state INV, EN low