Lecture 7

Combinational Logic (continue)

Outline

- Other Arithmetic Circuits
- Decoders and Encoders
- Multiplexers

Decimal Adder

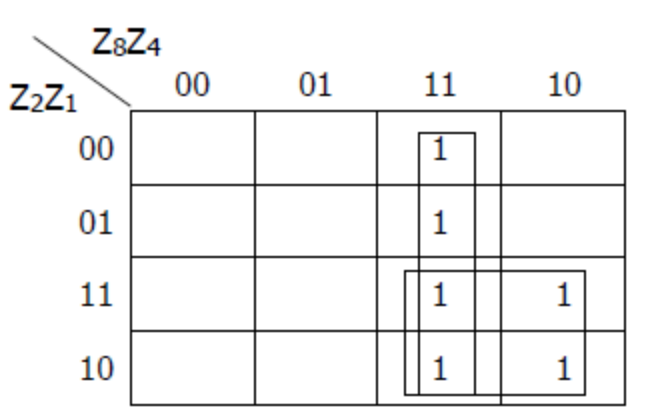
- A decimal adder requires a minimum of 9 inputs and 5 outputs
  - 1 digit requires 4-bit
  - Input: 2 digits + 1-bit carry
  - Output: 1 digit + 1-bit carry
- BCD adder
  - Perform the addition of two decimal digits in BCD, together with an input carry from a previous stage
  - The output sum cannot be greater than 19 (9+9+1)

Derivation of BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
		.....					.....			.....
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

BCD Adder

- When the binary sum is equal to or less than 1001b
  - BCD Sum = Binary Sum
  - C = 0
- When the binary sum is greater than 1001b
  - BCD Sum = Binary Sum + 0110b
  - C = 1

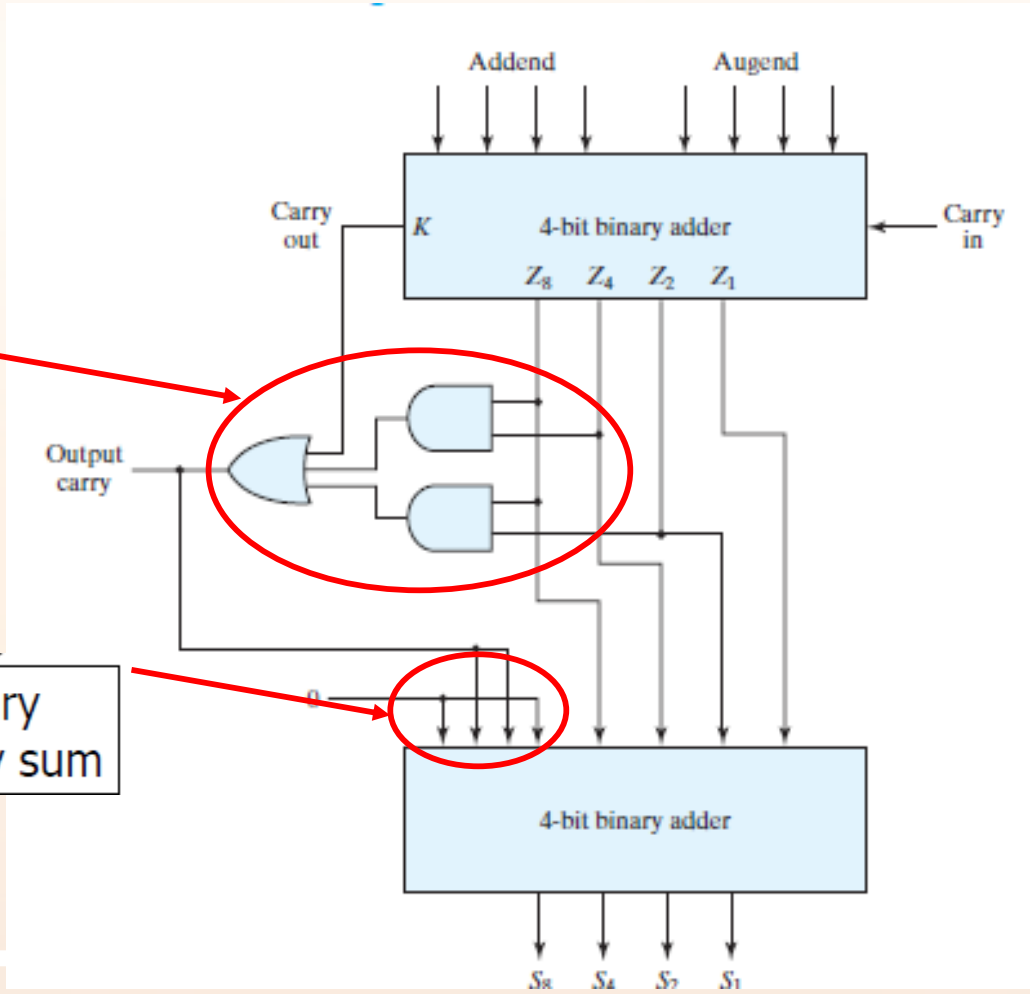


$$C = K + Z_8Z_4 + Z_8Z_2$$

Block Diagram of a BCD Adder

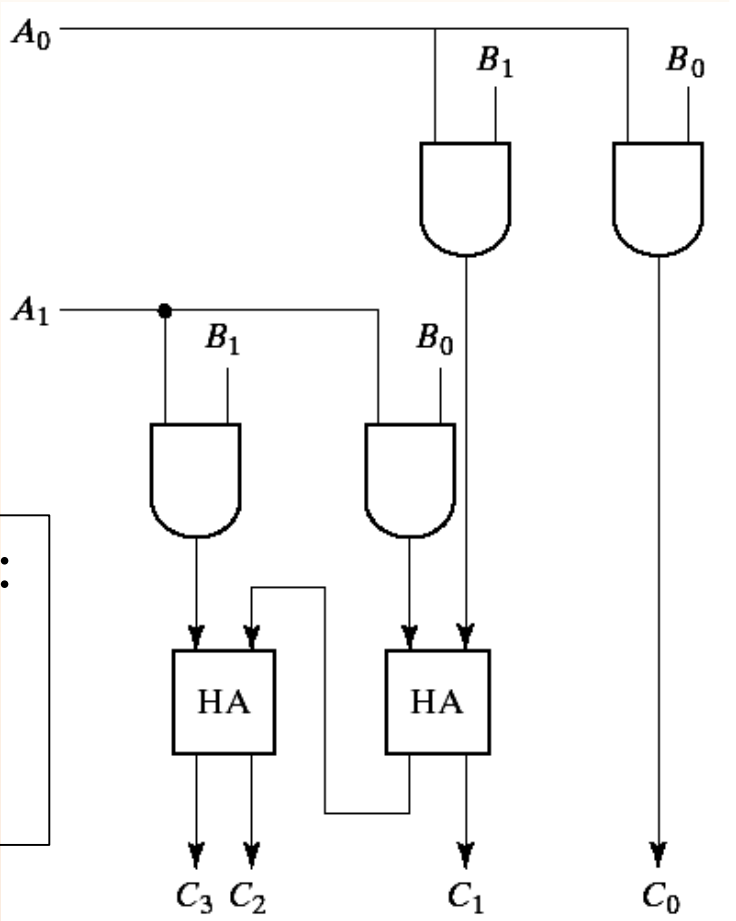
$C = K + Z_8Z_4 + Z_8Z_2$

If C =1 , it is necessary to add 0110 to binary sum



Binary Multiplier

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0



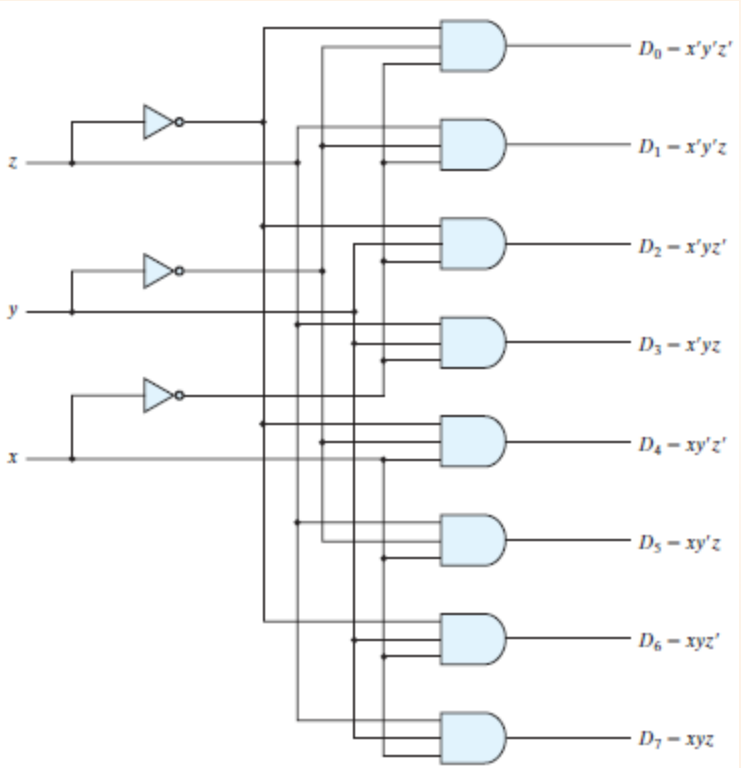
For a J*K bits multiplier , we need:
(J * K) AND gates
(J – 1) K-bit adders
to produce a product of J+K bits

Decoder

- A circuit that coverts binary information from n input lines to a maximum of 2^n unique output lines
  - May have fewer than 2^n outputs
- A n-to-m-line decoder ($m \leq 2^n$):
  - Generate the m minterns of n input variables
- For each possible input combination, there is only one output that is equal to 1
  - The output whose value is equal to 1 represents the minterm equivalent of the binary number presently available in the input lines

3-to-8-Line Decoder

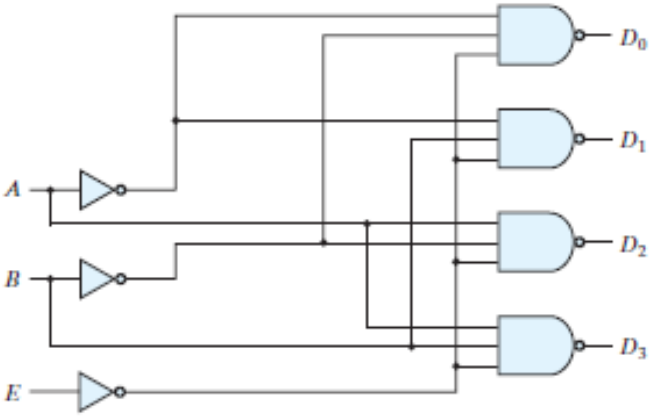
- The 3 inputs are decoded into 8 outputs
- Each represent one of the minterms of the inputs variables



Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

2-to-4-Line Decoder with Enable

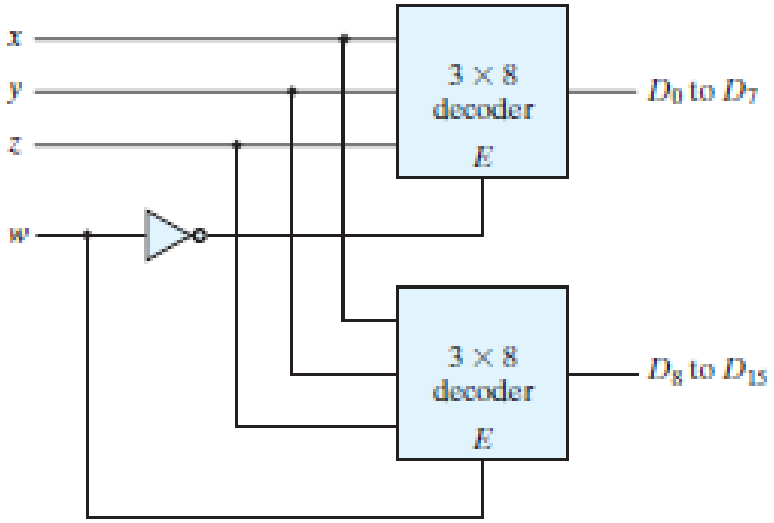
- Some decoders are constructed with NAND gates
  - Generate minterms in their complement form
- An **enable input can be added to control the operation**
  - E=1: disabled
  - None of the outputs are equal to 0



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

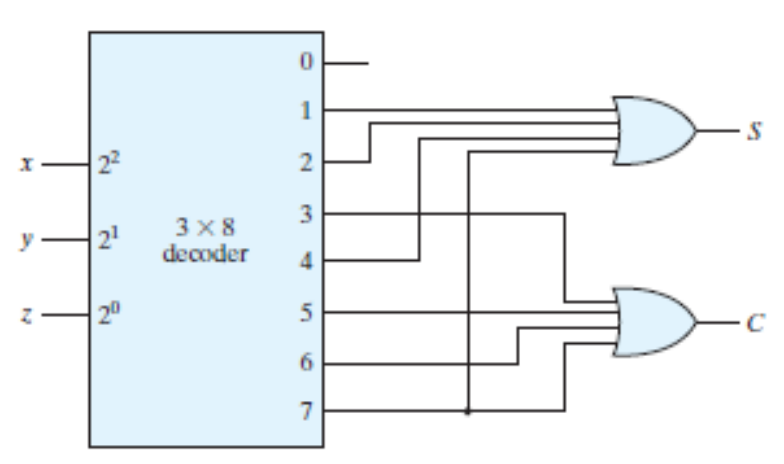
Construct Larger Decoders

- Decoders with enable inputs can be connected together to form a larger decoder
- The enable input is used as the most significant bit of the selection signal
  - w=0: the top decoder is enabled
  - w=1: the bottom one is enabled
- In general, enable inputs are a convenient feature for standard components to expand their numbers of inputs and outputs



Combinational Logic Implementation

- A decoder provides the 2^n minterms of n input variables
  - Can be used to form any combinational circuits with extra OR gates (sum of minterms)
- A function having a list of k minterms can be expressed in its complemented form F' with $2^n - k$ minterms
  - If $k > 2^n/2$, F' will have fewer minterms (fewer OR gates)
  - NOR gates are used instead for implementing F'



$$S(x,y,z) = \sum(1,2,4,7)$$
$$C(x,y,z) = \sum(3,5,6,7)$$

Encoder

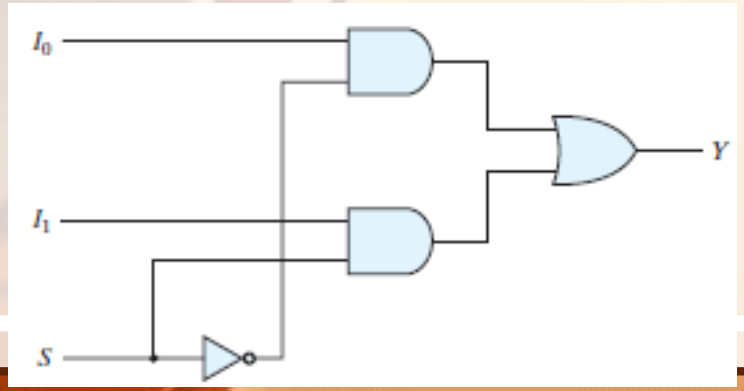
- A circuit that performs the inverse operation of a decoder
  - Have 2^n (or fewer) input lines and n output lines
  - The output lines generate the binary code of the input positions
- Only one input can be active at any given time
- An extra output may be required to distinguish the cases that D0 = 1 and all inputs are 0

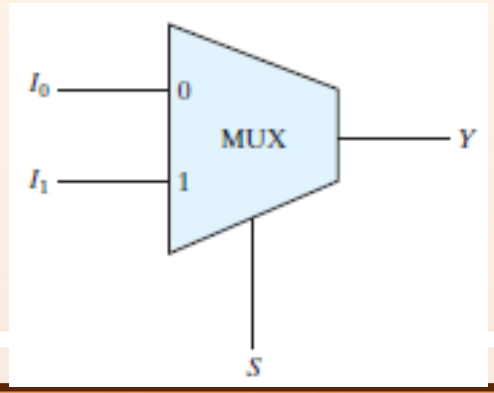
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

Multiplexer

- circuit that selects binary information from one of many input lines and directs it to a single output lines
  - Have 2^n input lines and n selection lines
  - Act like an electronic switch (also called a **data selector)**
- For the following 2-to-1-line multiplexer:
  - S=0 $\rightarrow$ Y = I_0 ; S=1 $\rightarrow$ Y = I_1



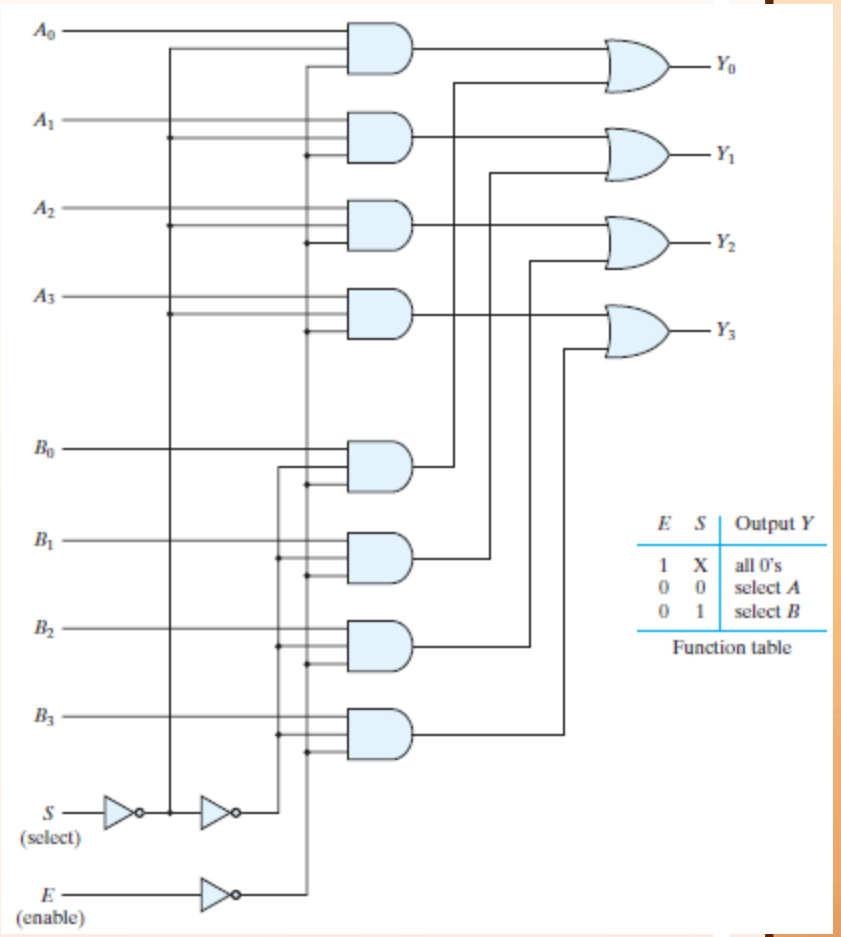


VHDL code for a 2-to-1 multiplexer.

```
LIBRARY ieee ;
USE ieee.std logic 1164.all ;
ENTITY mux2to1 IS
PORT ( w0, w1, s : IN STD LOGIC ;
                   f : OUT STD LOGIC ) ;
END mux2to1 ;
ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
WITH s SELECT
              f <w0 WHEN ’0’,
                  w1 WHEN OTHERS ;
 END Behavior ;
```

Quadruple 2-to-1-Line Multiplexer

- Multiplexers can be combined with **common selection inputs to provide multiple-bit** selection logic
- Quadruple 2-to-1-line multiplexer:
- Four 2-to-1-line multiplexers
  - Each capable of selecting one bit of the 2 4-bit inputs
  - E: enable input
  - E=1: disable the circuit (all outputs are 0)



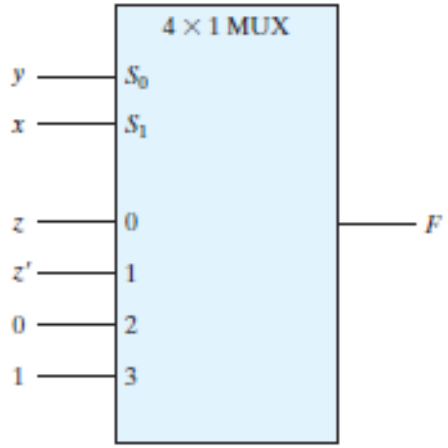
Boolean Function Implementation

- A multiplexer is essentially a decoder with an external OR gate
  - Can be used to implement Boolean functions without extra logic
- To implement a Boolean function of n variables:
  - Use a multiplexer with n - 1 selection inputs
  - The first n – 1 variables are connected to the selection inputs
  - The remaining variable is used for the data inputs

$$F(x,y,z) = \sum(1,2,6,7)$$

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

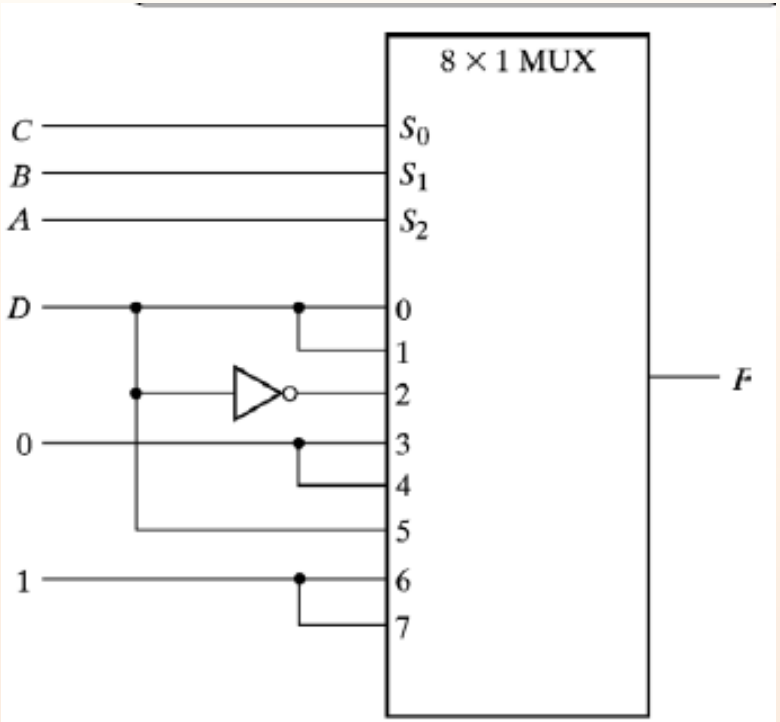


(b) Multiplexer implementation

Implementing a 4-Input Function

F(A,B,C,D) =
$\sum(1,3,4,11,12,13,14,15)$

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	



Three-State Gate

- A circuit that exhibits three states
  - logic 1, logic 0, and **high-impedance (z)**
- The high-impedance state acts like an open circuit (disconnected)
- The most commonly used three-state gate is the buffer gate
  - C=0 → disabled (high-impedance) ; C=1 → enabled (pass)
  - Can be used at the output of a function without altering the internal implementation