EEE 3101: Digital Logic and Circuits

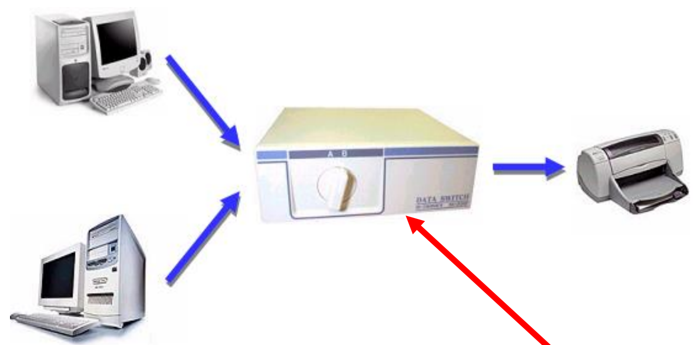
Multiplexer & De-Multiplexer

Course Teacher: Nafiz Ahmed Chisty

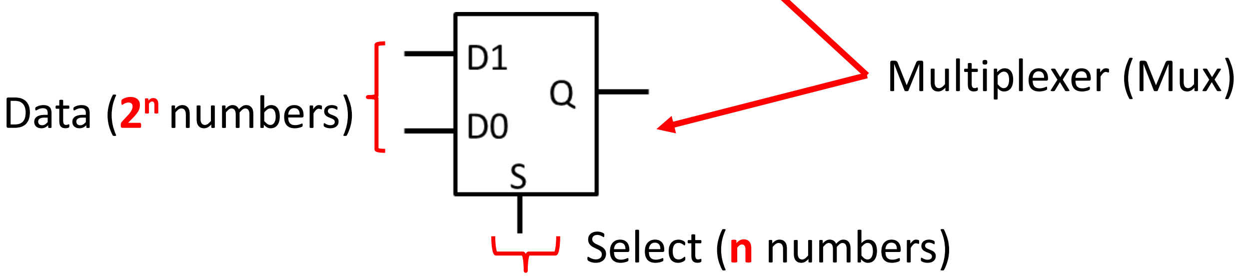
Associate Professor, Department of EEE & CoE
Head (UG), Department of EEE
Faculty of Engineering
Room# DNG03, Ground Floor, D Building
Email: chisty@aiub.edu
Website: http://engg.aiub.edu/
Website: www.nachisty.com

Multiplexer or Mux or Data Selector

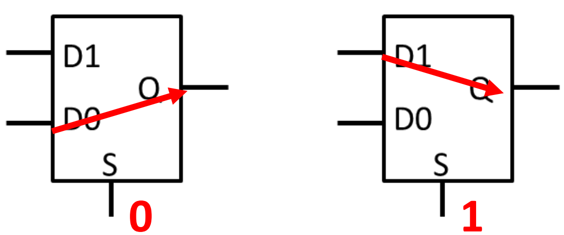
In the old days, several machines could share an I/O device with a Switch. The Switch allows one computer's output to go to the printer's input.



Multiplexer (or mux), also known as a data selector, is a device that selects between several input signals and forwards the selected input to a single output line. A multiplexer of 2^n inputs has n select lines, which are used to select which input line to send to the output.



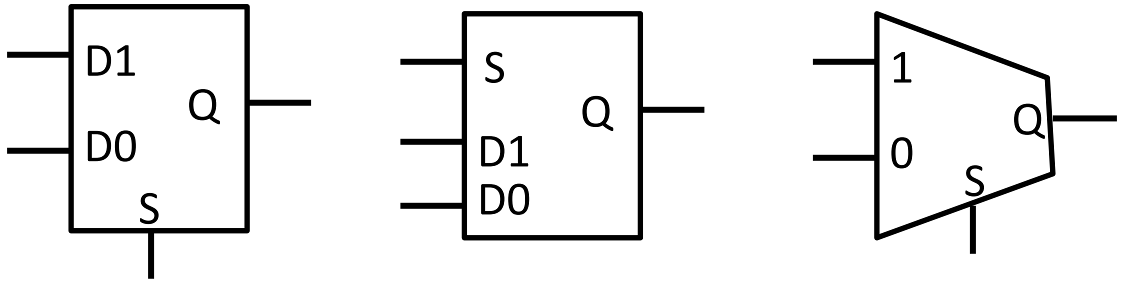
- This is a 2-to-1 multiplexer or mux.
- The multiplexer routes one of its data inputs (D0 or D1) to the output Q, based on the value of S:
  - If **S=0**, then D0 is the output (**Q=D0**).
  - If **S=1**, then D1 is the output (**Q=D1**).



Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

Block diagram, Truth table and Circuit

- Block Diagram:



- Truth table:

S	D1	D0	Q
0	0	0	0
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	1

S	Q
0	D0
1	D1

Q = S’ D0 + S D1

When **S=0**

Q = 0’ D0 + 0 D1

Q = 1 D0 + 0

Q = D0

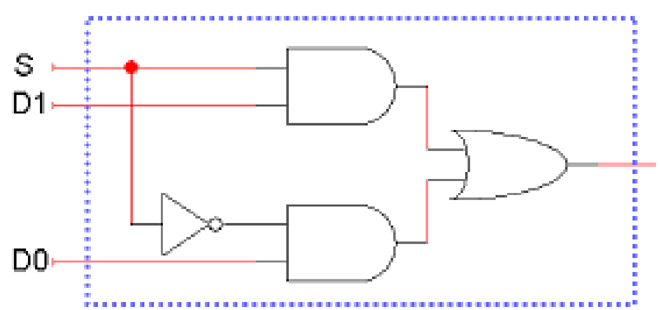
When **S=1**

Q = 1’ D0 + 1 D1

Q = 0 D0 + 1 D1

Q = D1

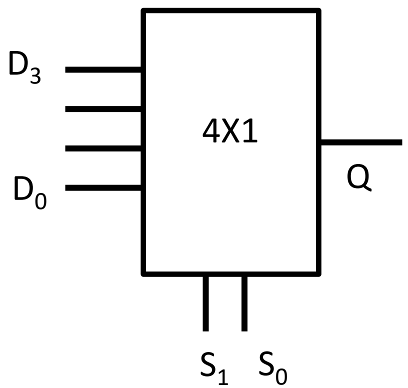
- Circuit Diagram:



Q = S’ D0 + S D1

4x1 Multiplexer

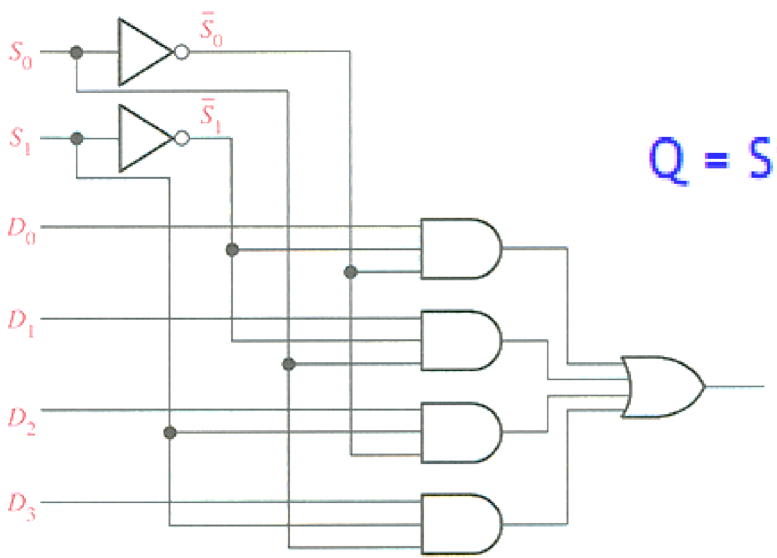
- Block Diagram:



- Truth table:

Select Input		Output (selected input)
S_1	S_0	Q
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

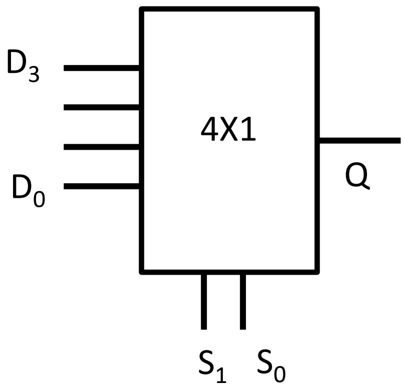
- Circuit Diagram:

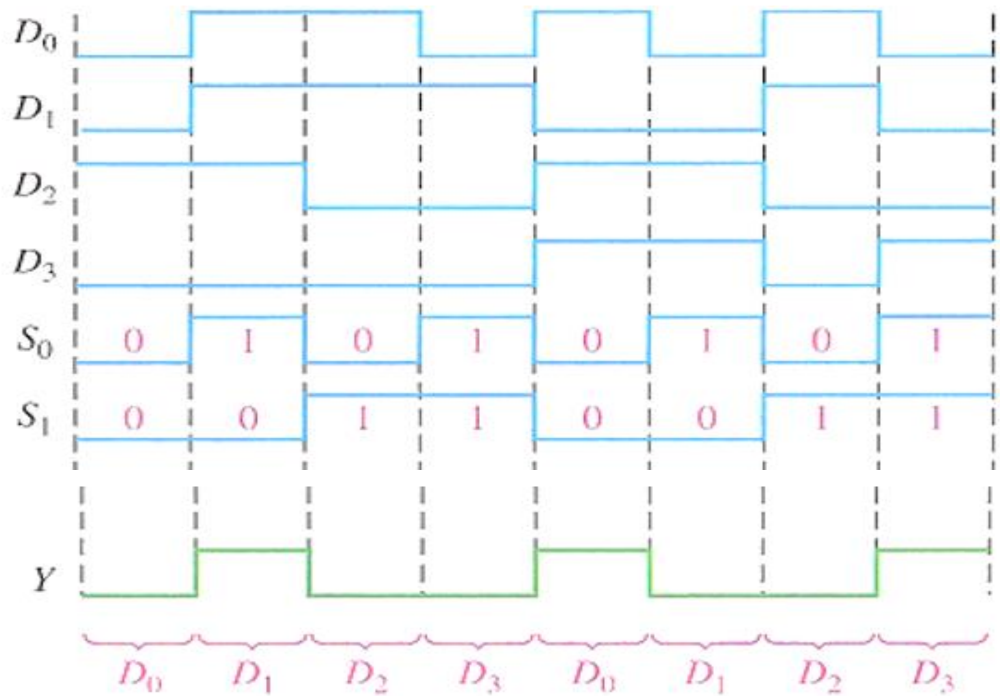


Q = S1'S0'D0 + S1'S0 D1 + S1 S0'D2 + S1 S0 D3

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

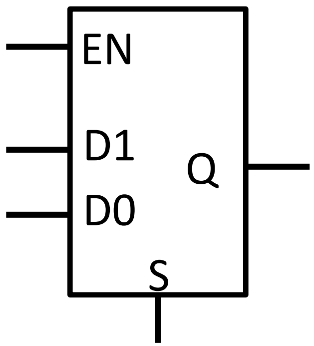
Timing Diagram





Select Input		Output (selected input)
S_1	S_0	Q
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Enable inputs



- Many devices have an additional enable input, which "activates" or "deactivates" the device.
- We could design a 2-to-1 multiplexer with an enable input that's used as follows.
  - EN=0 disables the multiplexer, which forces the output to be 0. (It does *not* turn off the multiplexer.)
  - EN=1 enables the multiplexer, and it works as specified earlier.
- Enable inputs are especially useful in combining smaller muxes together to make larger ones, as we'll see later today.

EN	S	D1	D0	Q
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

EN	S	D1	D0	Q
0	x	x	x	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

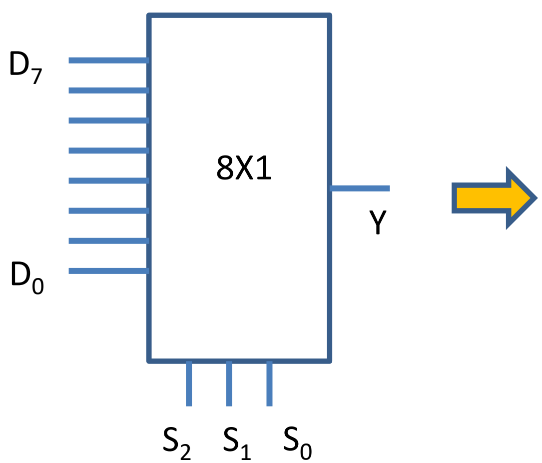
EN	S	Q
0	x	0
1	0	D0
1	1	D1

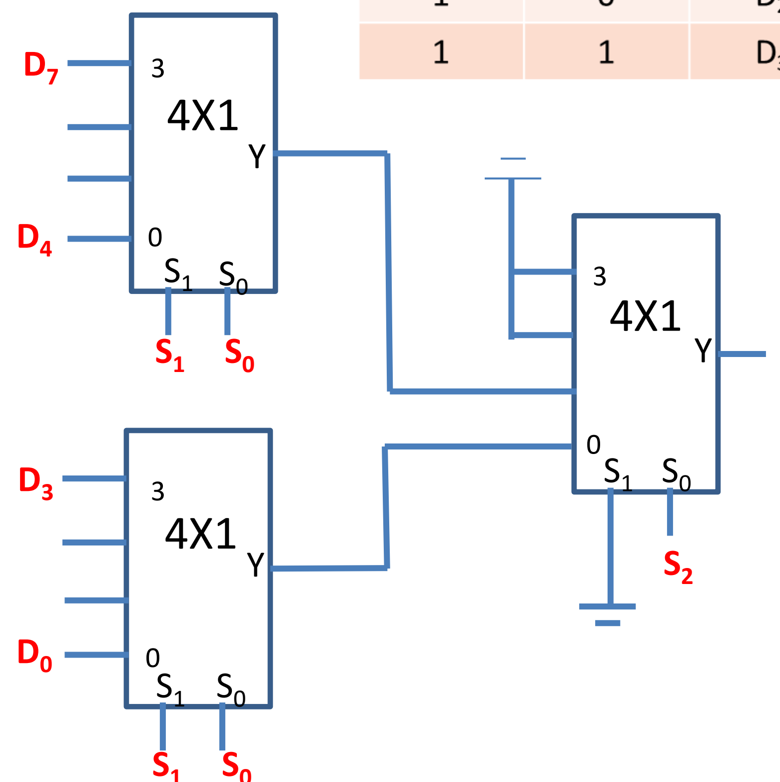
Implementation of Higher-order Multiplexers using lower-order Multiplexers

8x1 Mux using ONLY 4x1 Mux

Select Input			Output (selected input)
S_2	S_1	S_0	Q
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Select Input		Output (selected input)
S_1	S_0	Q
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3





Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

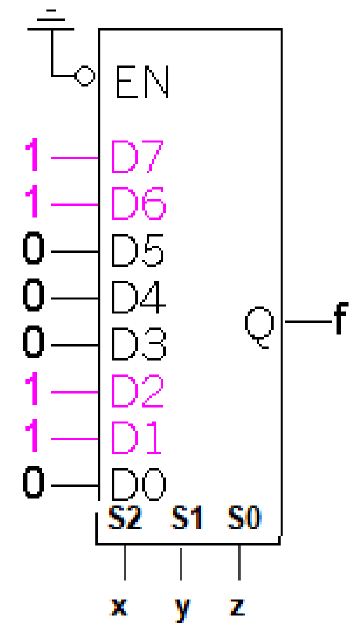
Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

Implementing functions with multiplexers

- Muxes can be used to implement arbitrary functions.
- **One way to implement a function of *n* variables is to use an *2^n-to-1* mux**:
- For example, let's look at **f(x,y,z) = Σ(1,2,6,7)**.

x	y	z	f	
0	0	0	0	→ D0
0	0	1	1	→ D1
0	1	0	1	→ D2
0	1	1	0	→ D3
1	0	0	0	→ D4
1	0	1	0	→ D5
1	1	0	1	→ D6
1	1	1	1	→ D7

Select Input			Output (selected input)
S_2	S_1	S_0	Q
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7



MULTIPLEXER

Boolean Function Implementation (advanced)

Question: Implement the following function with only one 4-to-1 multiplexer:

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

For **3 variables**, it takes:

a) **One** 8-to-1 Mux, or

b) **Three** 4-to-1 Mux

Only ONE 4-to-1 ..???

Procedure:

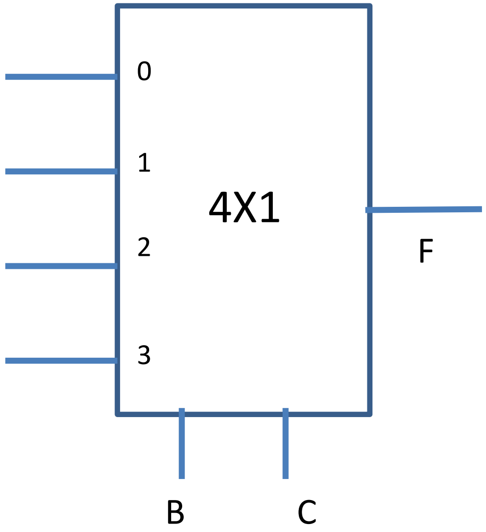
- Implement the truth table and write the SOP expression in the decimal format:

Minterms	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

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

- If the Boolean function has ***n+1*** Variables, then connect ***n*** of these variables to the select lines of a MUX maintain the order.

$$\boldsymbol{n+1 = 3}$$
$$\Rightarrow \boldsymbol{n = 2}$$



- Based on the select lines, find the total number of input lines for the MUX. The remaining variable will be used for the inputs of the MUX.

- Consider now the single variable ***A***. It can either be **0 or 1**.
- From the truth table, find the minterms for which **A** is **0**. The minterms are 0, 1, 2 & 3.
- From the truth table, find the minterms for which **A** is **1**. The minterms are 4, 5, 6 & 7.

Minterms	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

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

- List the inputs of the MUX and under them list all the minterms in two rows.
- The first row lists all those minterms where **A** is ***0***, and the second row all the minterms with **A** is ***1***.

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

- **Circle** all the minterms of the function and inspect **each column** separately.

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

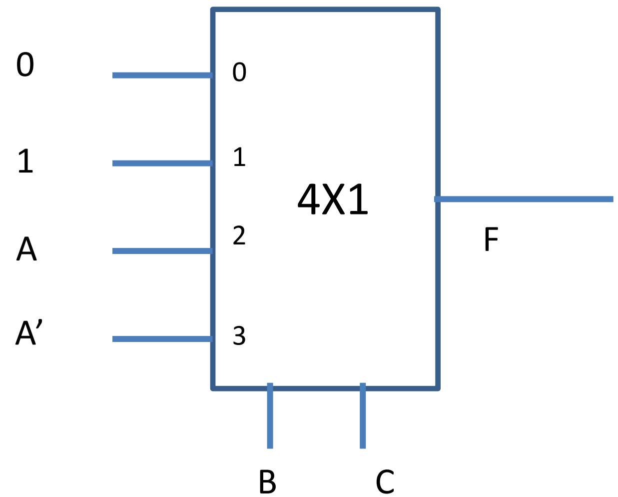
	I_0	I_1	I_2	I_3
A’	0	1	2	3
A	4	5	6	7
	0	1	A	A’

- If the two minterms in a column are not circled, **apply 0** to the corresponding MUX input.
- If the two minterms are circled, **apply 1** to the corresponding MUX input.
- If the bottom minterm is circled and the top is not circled, **apply A** (for this example) to the corresponding MUX input.
- If the top minterm is circled and the bottom is not circled, **apply A’** (for this example) to the corresponding MUX input.

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

Minterms	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

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

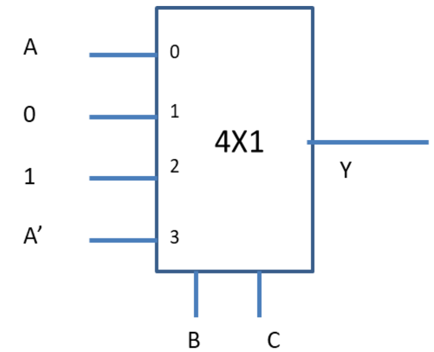


Example: Implement the following function using 4x1 MUX:

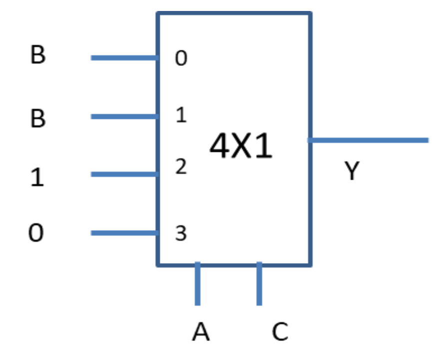
$$F(A,B,C)= \Sigma\ (2, 3, 4, 6)$$

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

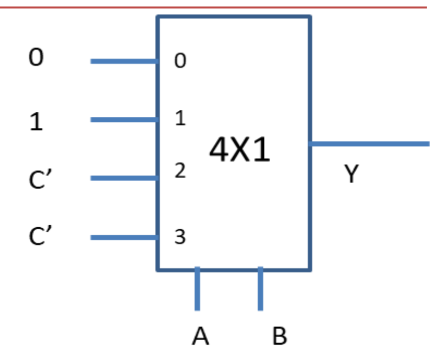
	I_0	I_1	I_2	I_3
A’	0	1	2	3
A	4	5	6	7
	A	0	1	A’



	I_0	I_1	I_2	I_3
B’	0	1	4	5
B	2	3	6	7
	B	B	1	0



	I_0	I_1	I_2	I_3
C’	0	2	4	6
C	1	3	5	7
	0	1	C’	C’



Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

Example:

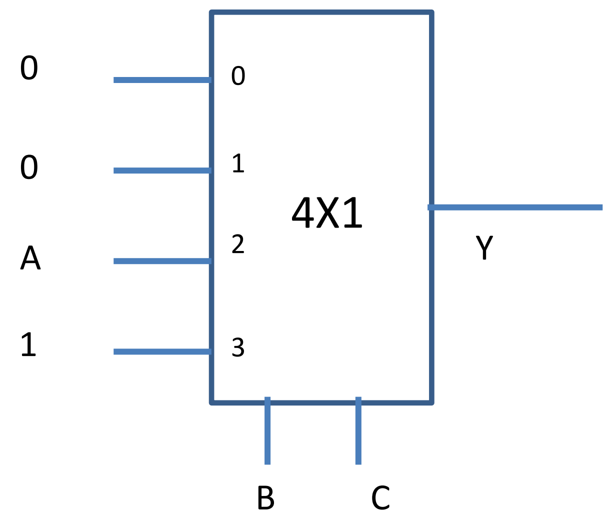
$$F(A,B,C) = AB+BC$$

$$\begin{aligned} F(A,B,C) &= AB+BC \\ &= AB(C+C')+(A+A')BC \\ &= ABC+ABC'+ABC+A'BC \\ &= \underset{7}{ABC}+\underset{6}{ABC'}+\underset{3}{A'BC} \end{aligned}$$

$$F(A,B,C)= \Sigma\,(3,6,7)$$

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

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

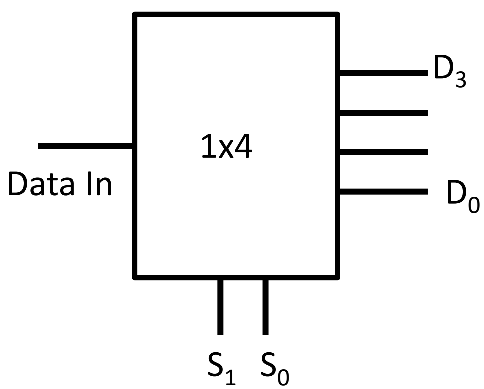


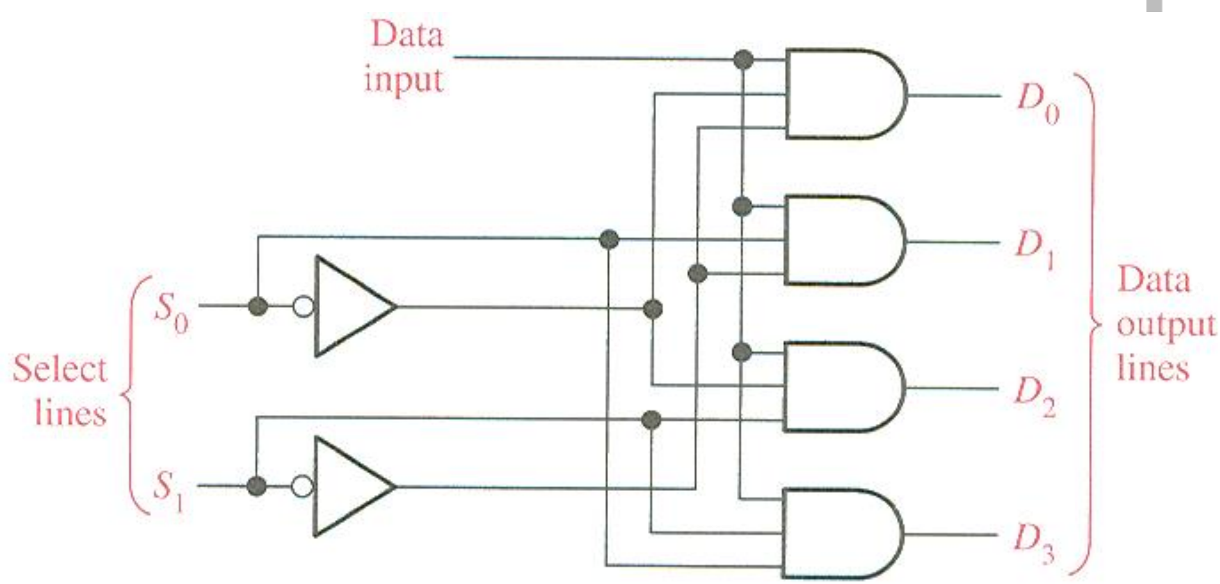
Demultiplexers

1-line-to 4-line demux

A **demultiplexer (DEMUX)** basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. As you will learn, decoders can also be used as demultiplexers.

DATA-SELECT INPUTS S_1	S_0	OUTPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3





$D_0=S_1'S_0'D_{in}$

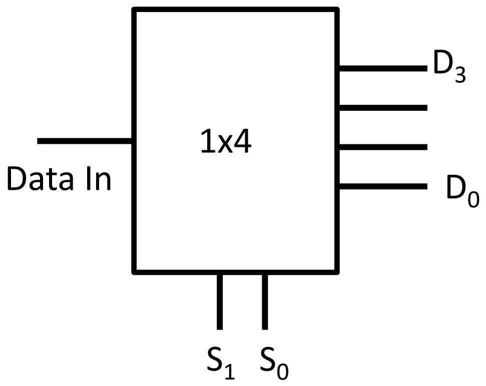
$D_1=S_1'S_0D_{in}$

$D_2=S_1S_0'D_{in}$

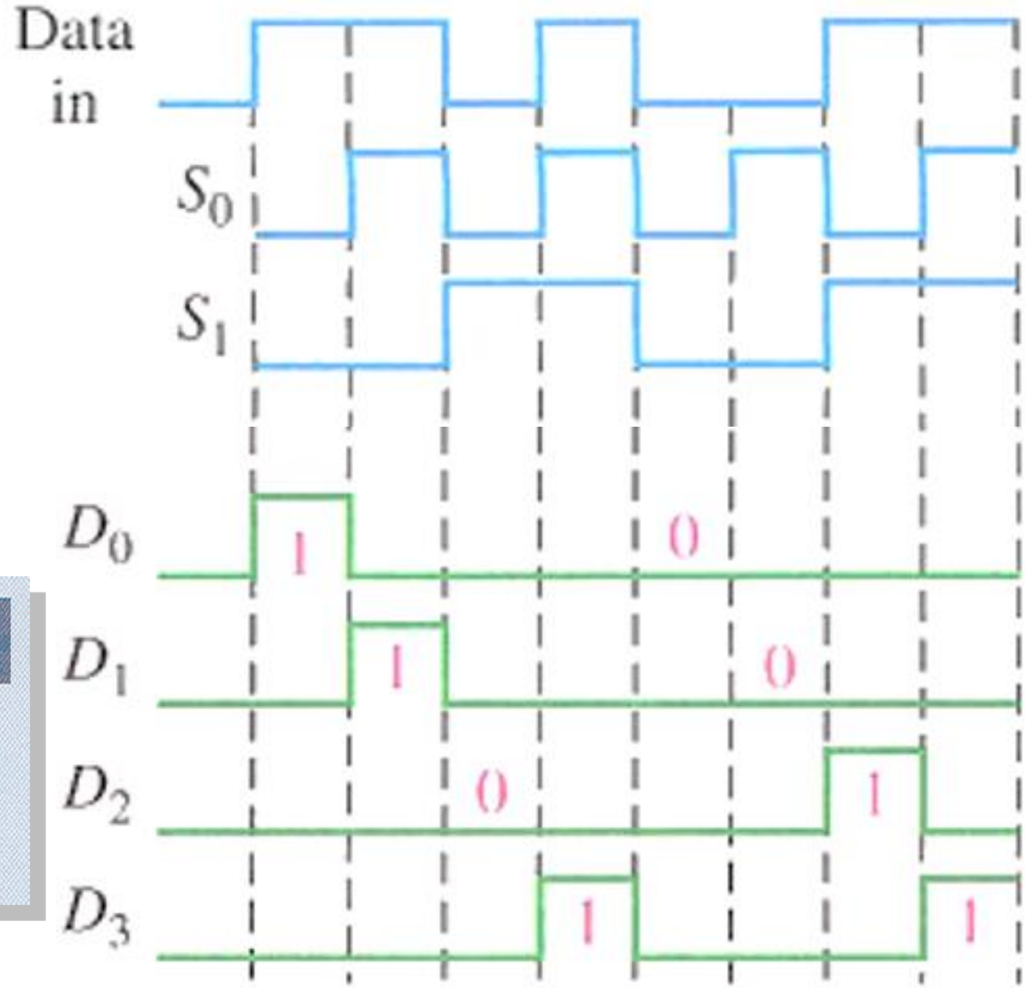
$D_3=S_1S_0D_{in}$

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

Timing Diagram

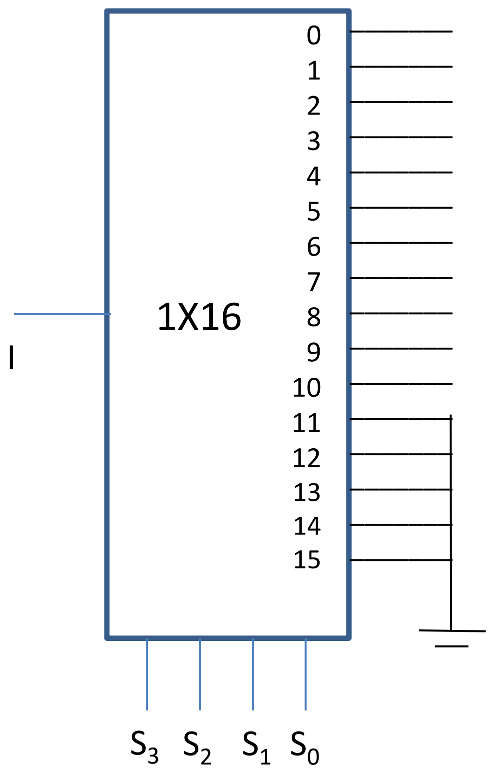


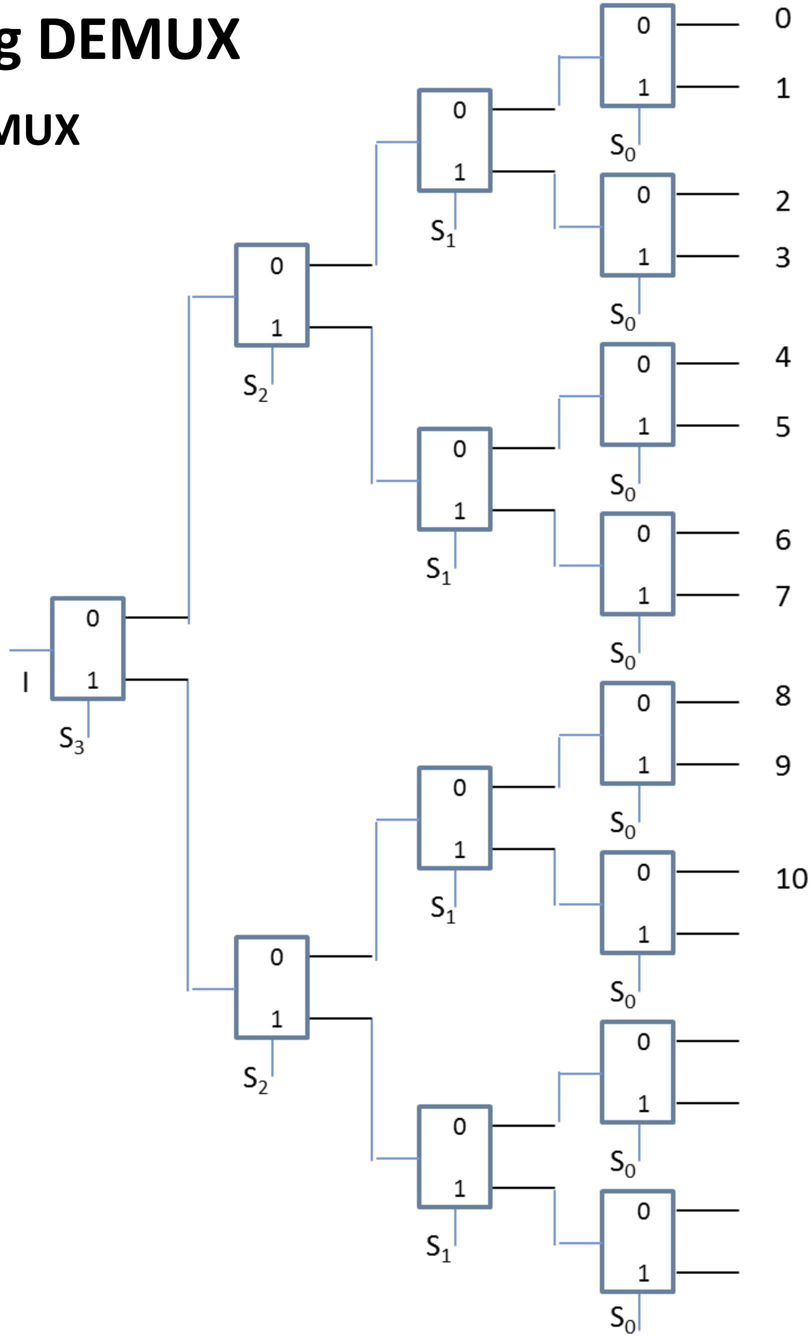
DATA-SELECT INPUTS S_1	S_0	OUTPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



Cascading DEMUX

Designing a 1x11 DEMUX using 1x2 DEMUX





Reference:

[1] Thomas L. Floyd, "Digital Fundamentals" 11th edition, Prentice Hall.
[2] M. Morris Mano, "Digital Logic & Computer Design" Prentice Hall.
[3] Mixed contents from Vahid And Howard.

Thanks