Demultiplexers
Lecture 5

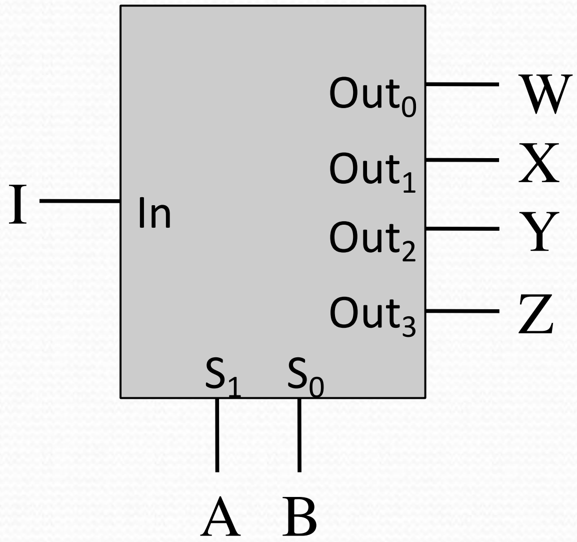
Dronacharya Group of Institutions

Demultiplexers

Demultiplexers

- A demultiplexer has
  - N control inputs
  - 1 data input
  - 2^N outputs
- A demultiplexer routes (or connects) the data input to the selected output.
  - The value of the control inputs determines the output that is selected.
- A demultiplexer performs the opposite function of a multiplexer.

Demultiplexers



W = A'.B'.I

X = A.B'.I

Y = A'.B.I

Z = A.B.I

A	B	W	X	Y	Z
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

EXERCISES

Designing logic circuits using <u>multiplexers</u> and <u>decoders</u>

Using an *n*-input Multiplexer

- Use an *n*-input multiplexer to realize a logic circuit for a function with *n* minterms.
  - $m = 2^n$, where m = # of variables in the function
- Each minterm of the function can be mapped to an input of the multiplexer.
- For each row in the truth table, for the function, where the output is 1, set the corresponding input of the multiplexer to 1.
  - That is, for each minterm in the minterm expansion of the function, set the corresponding input of the multiplexer to 1.
- Set the remaining inputs of the multiplexer to 0.

Using an *n*-input Mux

Example:

Using an 8-to-1 multiplexer, design a logic circuit to realize the following Boolean function

$$F(A,B,C) = \Sigma m(2, 3, 5, 6, 7)$$

Using an <u>*n*-input</u> Mux

Example:

Using an 8-to-1 multiplexer, design a logic circuit to realize the following Boolean function

$F(A,B,C) = \Sigma m(1, 2, 4)$

Using an <u>*(n / 2)*-input</u> Multiplexer

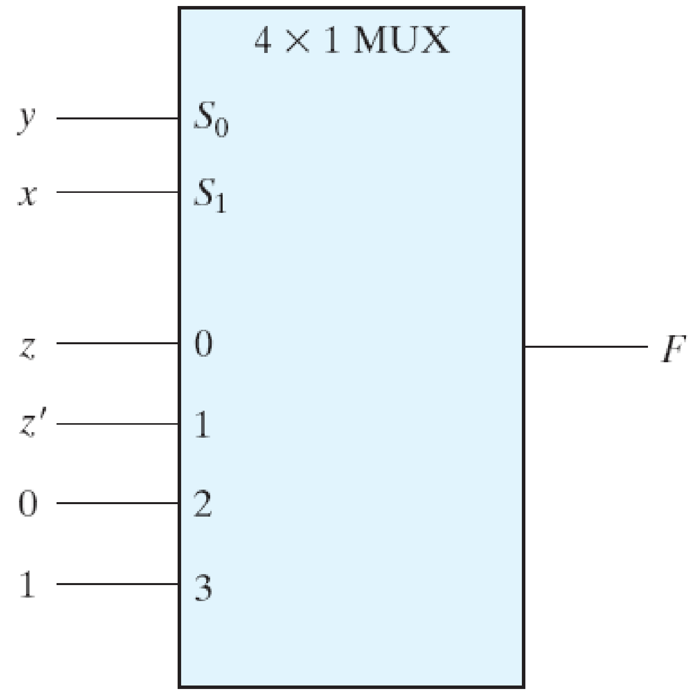
- Use an *(n / 2)*-input multiplexer to realize a logic circuit for a function with *n* minterms.
  - $m = 2^n$, where m = # of variables in the function
- Group the rows of the truth table, for the function, into *(n / 2)* pairs of rows.
  - Each pair of rows represents a product term of *(m – 1)* variables.
  - Each pair of rows can be mapped to a multiplexer input.
- Determine the logical function of each pair of rows in terms of the m^{th} variable.
  - If the m^{th} variable, for example, is x, then the possible values are x, x', 0, and 1.

Using an *(n / 2)*-input Mux

Example: $F(x,y,z) = \Sigma m(1, 2, 6, 7)$

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

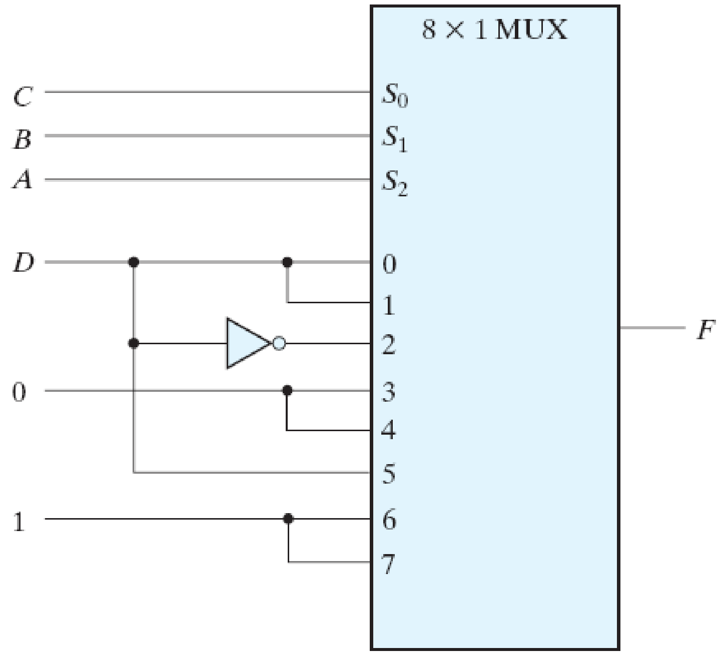


(b) Multiplexer implementation

Using an *(n / 2)*-input Mux

Example: F(A,B,C,D) = Σm(1,3,4,11,12–15)

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	$F = D'$
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	



Using an *(n / 4)*-input Mux

The design of a logic circuit using an *(n / 2)*-input multiplexer can be easily extended to the use of an *(n / 4)*-input multiplexer.

Using an *n*-output Decoder

- Use an *n*-output decoder to realize a logic circuit for a function with *n* minterms.
- Each minterm of the function can be mapped to an output of the decoder.
- For each row in the truth table, for the function, where the output is 1, sum (or "OR") the corresponding outputs of the decoder.
  - That is, for each minterm in the minterm expansion of the function, OR the corresponding outputs of the decoder.
- Leave remaining outputs of the decoder unconnected.

Using an *n*-output Decoder

Example:

Using a 3-to-8 decoder, design a logic circuit to realize the following Boolean function

$$F(A,B,C) = \Sigma m(2, 3, 5, 6, 7)$$

Using an *n*-output Decoder

Example:

Using two 2-to-4 decoders, design a logic circuit to realize the following Boolean function

$F(A,B,C) = \Sigma m(0, 1, 4, 6, 7)$