Multiplexers/Demultiplexers

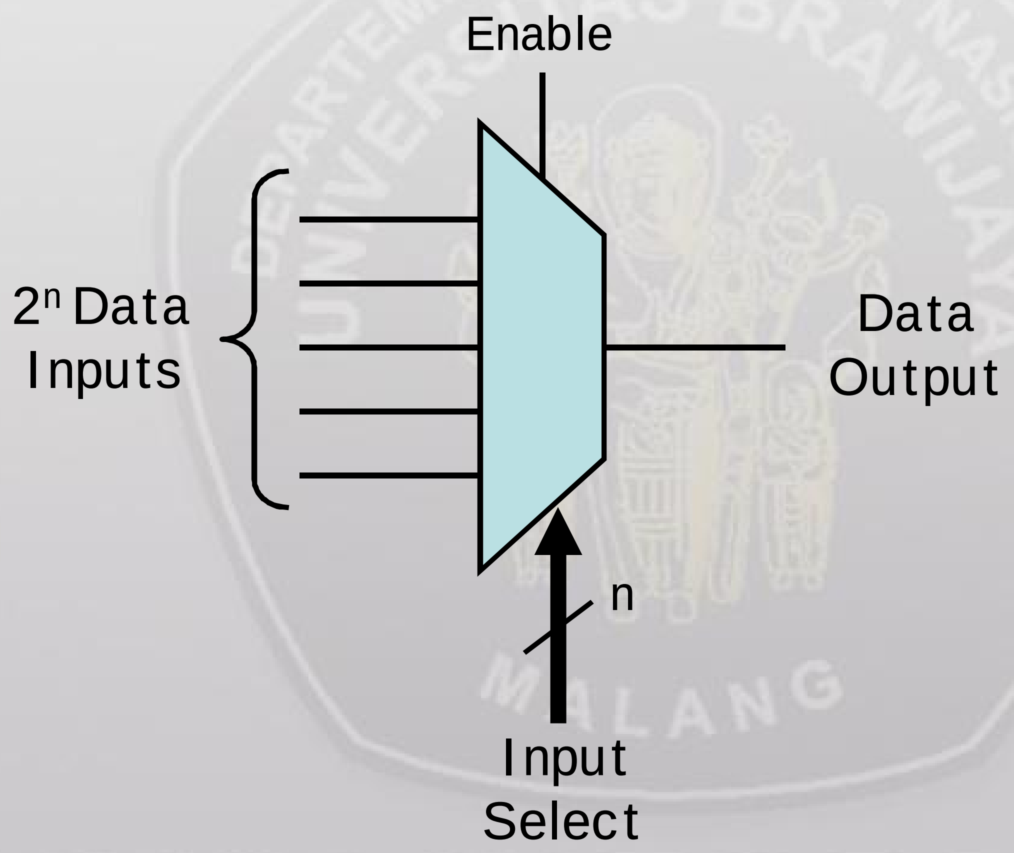
Overview

- Multiplexers (MUXs)
  - Functionality
  - Circuit implementation with MUXs
- Demultiplexers

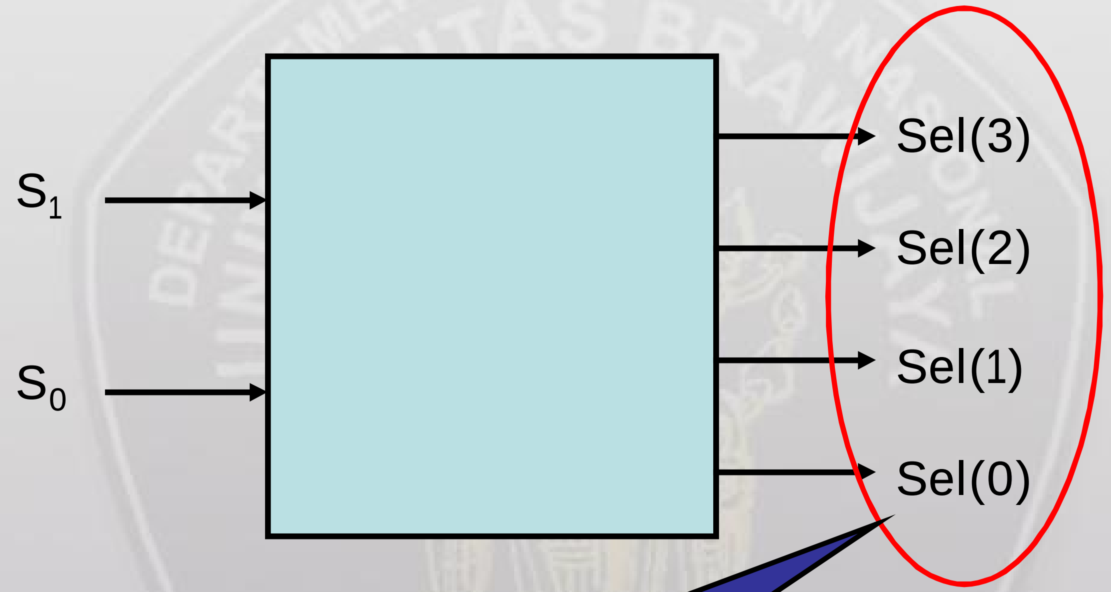
Multiplexer

- “Selects” binary information from one of many input lines and directs it to a single output line.
- Also known as the “selector” circuit,
- Selection is controlled by a particular set of inputs lines whose # depends on the # of the data input lines.
- For a 2^n-to-1 multiplexer, there are 2^n data input lines and *n* selection lines whose bit combination determines which input is selected.

MUX

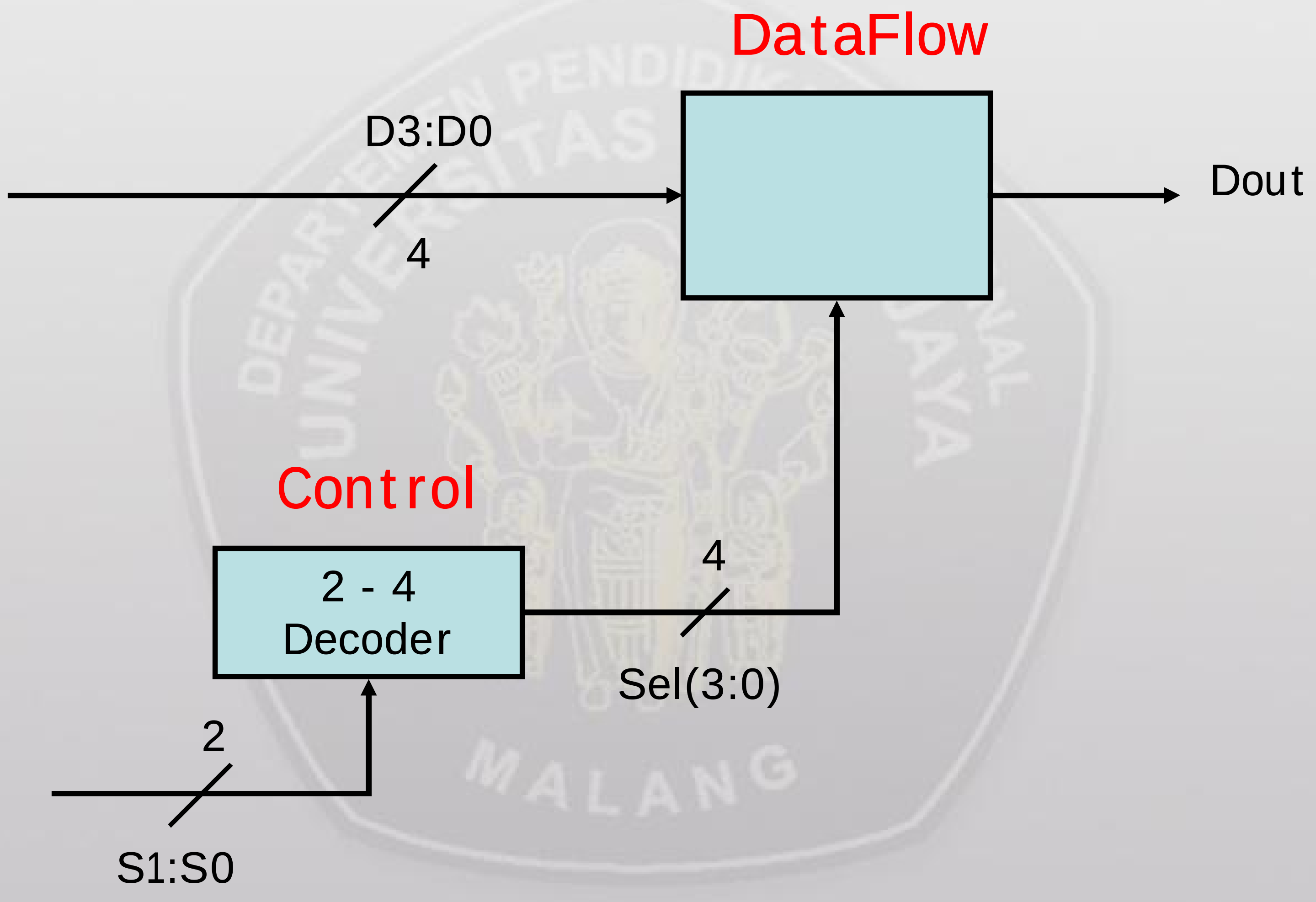


Remember the 2 – 4 Decoder?

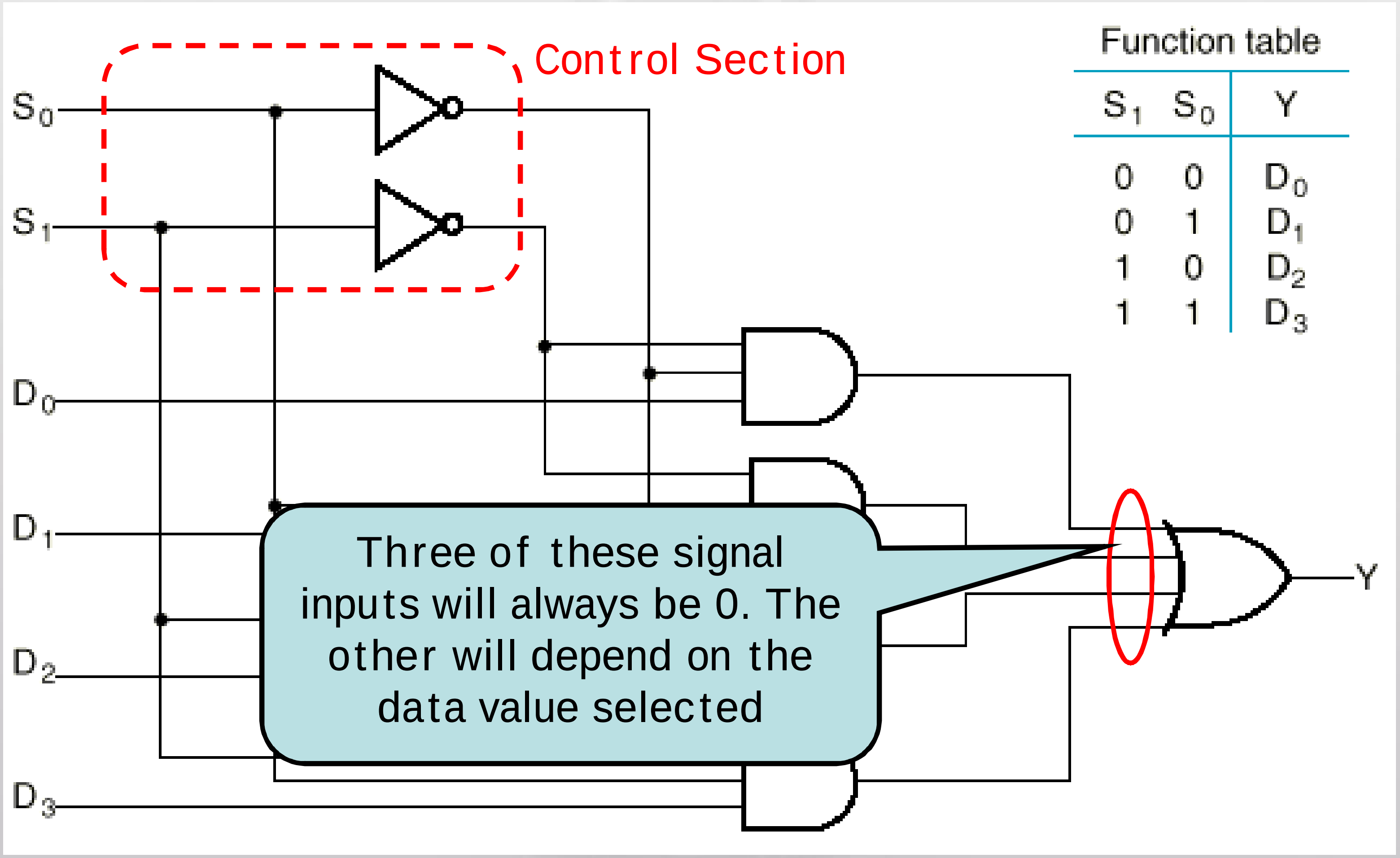


Mutually Exclusive
(Only one O/P asserted at any time

4 to 1 MUX



4-to-1 MUX (Gate level)


Control Section
S_0
S_1
D_0
D_1
D_2
D_3
Y
Three of these signal inputs will always be 0. The other will depend on the data value selected

Function table

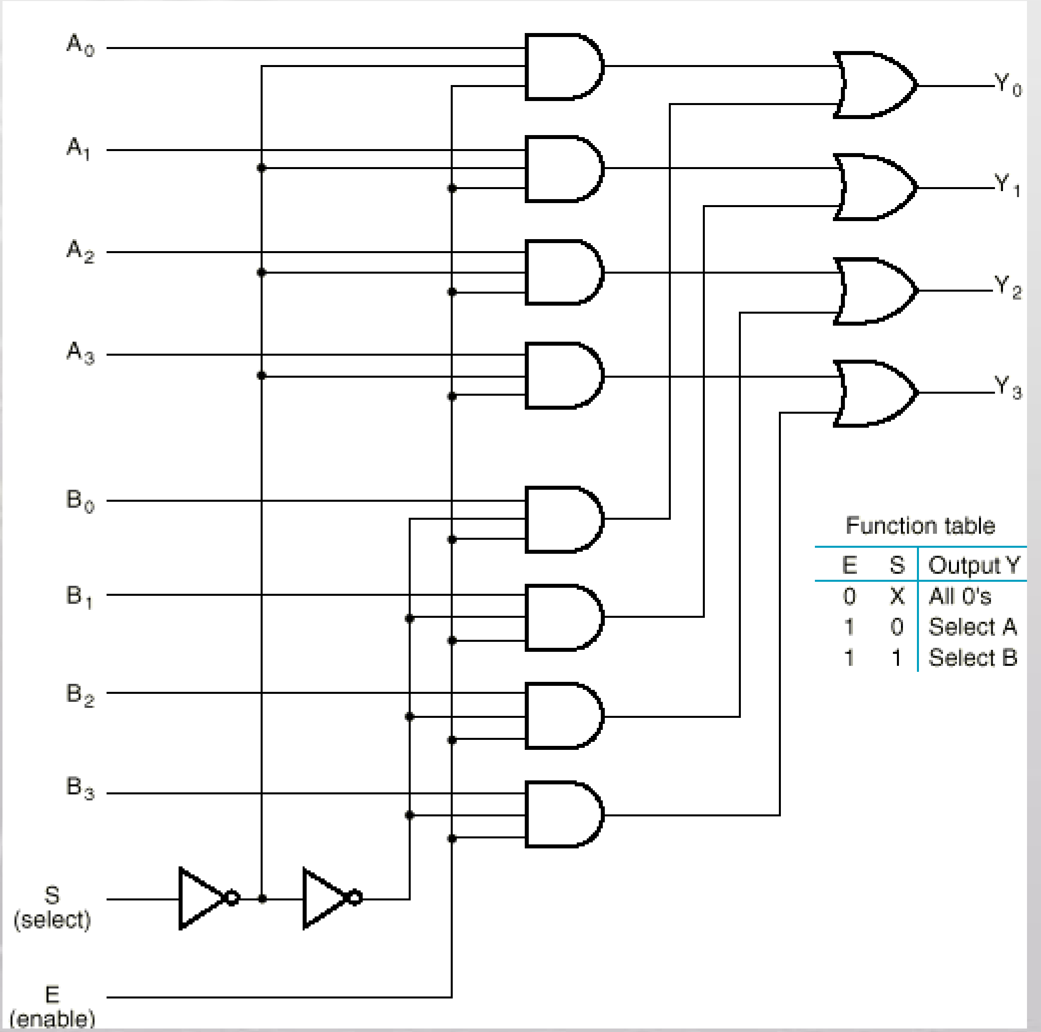
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Multiplexer (cont.)

- Until now, we have examined single-bit data selected by a MUX. What if we want to select m-bit data/words?
  à Combine MUX blocks in parallel with common select and enable signals
- Example: Construct a logic circuit that selects between 2 sets of 4-bit inputs (see next slide for solution).

Example: Quad 2-to-1 MUX

- Uses four 4-to-1 MUXs with common select (S) and enable (E).
- Select line chooses between A_i's and B_i's. The selected four-wire digital signal is sent to the Y_i's
- Enable line turns MUX on and off (E=1 is on).



Function table

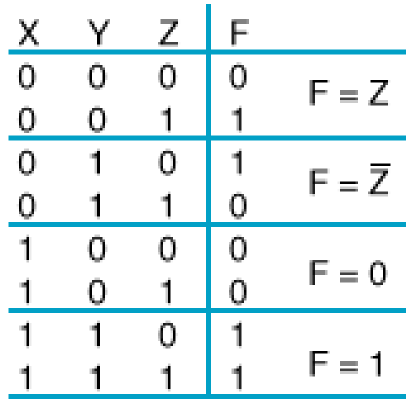
E	S	Output Y
0	X	All 0's
1	0	Select A
1	1	Select B

Implementing Boolean functions with Multiplexers

- Any Boolean function of *n* variables can be implemented using a 2^{n-1}-to-1 multiplexer. A MUX is basically a decoder with outputs ORed together, hence this isn't surprising.
- The SELECT signals generate the minterms of the function.
- The data inputs identify which minterms are to be combined with an OR.

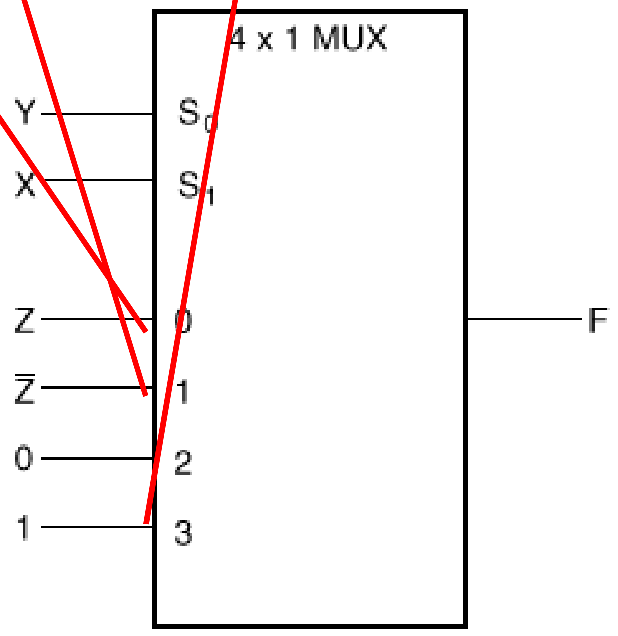
Example

- F(X,Y,Z) = X'Y'Z + X'YZ' + XYZ' + XYZ = Σm(1,2,6,7)
- There are n=3 inputs, thus we need a 2^2-to-1 MUX
- The first n-1 (=2) inputs serve as the selection lines



X	Y	Z	F	
0	0	0	0	F = Z
0	0	1	1	
0	1	0	1	F = $\bar{Z}$
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

Efficient Method for implementing Boolean functions

- For an *n*-variable function (*e.g.*, f(A,B,C,D)):
  - Need a 2^{n-1} line MUX with *n*-1 select lines.
  - Enumerate function as a truth table with consistent ordering of variables (*e.g.*, A,B,C,D)
  - Attach the most significant *n*-1 variables to the *n*-1 select lines (*e.g.*, A,B,C)
  - Examine pairs of adjacent rows (only the least significant variable differs, *e.g.*, D=0 and D=1).
  - Determine whether the function output for the (A,B,C,0) and (A,B,C,1) combination is (0,0), (0,1), (1,0), or (1,1).
  - Attach 0, D, D', or 1 to the data input corresponding to (A,B,C) respectively.

Another Example

- Consider F(A,B,C) = åm(1,3,5,6). We can implement this function using a 4-to-1 MUX as follows.
- The index is ABC. Apply A and B to the S_1 and S_0 selection inputs of the MUX (A is most sig, S_1 is most sig.)
- Enumerate function in a truth table.

MUX Example (cont.)

When A=B=0, F=C

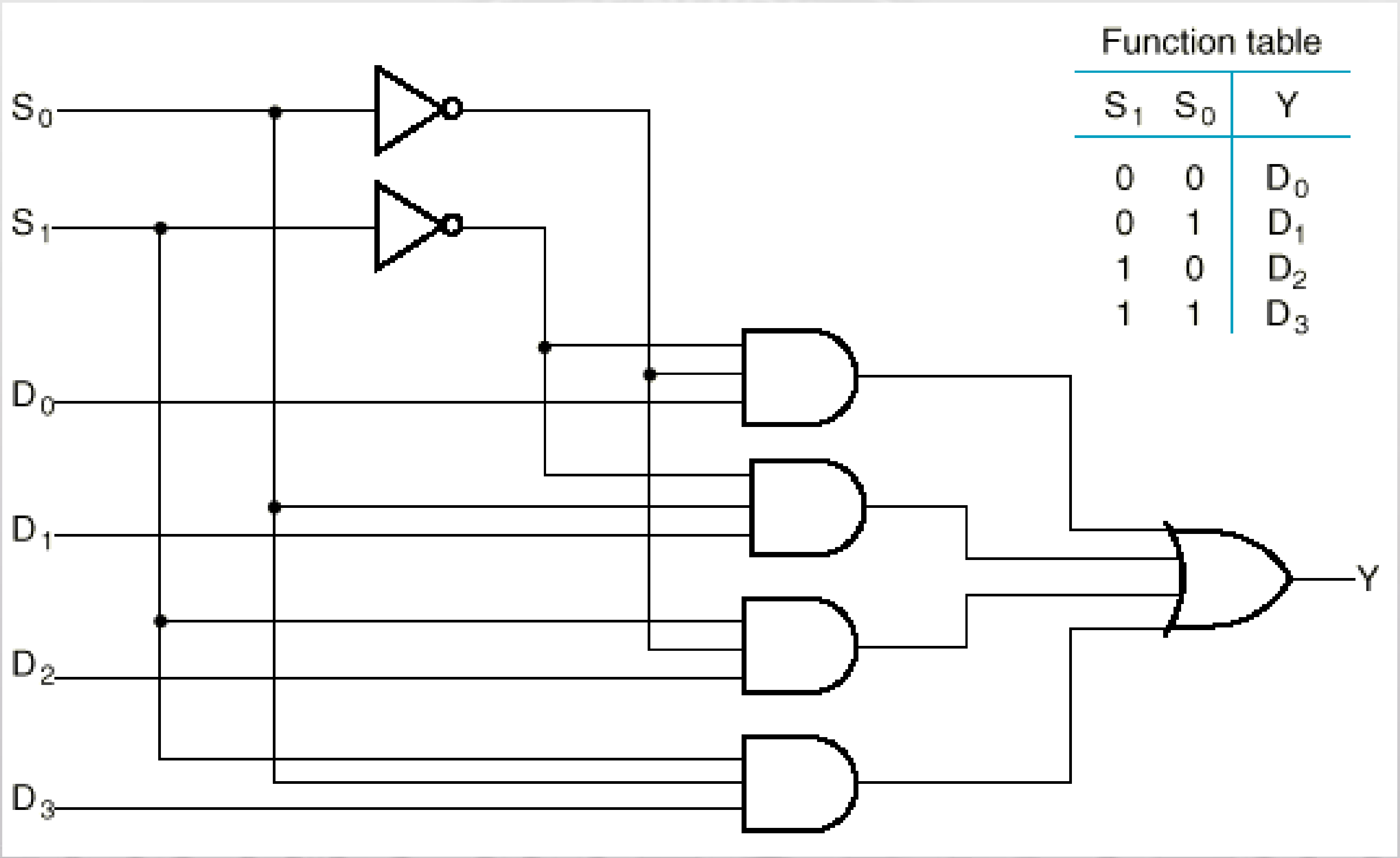
When A=0, B=1, F=C

When A=1, B=0, F=C

When A=B=1, F=C’

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

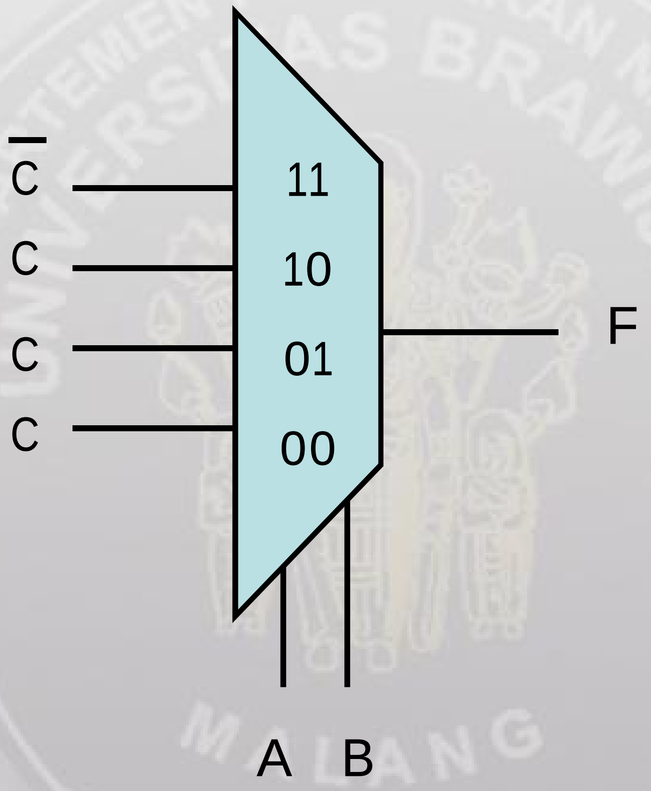
MUX implementation of F(A,B,C) = $\sum m(1,3,5,6)$



Function table

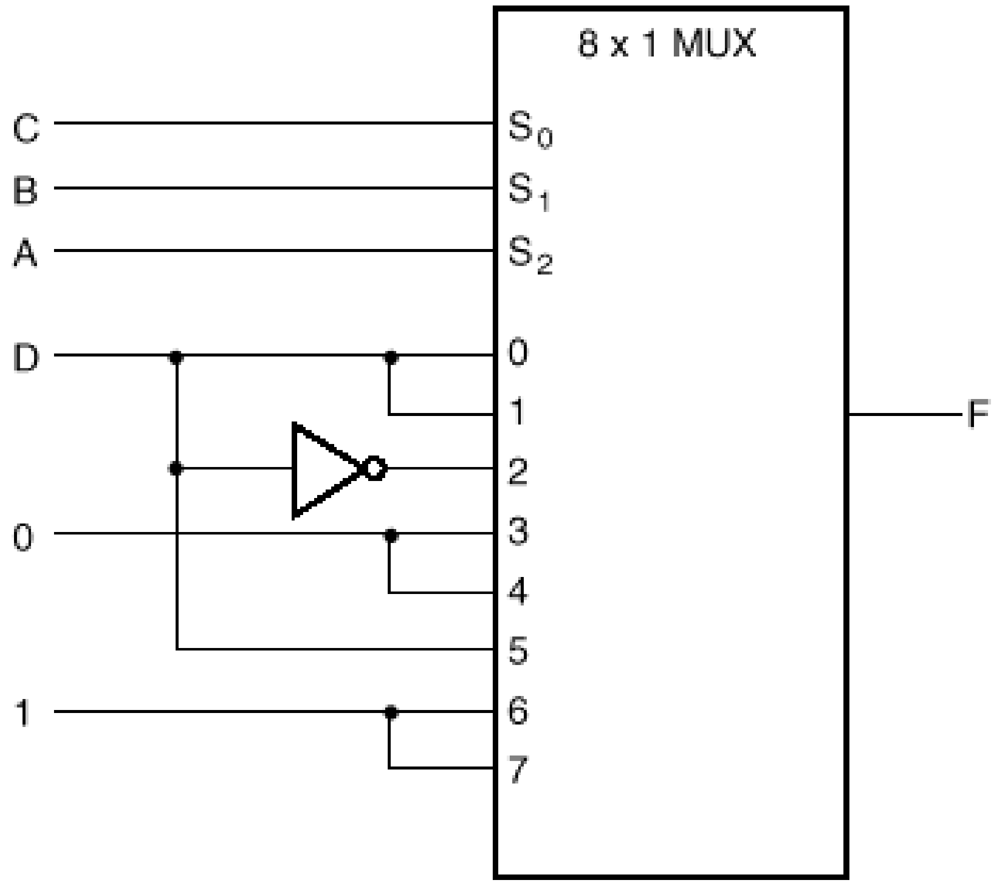
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Or Simply....



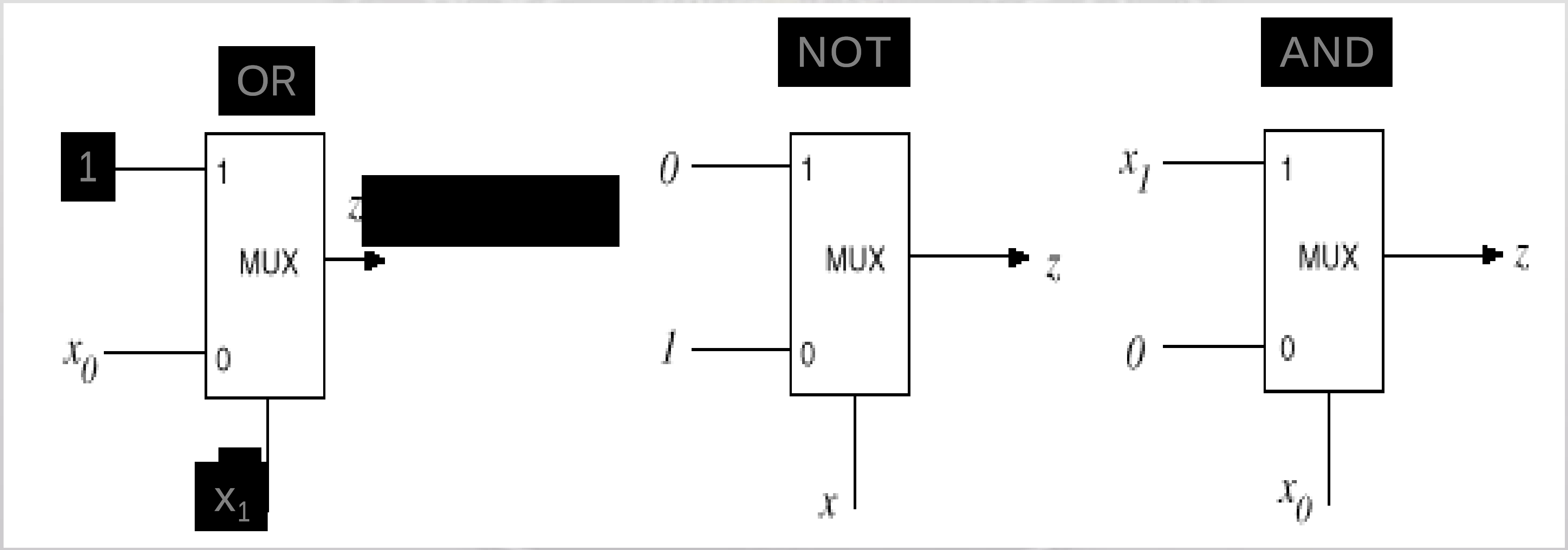
A larger Example

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 = $\overline{D}$
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	



MUX as a Universal Gate

- We can construct OR, AND, and NOT gates using 2-to-1 MUXs. Thus, 2-to-1 MUX is a universal gate.



$z = x_1 + x_1'x_0$

$= x_1x_0' + x_1x_0 + x_1'x_0 = x_1 + x_0$

$z = 0x + 1x' = x'$

$z = x_1x_0 + 0x_0' = x_1x_0$

Demultiplexers (DMUX)

- Performs the inverse of a multiplexing operation:
  - Receives data from a single line
  - Transmit it to one of the *2^n* possible output lines
  - Selection of a specific output is controlled by the *n* select lines
  - Demultiplexers are basically decoders! For example, a 2-to-4 DMUX is a 2-to-4 decoder with enable input.