ENGIN 112

Intro to Electrical and Computer Engineering

Lecture 15
Magnitude Comparators and Multiplexers

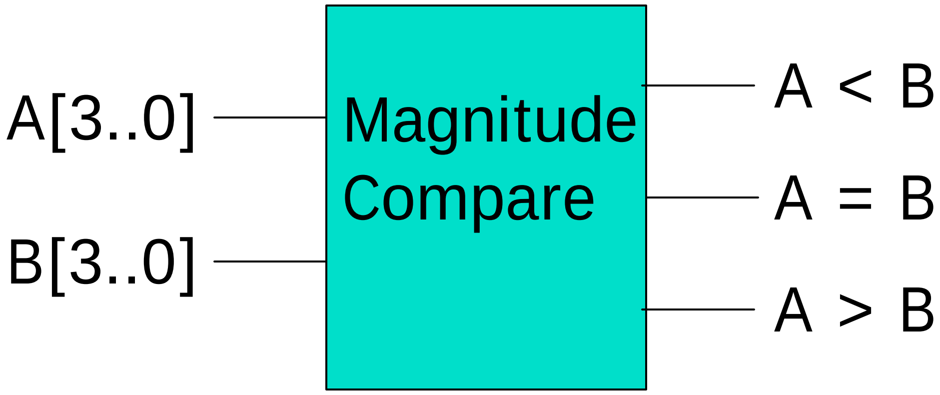
ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF MASSACHUSETTS AMHERST

Overview

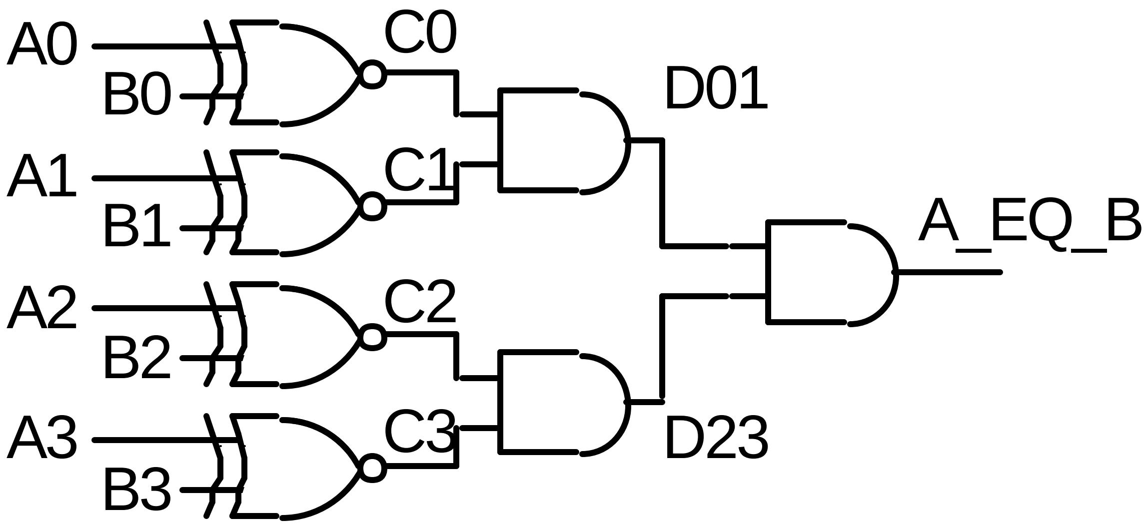
- Discussion of two digital building blocks
- Magnitude comparators
  - Compare two multi-bit binary numbers
  - Create a single bit comparator
  - Use repetitive pattern
- Multiplexers
  - Select one out of several bits
  - Some inputs used for selection
  - Also can be used to implement logic

Magnitude Comparator

- **The comparison of two numbers**
  - **outputs: A>B, A=B, A<B**
- **Design Approaches**
  - **the truth table**
    - **2^{2n} entries - too cumbersome for large n**
  - **use inherent regularity of the problem**
    - **reduce design efforts**
    - **reduce human errors**



Magnitude Comparator

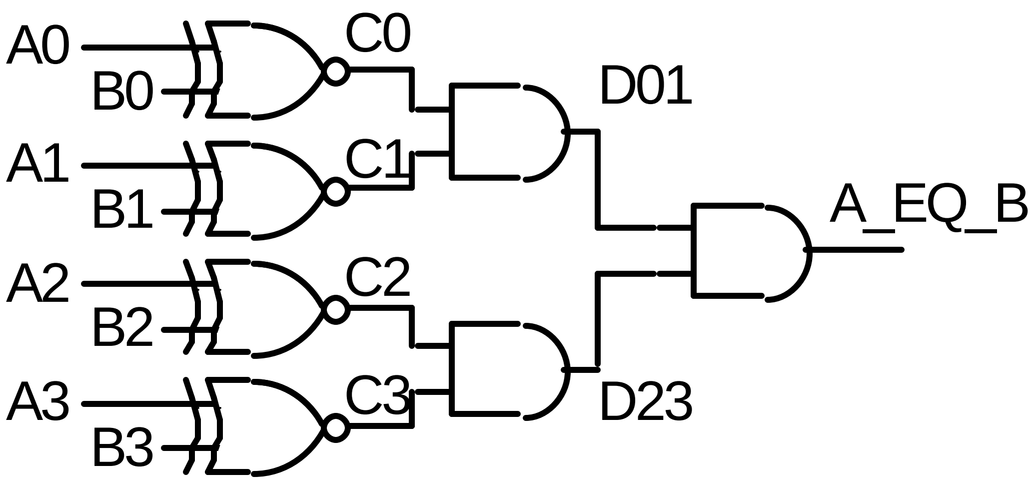


How can we find A > B?

How many rows would a truth table have?

$$2^8 = 256$$

Magnitude Comparator



Find A > B

If A = 1001 and
B = 0111
is A > B?
Why?

Because A3 > B3
i.e. A3 . B3' = 1

Therefore, one term in the logic equation for A > B is A3 . B3'

Magnitude Comparator

If A = 1010 and
B = 1001
is A > B?
Why?

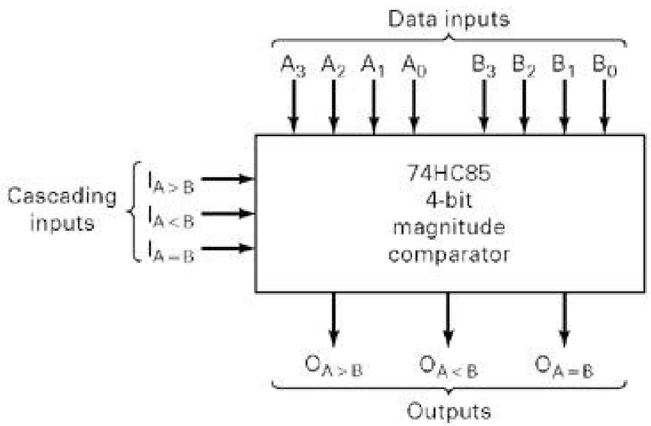
A > B = A3 . B3'
+ C3 . A2 . B2'
+ .....

Because A3 = B3 and
A2 = B2 and
A1 > B1
i.e. C3 = 1 and C2 = 1 and
A1 . B1' = 1

Therefore, the next term in the logic equation for A > B is
C3 . C2 . A1 . B1'

Magnitude Comparison

- **Algorithm -> logic**
  - $A = A_3A_2A_1A_0$; $B = B_3B_2B_1B_0$
  - A=B if $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ and $A_1=B_1$
- **Test each bit:**
  - equality: $x_i= A_iB_i+A_i'B_i'$
  - $(A=B) = x_3x_2x_1x_0$
- **More difficult to test less than/greater than**
  - $(A>B) = A_3B_3'+x_3A_2B_2'+x_3x_2A_1B_1'+x_3x_2x_1 A_0B_0'$
  - $(A<B) = A_3'B_3+x_3A_2'B_2+x_3x_2A_1'B_1+x_3x_2x_1 A_0'B_0$
  - Start comparisons from high-order bits
- **Implementation**
  - $x_i = (A_iB_i'+A_i'B_i)'$



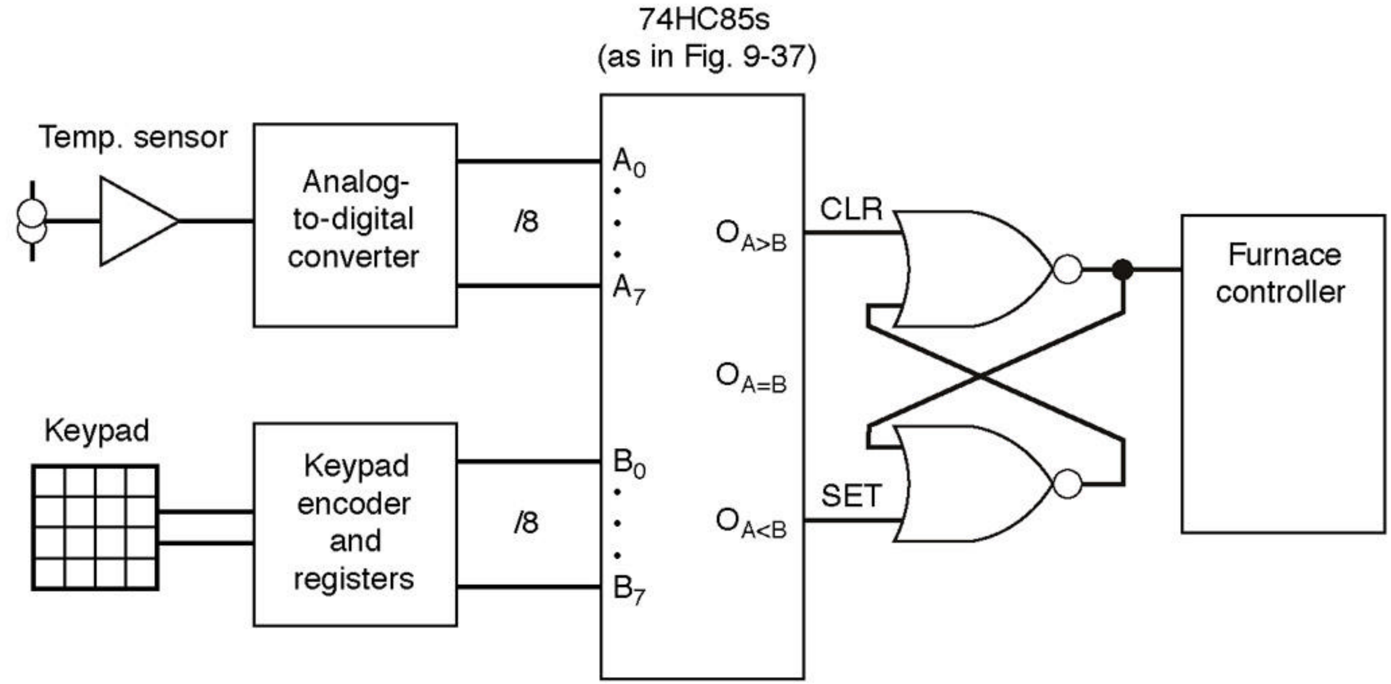
TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

Magnitude Comparator

- **Real-world application**
  - **Thermostat controller**



Multiplexers

- Select an input value with one or more select bits
- Use for transmitting data
- Allows for conditional transfer of data
- Sometimes called a mux

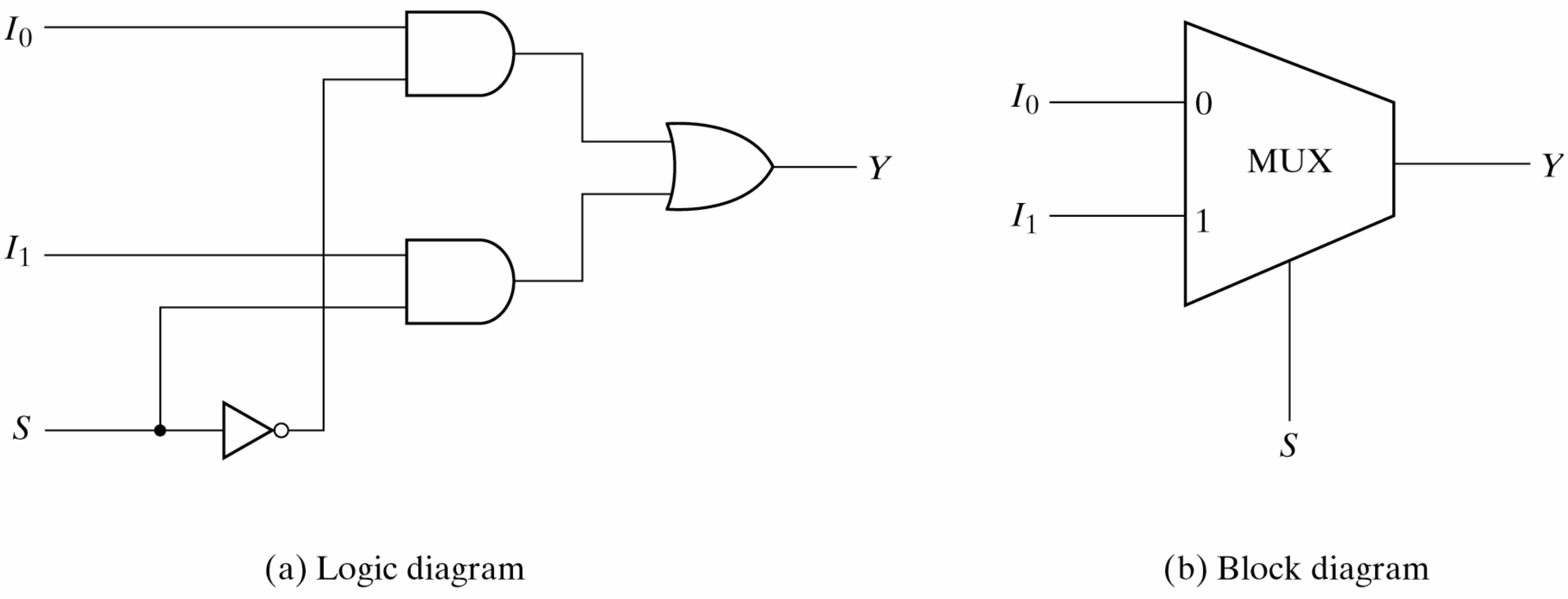
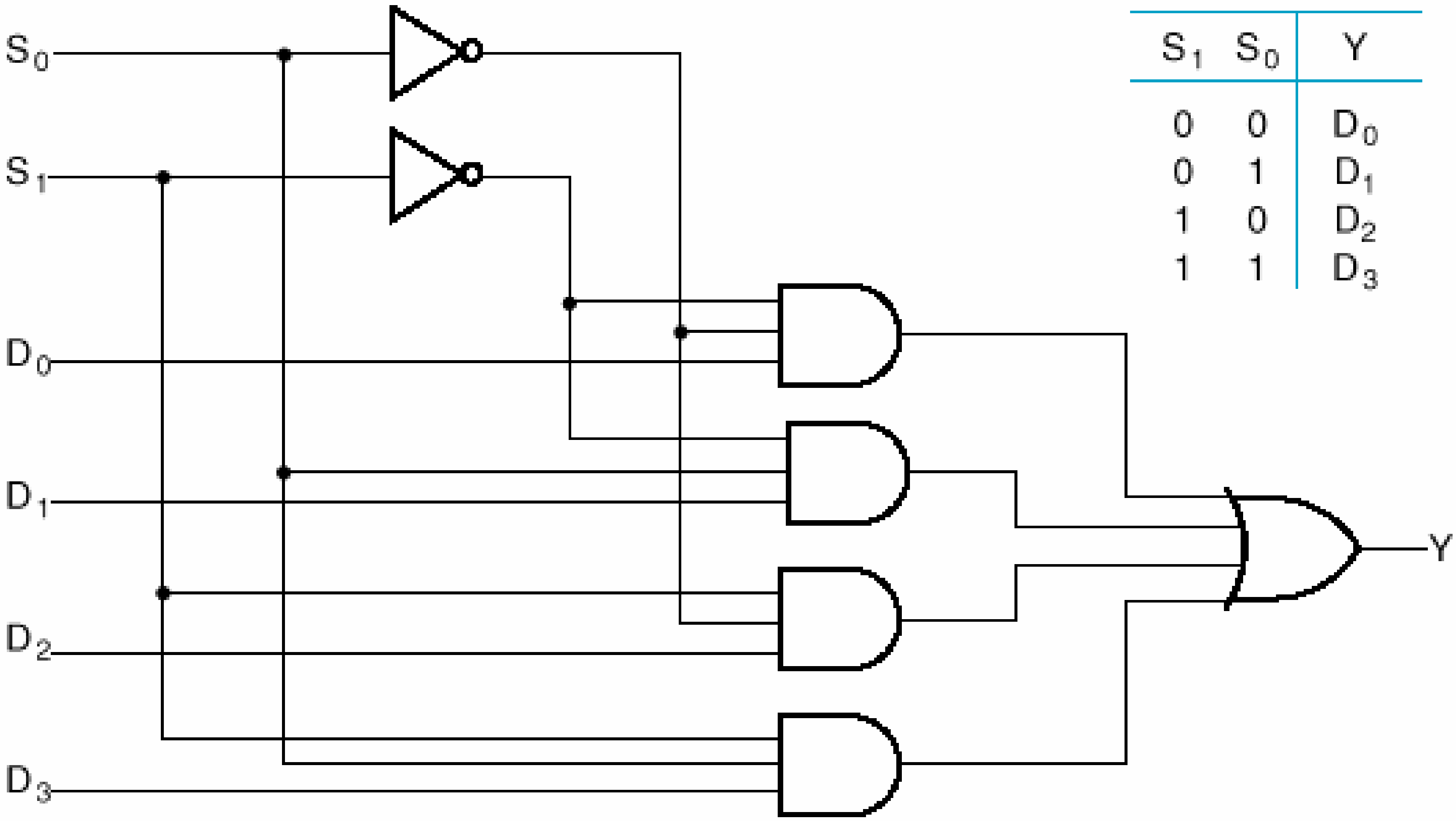


Fig. 4-24 2-to-1-Line Multiplexer

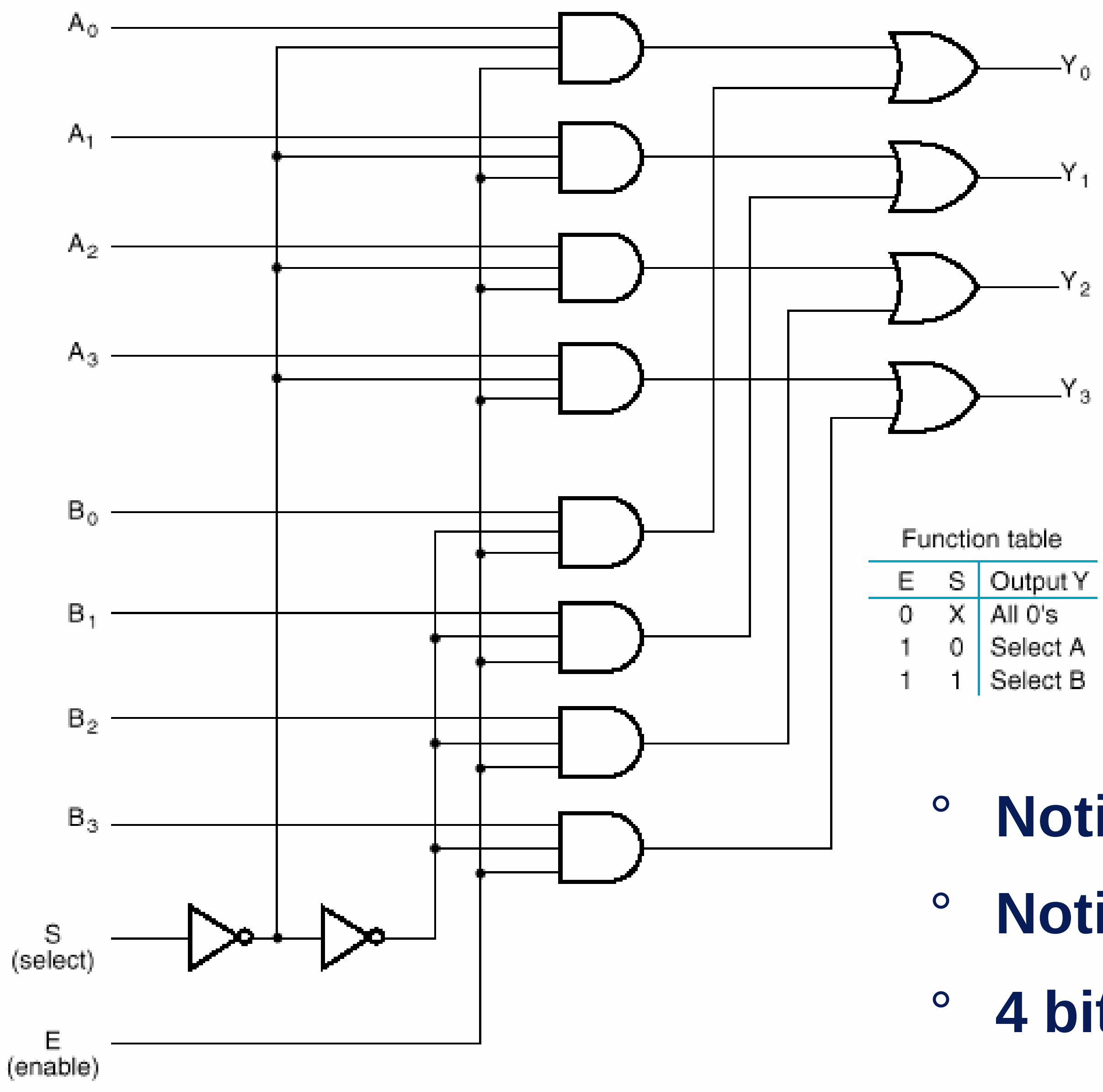
4– to– 1- Line Multiplexer

Function table

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



Quadruple 2–to–1-Line Multiplexer



Function table

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

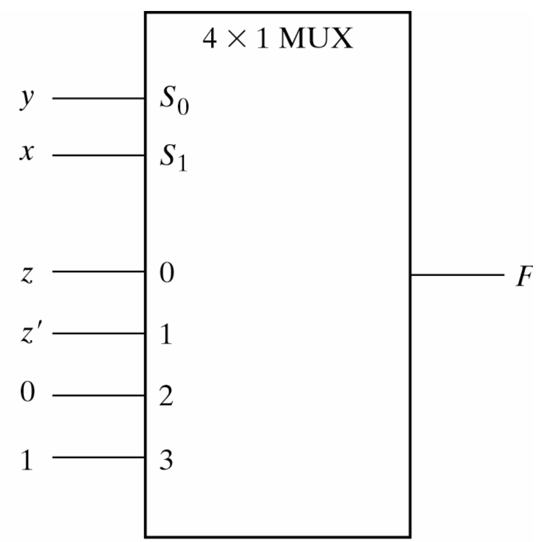
- Notice enable bit
- Notice select bit
- 4 bit inputs

Multiplexer as combinational modules

- **Connect input variables to select inputs of multiplexer *(n-1 for n variables)***
- **Set data inputs to multiplexer equal to values of function for corresponding assignment of select variables**
- **Using a variable at data inputs reduces size of the multiplexer**

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

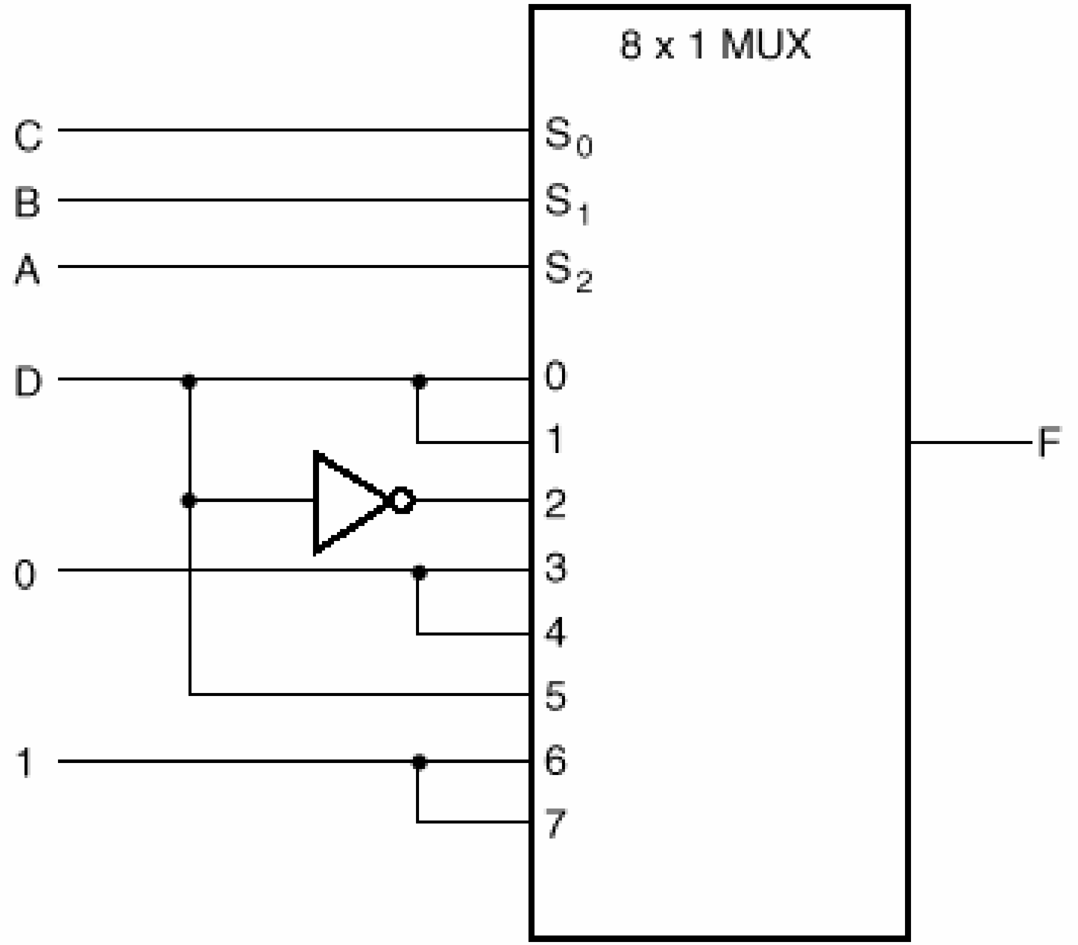


(b) Multiplexer implementation

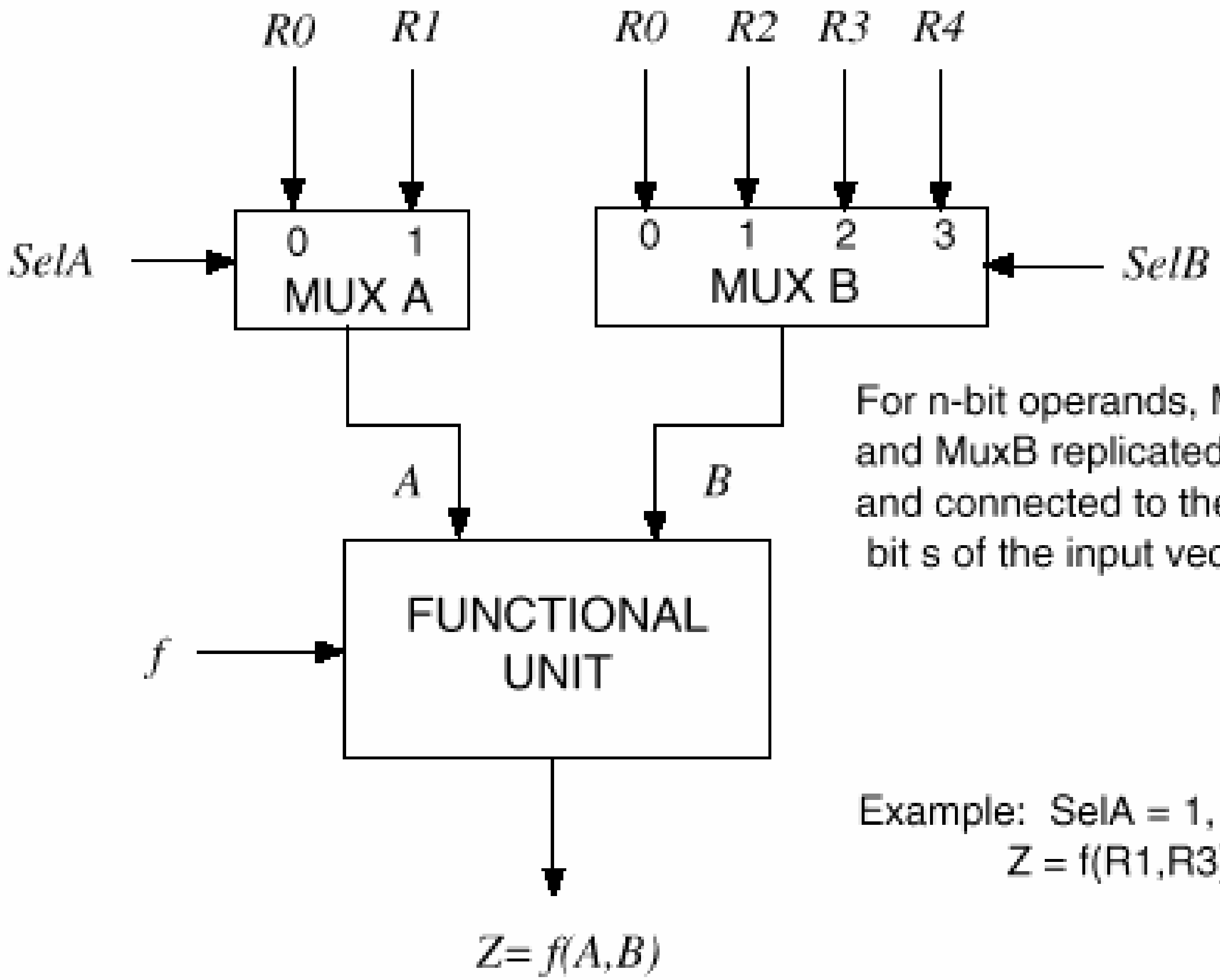
Fig. 4-27 Implementing a Boolean Function with a Multiplexer

Implementing a Four- Input Function with a Multiplexer

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	



Typical multiplexer uses



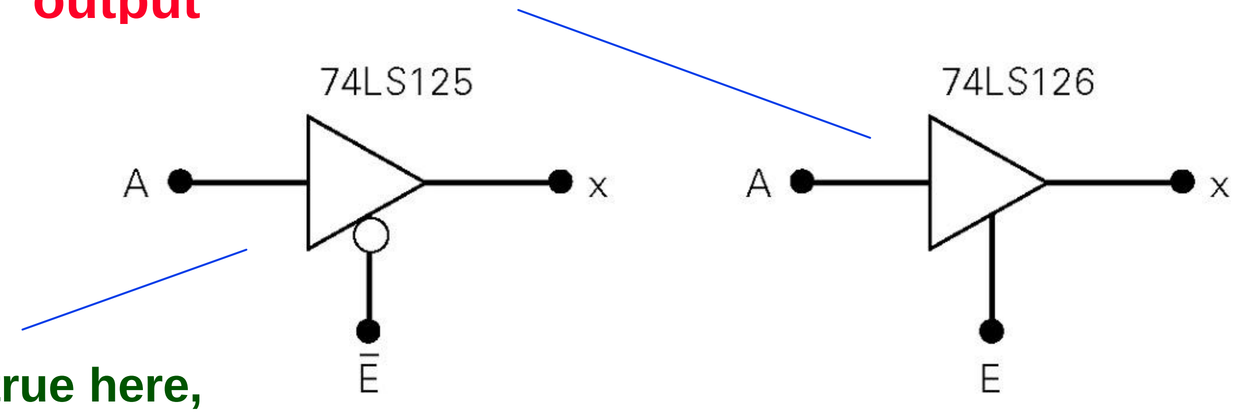
For n-bit operands, Mux A and MuxB replicated n times and connected to the corresponding bit s of the input vectors.

Example: SelA = 1, SelB = 2
Z = f(R1,R3)

Figure 9.21: Multiplexer example of use.

Three-state gates

- **A multiplexer can be constructed with three-state gates**
- **Output state: 0, 1, and high-impedance (open ckts)**
- **If the select input (E) is 0, the three-state gate has no output**



Opposite true here,

No output if $\overline{E}$ is 1

$\overline{E}$	x
0	A
1	Hi-Z

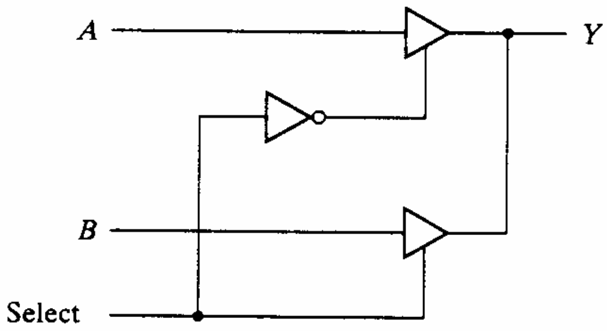
(a)

E	x
0	Hi-Z
1	A

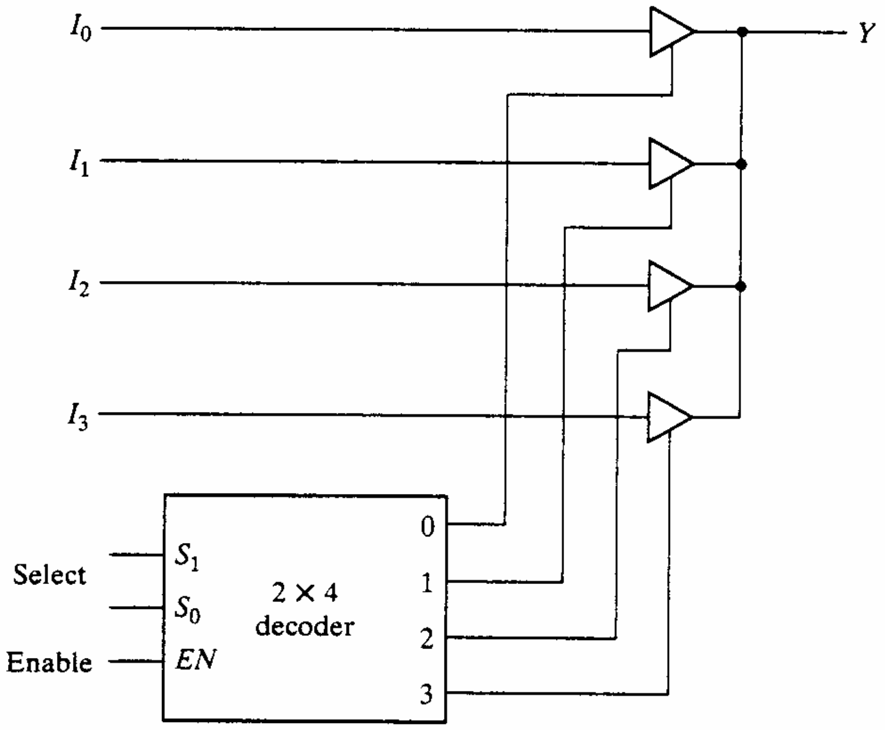
(b)

Three-state gates

- **A multiplexer can be constructed with three-state gates**
- **Output state: 0, 1, and high-impedance (open ckts)**
- **If the select input is low, the three-state gate has no output**



(a) 2-to-1- line mux



(b) 4 - to - 1 line mux

Summary

- Magnitude comparators allow for data comparison
  - Can be built using and-or gates
- Greater/less than requires more hardware than equality
- Multiplexers are fundamental digital components
  - Can be used for logic
  - Useful for datapaths
  - Scalable
- Tristate buffers have three types of outputs
  - 0, 1, high-impedence (Z)
  - Useful for datapaths