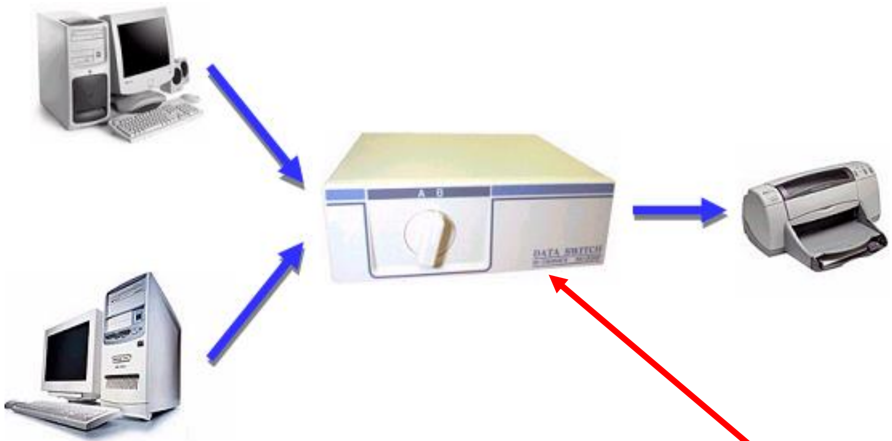
Digital Logic Design

Multiplexer & De-Multiplexer

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.

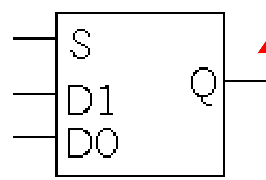


- Here is the circuit similar to that printer switch.

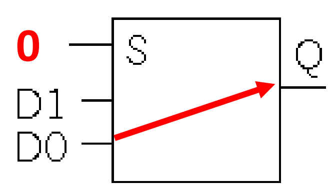
Multiplexer (Mux)

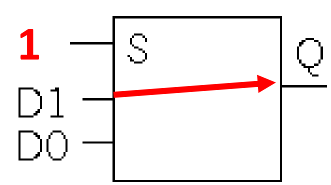
Select (**n** numbers)

Data (**2^n** numbers)



- 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**).





Truth table abbreviations, Block diagram and Circuit

- 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**

$Q = 0' D0 + 0 D1$

$Q = 1 D0 + 0$

$Q = D0$

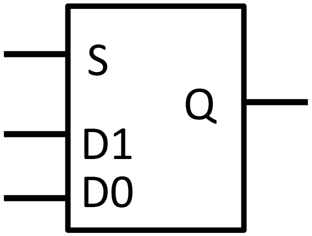
When **S=1**

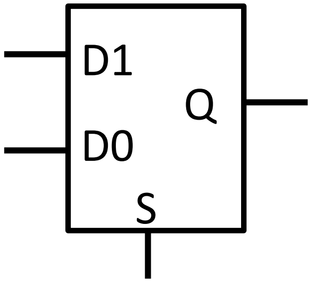
$Q = 1' D0 + 1 D1$

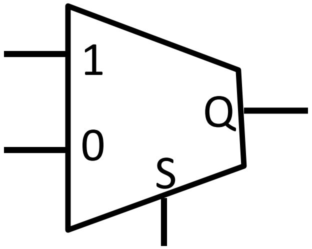
$Q = 0 D0 + 1 D1$

$Q = D1$

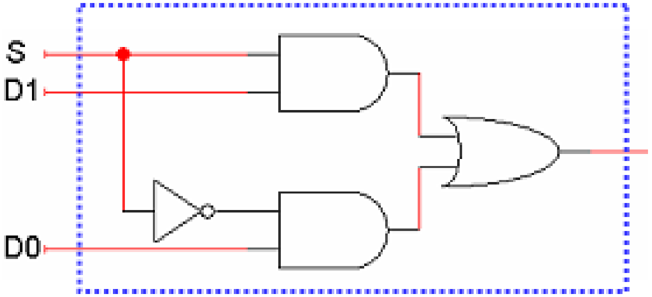
- Block Diagram:







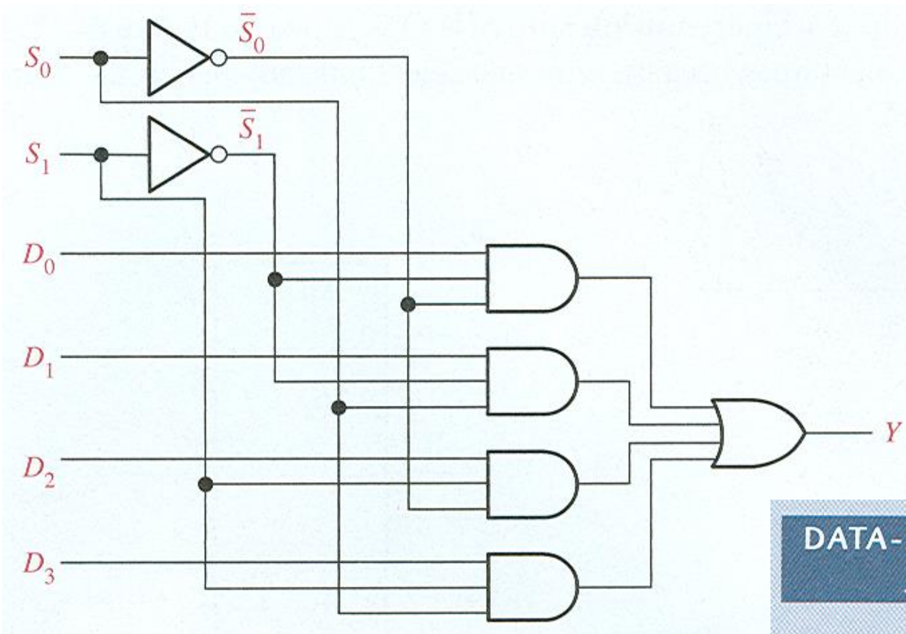
- Circuit Diagram:

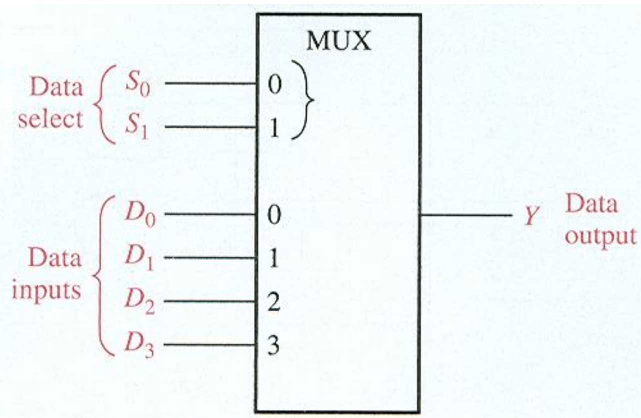


$Q = S' D0 + S D1$

4-input multiplexer

- Here is a block diagram and abbreviated truth table for a 4-to-1 mux, which directs one of four different inputs to the single output line.
  - There are four data inputs, so we need *two* bits, S1 and S0, for the mux selection input.

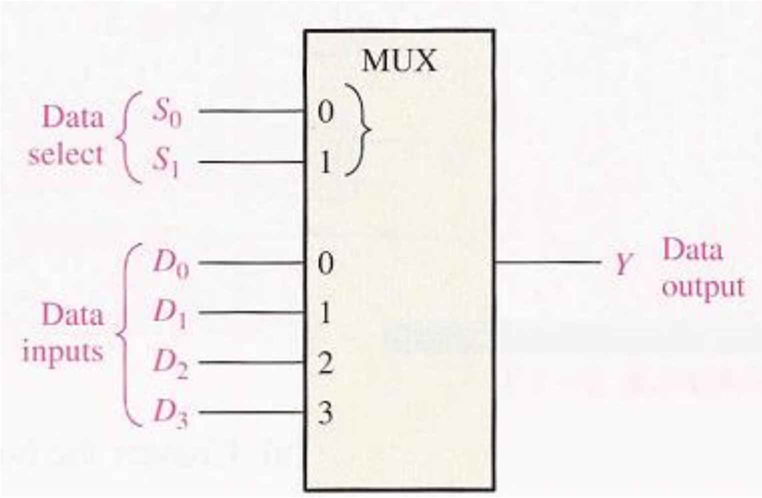




DATA-SELECT INPUTS S_1	S_0	INPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

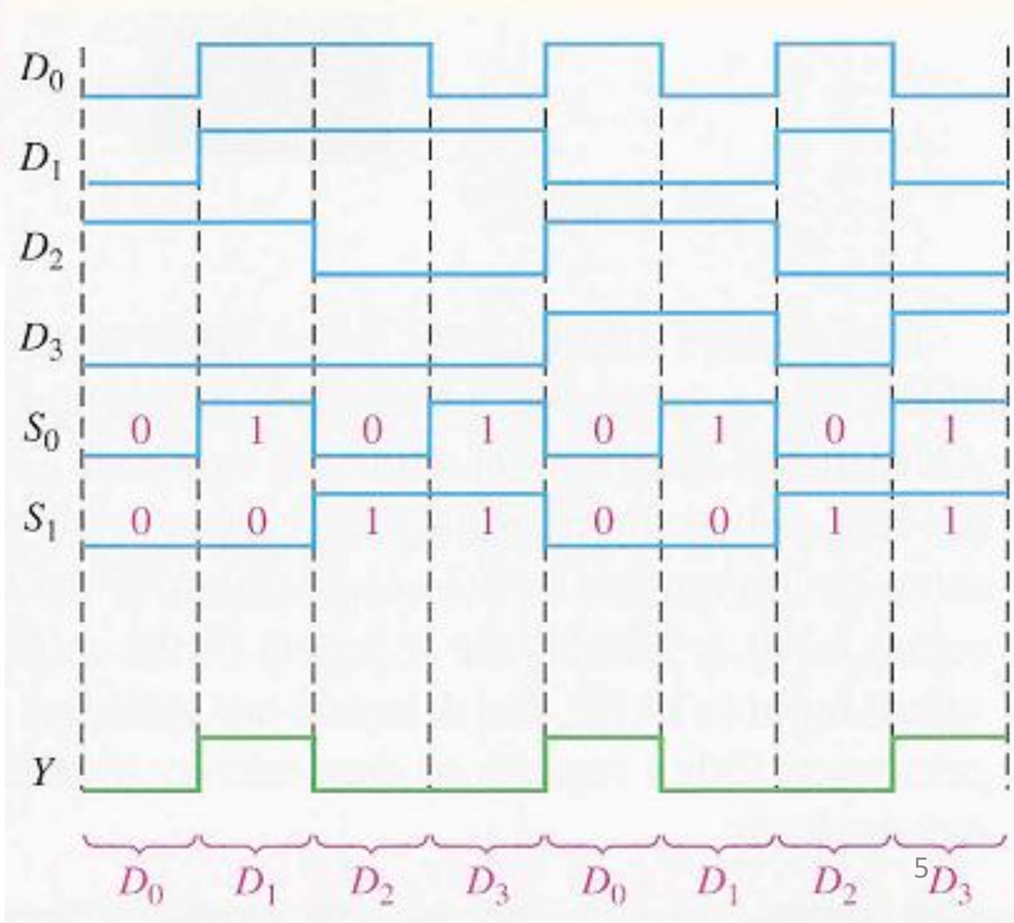
Q = S1'S0'D0 + S1'S0 D1 + S1 S0'D2 + S1 S0 D3

Timing Diagram



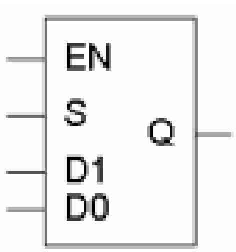
$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

DATA-SELECT INPUTS S_1	S_0	INPUT SELECTED
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

Truth table abbreviations

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

- Notice that when EN=0, then Q is always 0, regardless of what S, D1 and D0 are set to.
- We can shorten the truth table by including Xs in the input variable columns, as shown on the bottom right.

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

Another abbr. 4 U

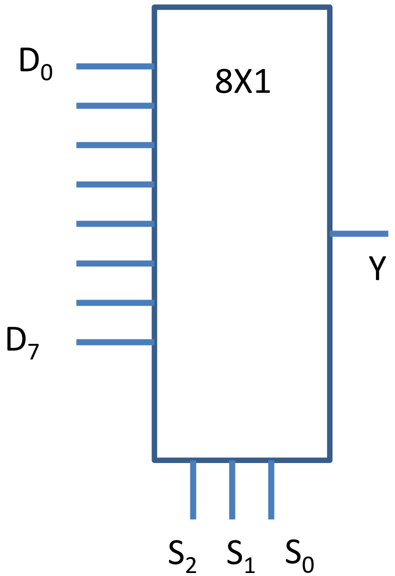
- Also, when EN=1 notice that if S=0 then Q=D0, but if S=1 then Q=D1.
- Another way to abbreviate a truth table is to list input variables in the output columns, as shown on the right.

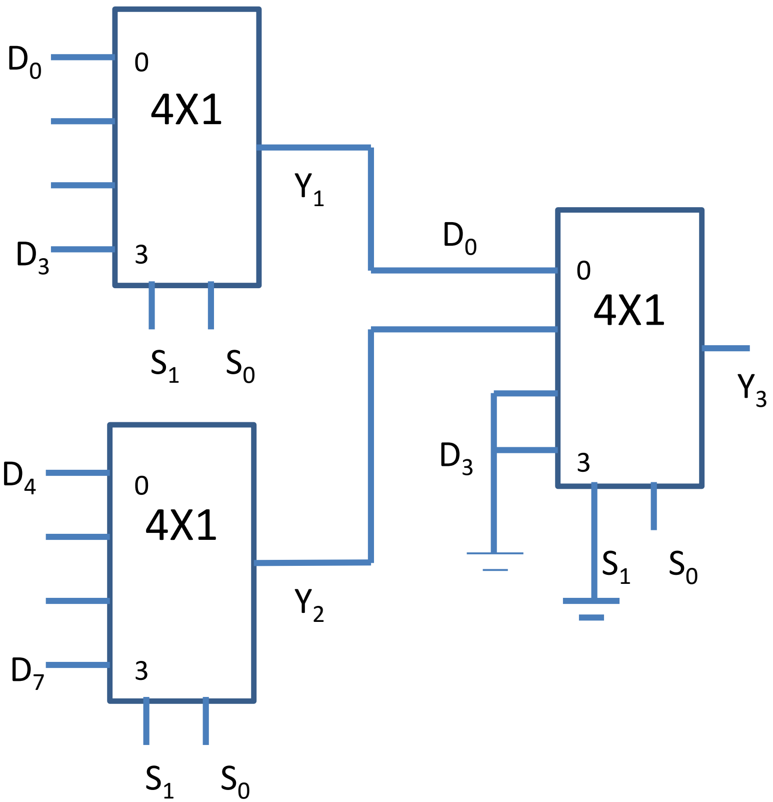
EN	S	D1	D0	Q
0	x	x	x	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

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

- This final version of the 2-to-1 multiplexer truth table is much clearer, and matches the equation $Q = S'D0 + S\,D1$ very closely.

8-to- 1 Mux using ONLY 4-to-1 Mux

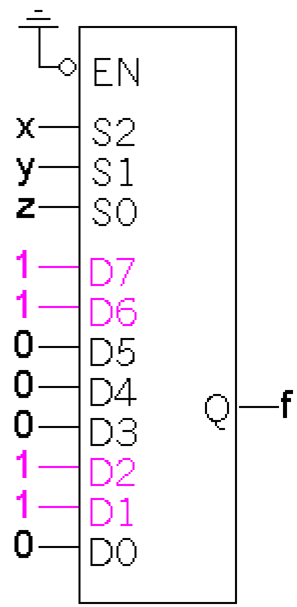




Implementing functions with multiplexers

- 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 **$f(x,y,z) = \Sigma(1,2,6,7)$.**

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



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

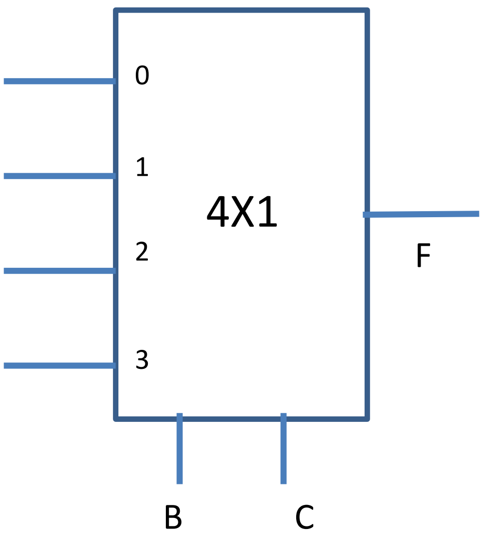
Only ONE 4-to-1 ..???

Procedure:

1) Implement the truth table,
2) Write the SOP expression in the decimal format, **F(A,B,C)= Σ (1, 3, 5, 6)**
3) If the Boolean function has ***n+1*** Variables, then connect ***n*** of these variables to the select lines of a MUX maintain the order.
4) 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.

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)= Σ (1, 3, 5, 6)



- 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***.

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

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

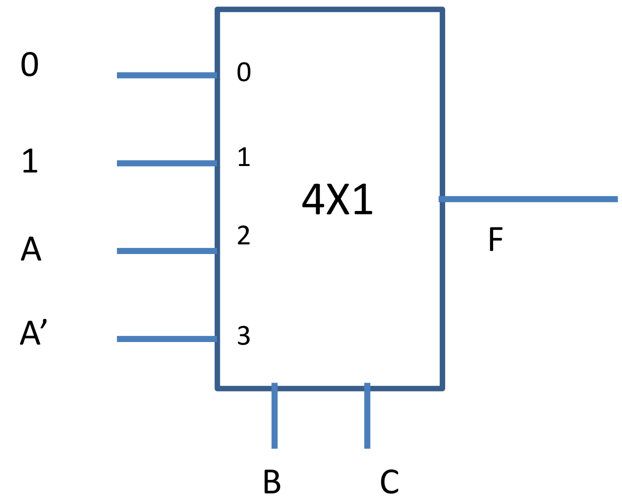
	I_0	I_1	I_2	I_3
A’	0	1	2	3
A	4	5	6	7
	0	1	A	A’

- 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** to the corresponding MUX input.
- If the top minterm is circled and the bottom is not circled, **apply A’** 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’

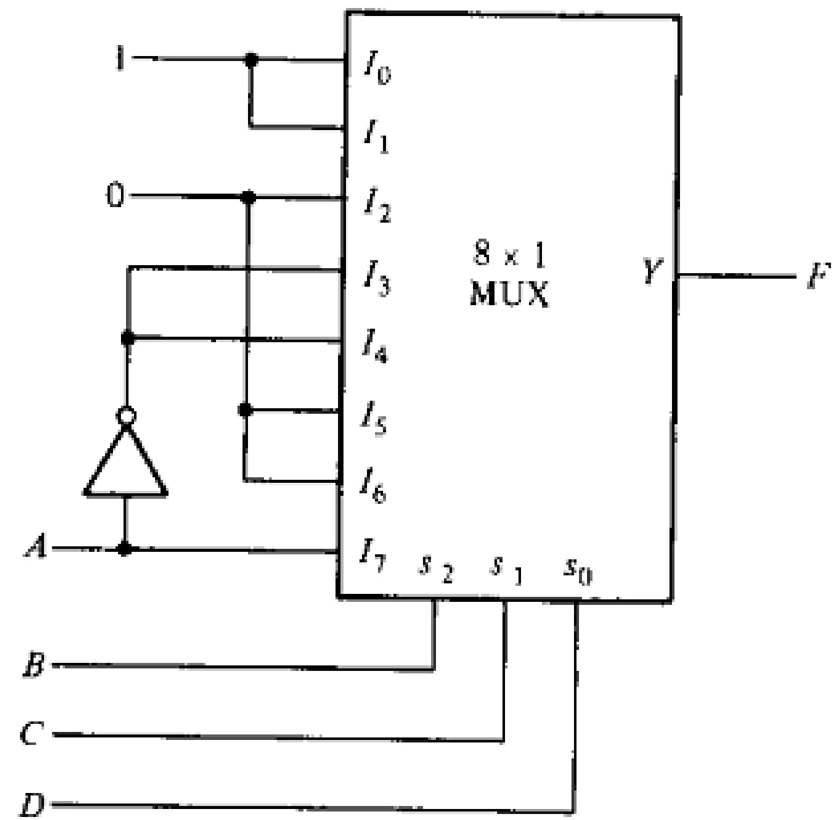


Example:

Implement the following function with **only one 8-to-1** multiplexer:

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

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A’	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A’	A’	0	0	A

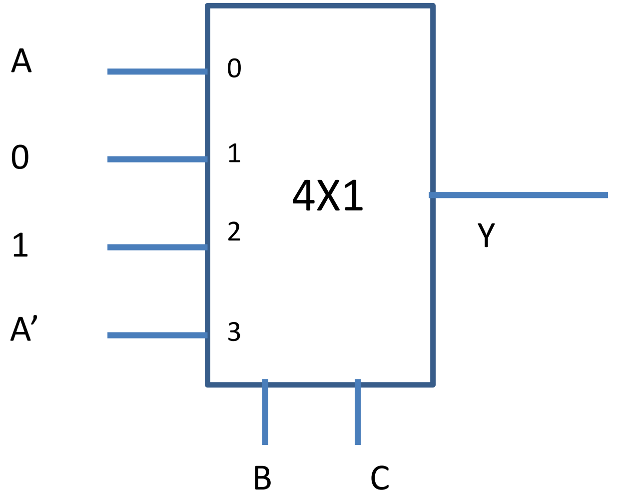


Example:

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'

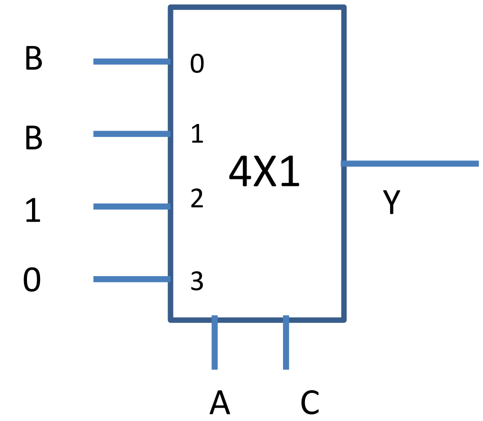


Example:

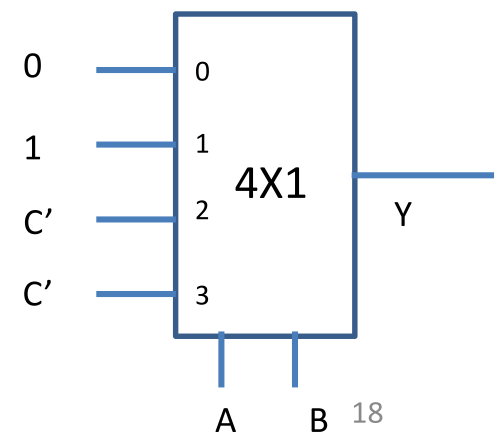
$$F(A,B,C)= \Sigma\ (2, 3, 4, 6)$$

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
B’	0	1	4	5
B	2	3	6	7
	B	B	1	0



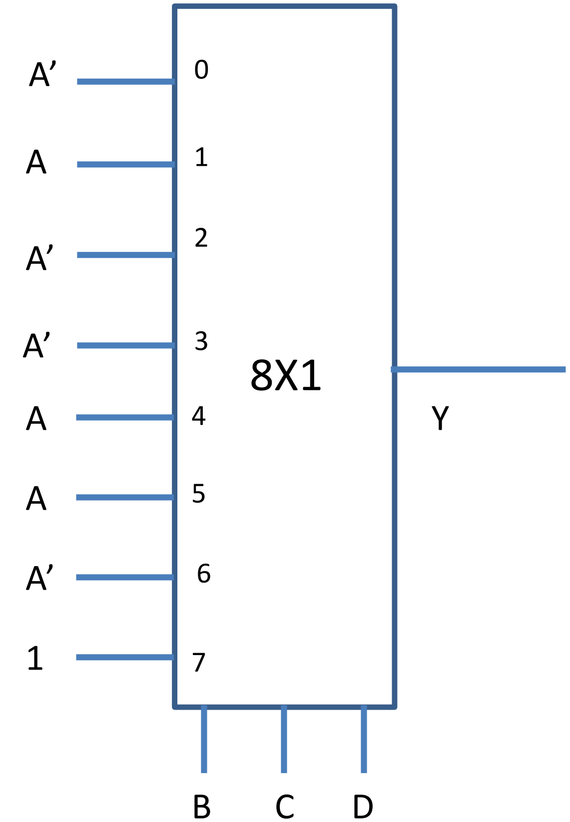
	I_0	I_1	I_2	I_3
C’	0	2	4	6
C	1	3	5	7
	0	1	C’	C’



Example:

$$F(A,B,C,D)= \Sigma\ (0,2,3,6,7,9,12,13,15)$$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A’	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	A’	A	A’	A’	A	A	A’	1

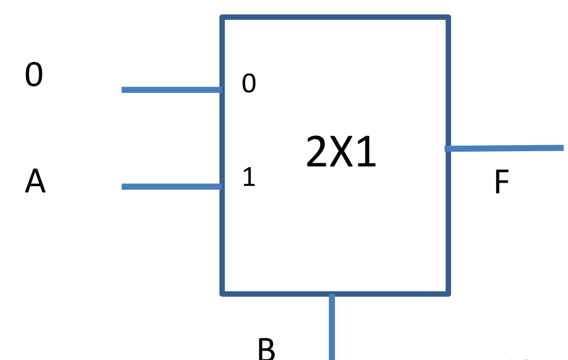


Example: AND gate

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

$$F(A,B)= AB$$

	I_0	I_1
A’	0	1
A	2	3
	0	A



Example:

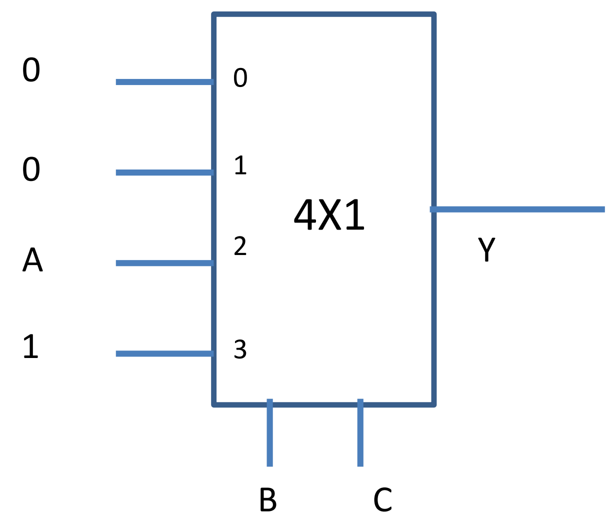
$$F(A,B,C) = AB+BC$$

$$\begin{aligned} F &= AB+BC \\ &= AB(C+C')+(A+A')BC \\ &= ABC+ABC'+ABC+A'BC \\ &= ABC+ABC' +A'BC \end{aligned}$$

$$\quad 7 \qquad\quad 6 \qquad\quad 3$$

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



Summary

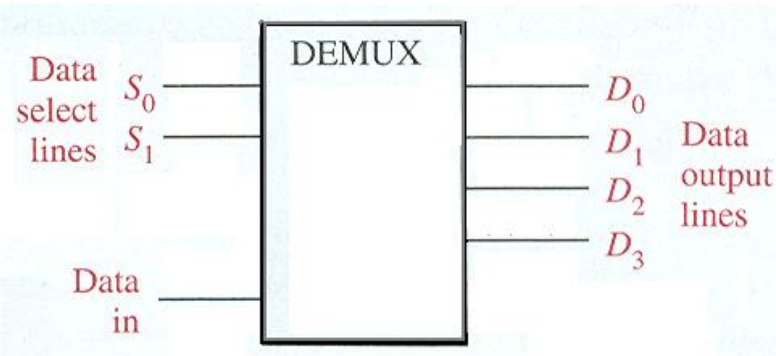
- A 2^n-to-1 multiplexer routes one of 2^n input lines to a single output line.
- Just like decoders,
  - Muxes are common enough to be supplied as stand-alone devices for use in modular designs.
  - Muxes can implement arbitrary functions.
- We saw some variations of the standard multiplexer:
  - Smaller muxes can be combined to produce larger ones.
  - We can add active-low or active-high enable inputs.
- As always, we use truth tables and Boolean algebra to analyze things.

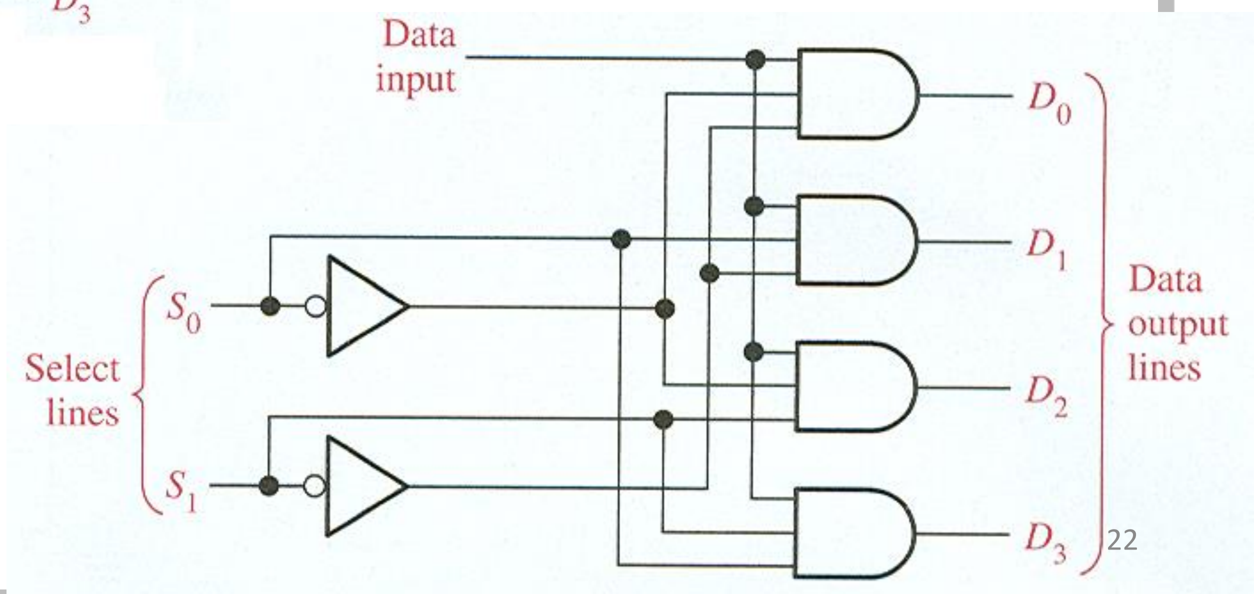
Demultiplexers

2-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





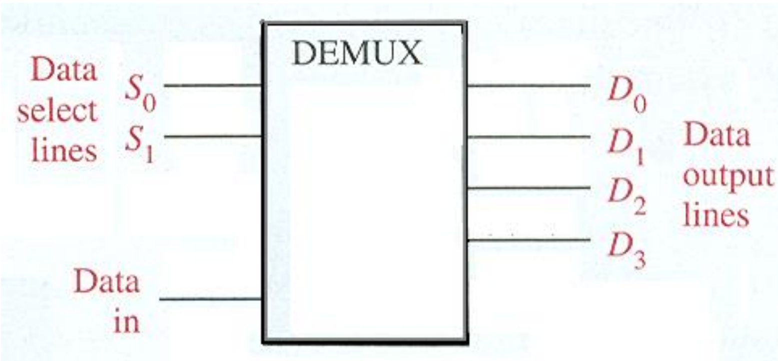
$D_0=S_1'S_0'D_{in}$

$D_1=S_1'S_0D_{in}$

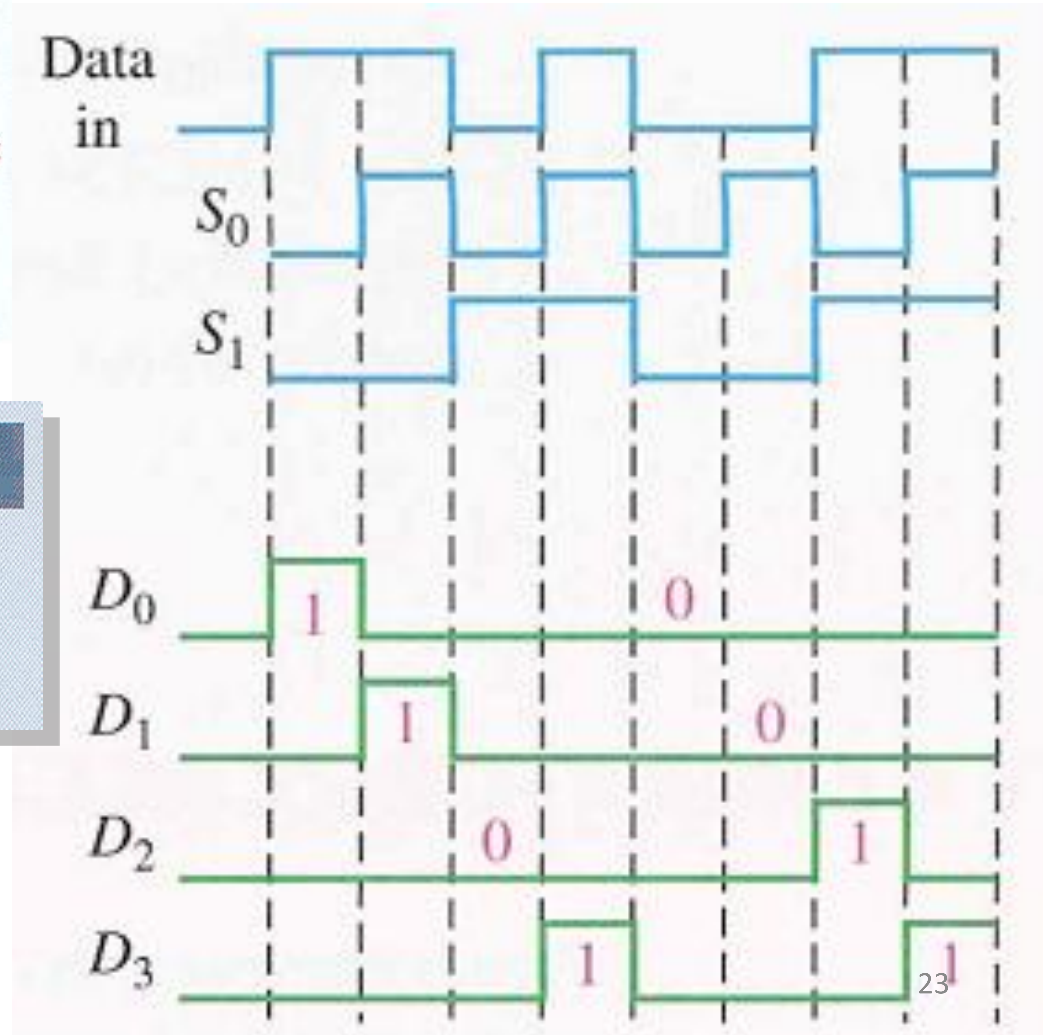
$D_2=S_1S_0'D_{in}$

$D_3=S_1S_0D_{in}$

Timing Diagram

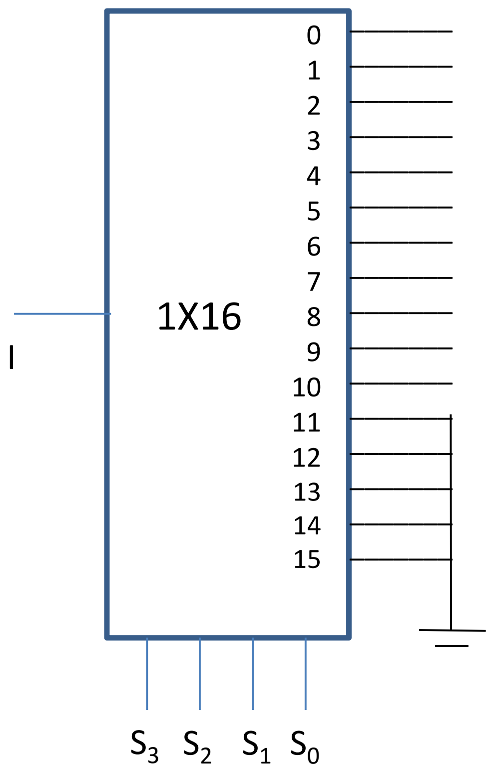


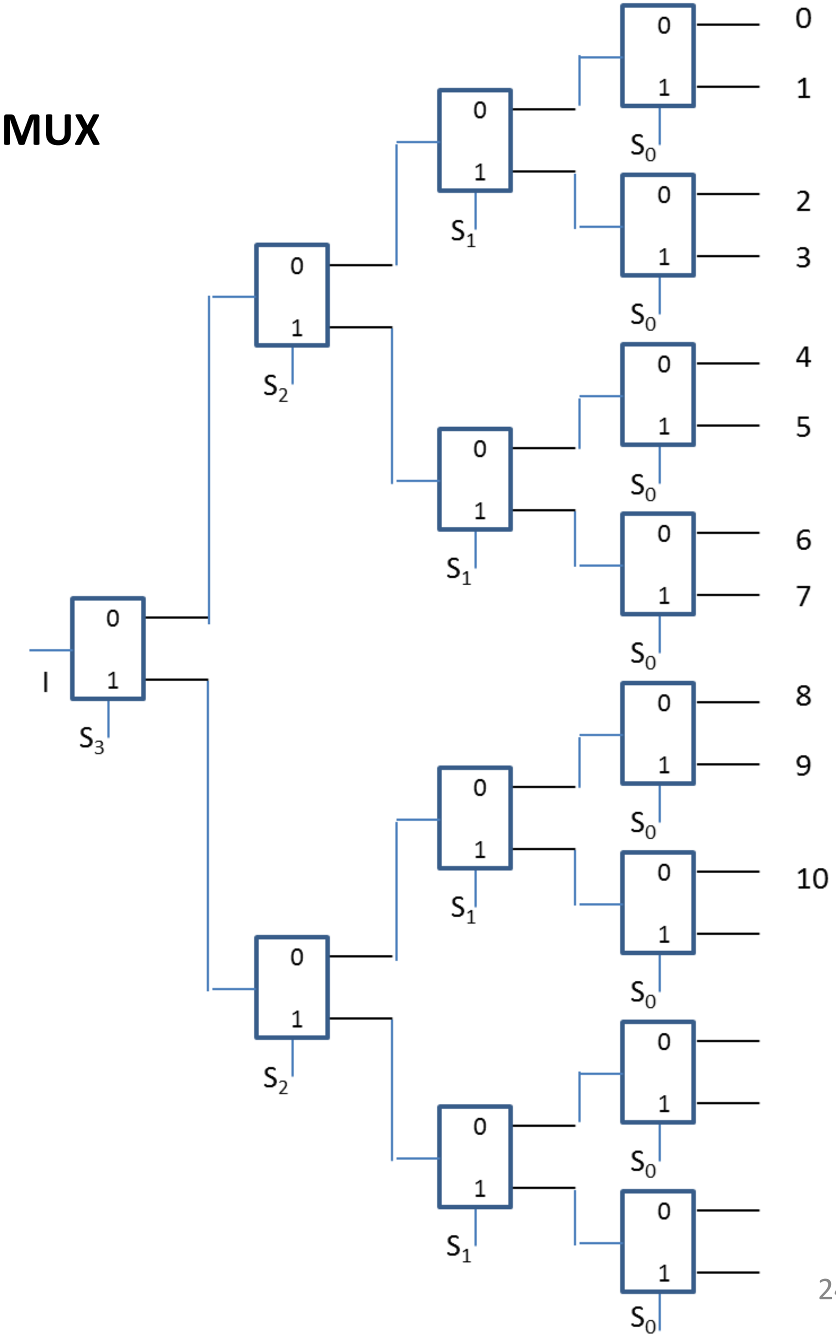
DATA-SELECT INPUTS S_1	S_0	OUTPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



Cascading DEMUX

Designing a 1x11 DEMUX using 1x2 DEMUX





Reference:

Mixed contents from books by Floyd; Mano; Vahid And Howard.

Acknowledgement:

Nafiz Ahmed Chisty

Thanks