



# Chapter 1

# MSI LOGIC CIRCUITS

Faculty of Computer Science and Engineering  
Department of Computer Engineering



Nguyen Quang Huy  
[huynguyen@cse.hcmut.edu.vn](mailto:huynguyen@cse.hcmut.edu.vn)

# LOGIC DESIGN 2

MSI logic circuits

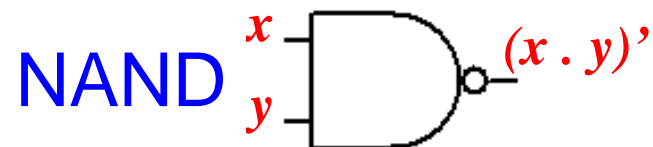
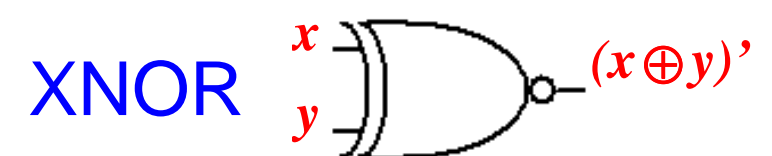
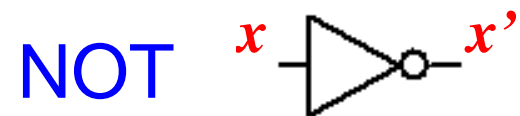
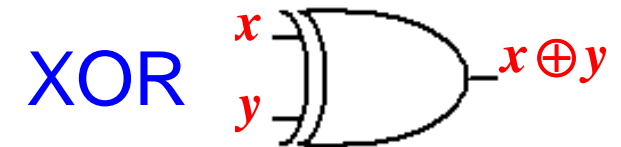
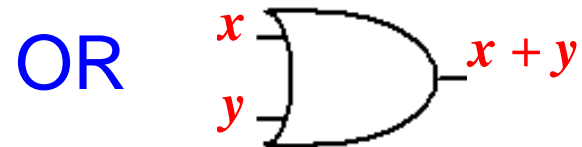
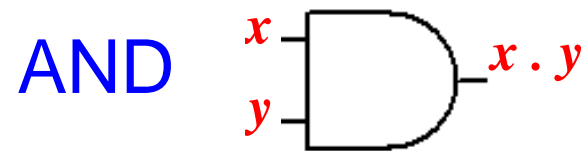
Memory

ADC / DAC

Logic family

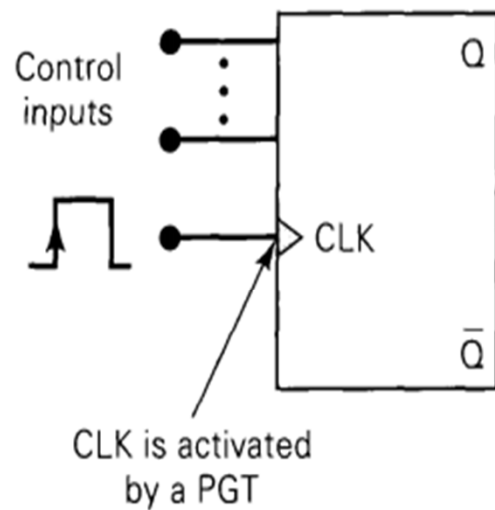
# Review (1)

- Logic gates

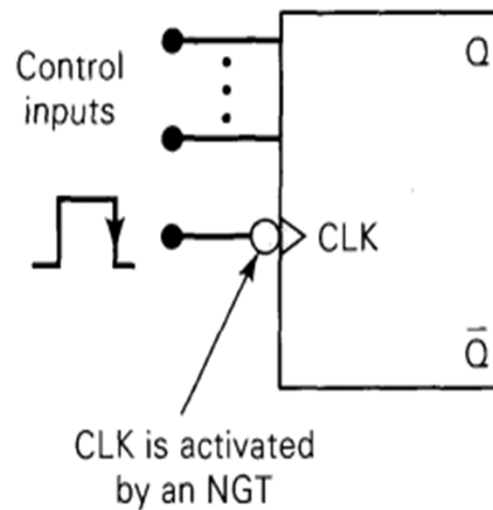


# Review (2)

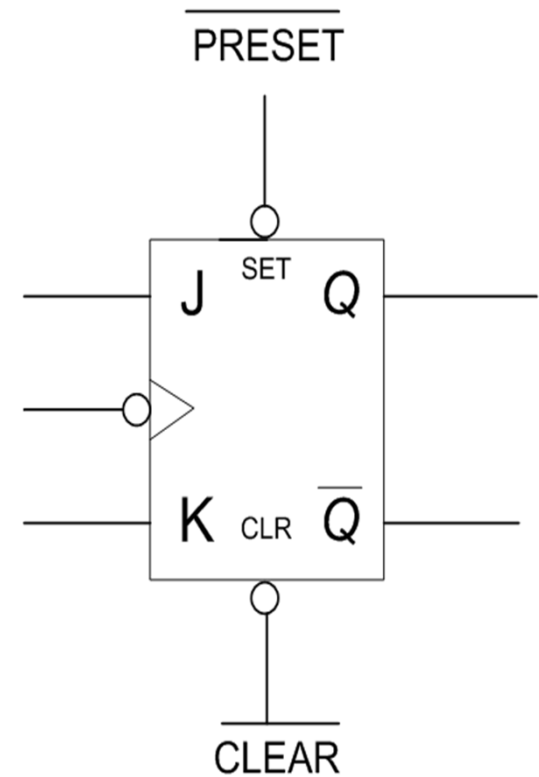
- Flip-Flop: SC-FF, JK-FF, D-FF



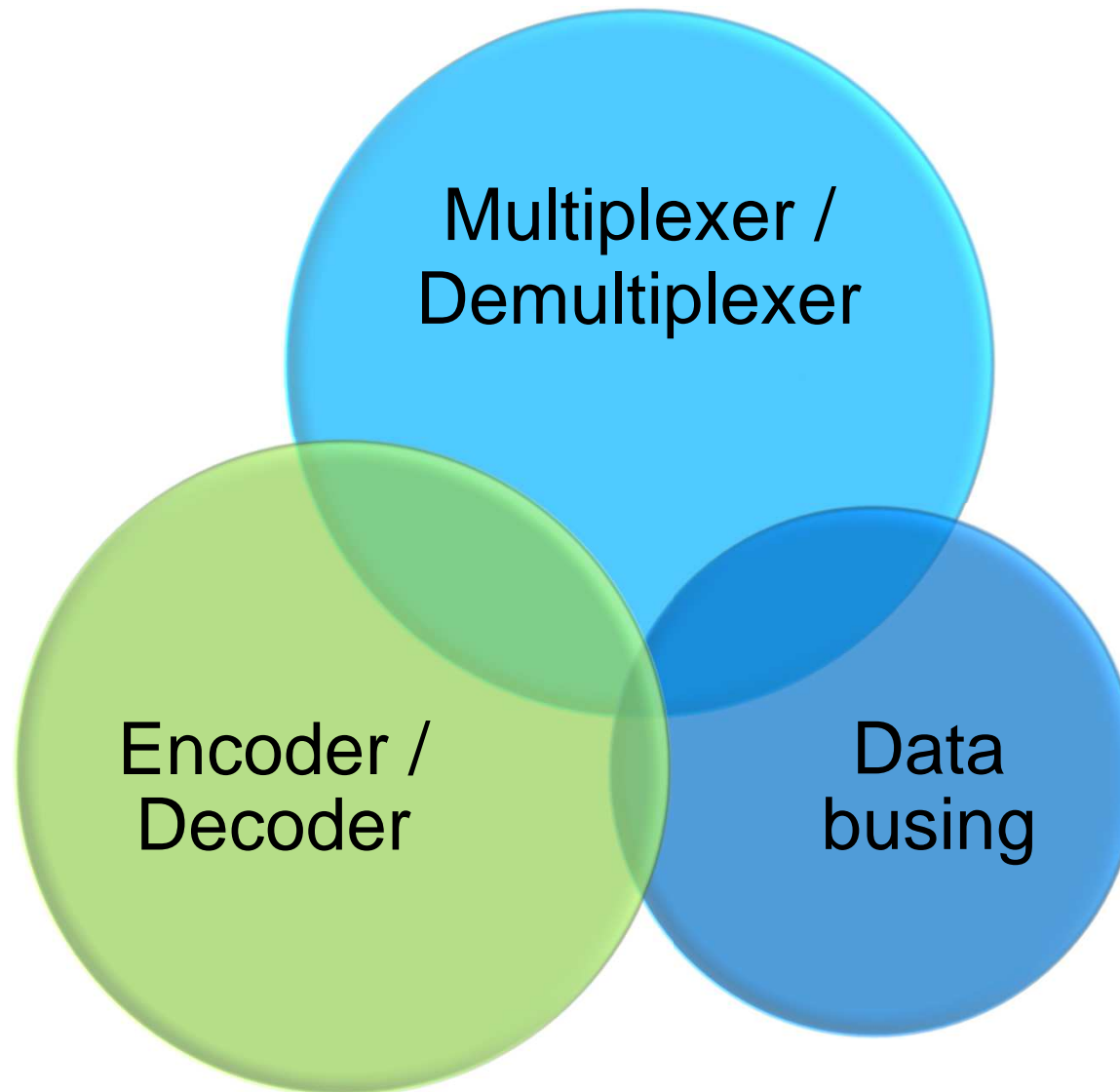
(a)



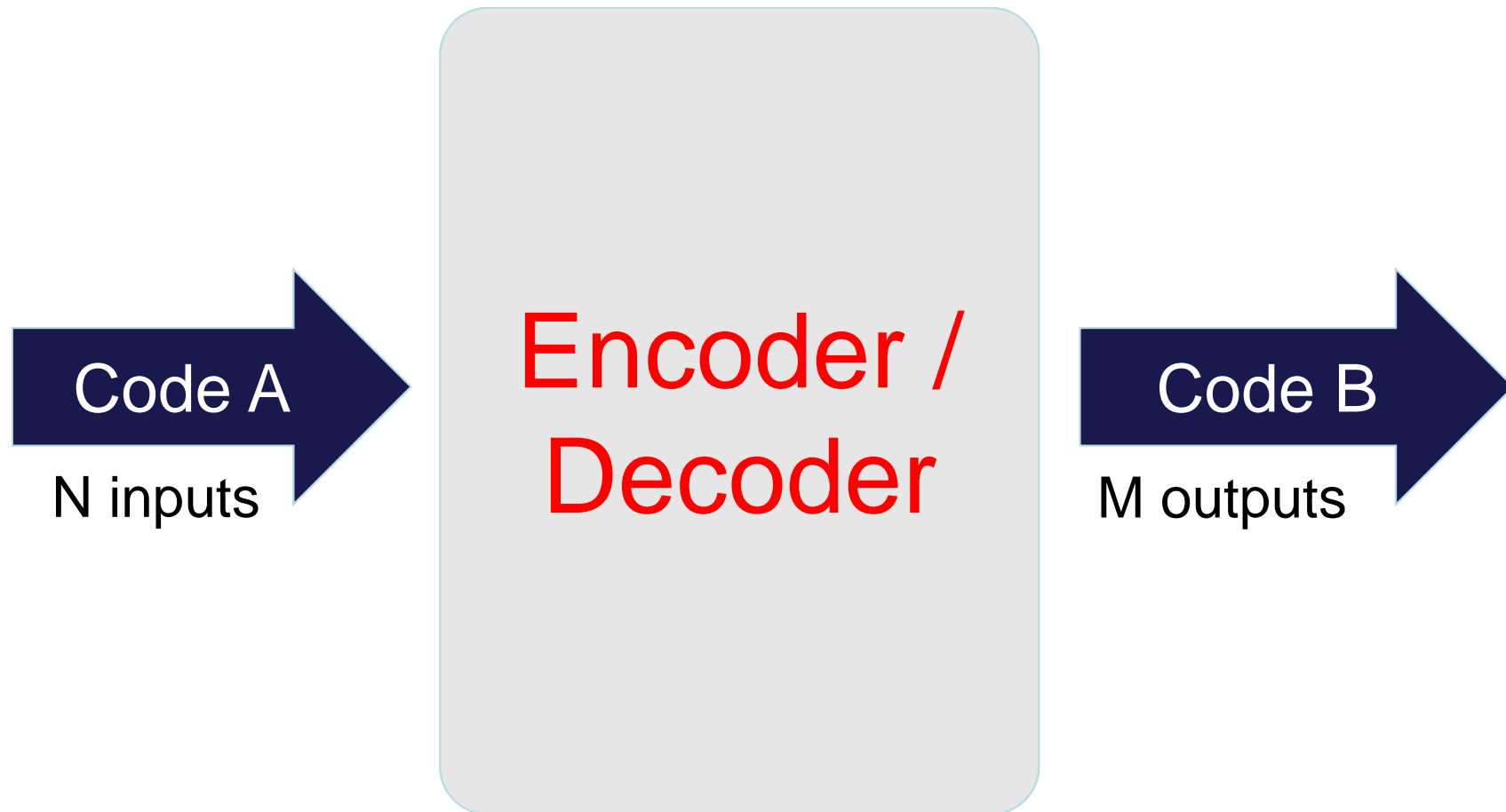
(b)





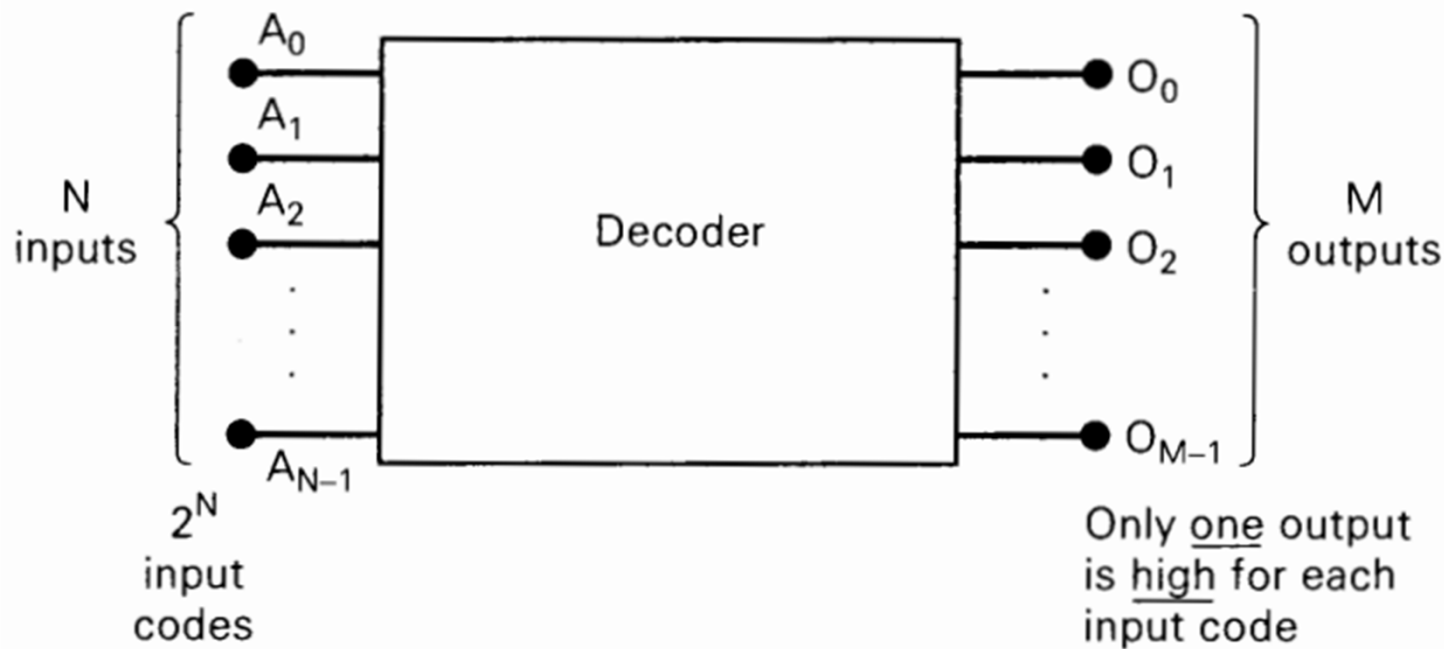


# What is encoder/decoder



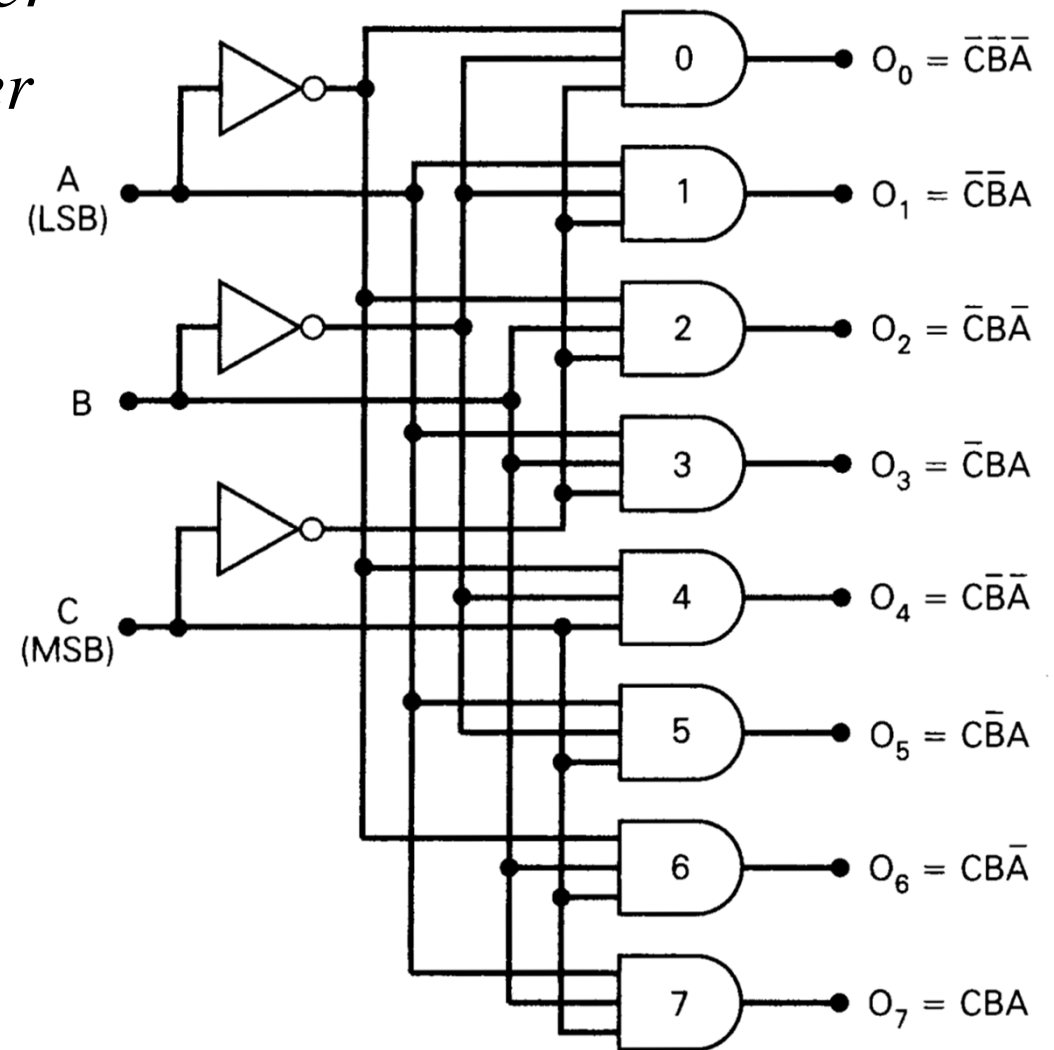
# Decoder

- Decoder N inputs, M outputs
  - Input: represent a binary number
  - Activated output: corresponding to the input number
- May not utilize all of  $2^N$  possible input codes



# Decoder – Example

- *3-line-to-8-line decoder*
  - *binary-to-octal decoder*
  - *1-of-8 decoder*

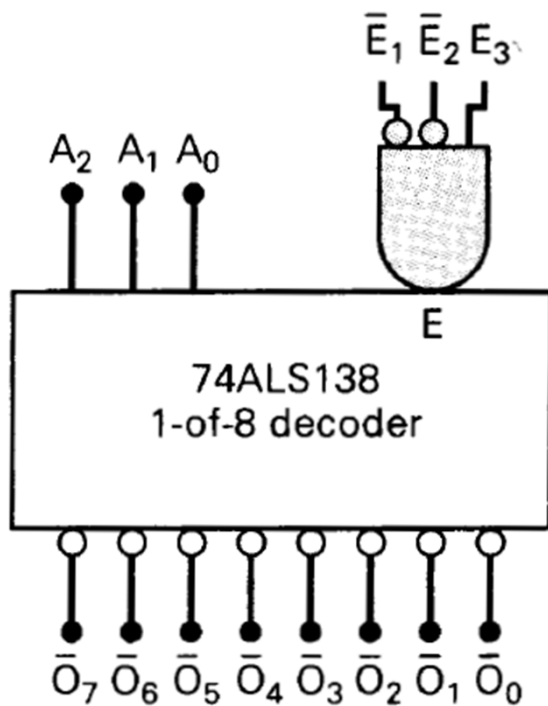


# Decoder – Example

C	B	A	O <sub>7</sub>	O <sub>6</sub>	O <sub>5</sub>	O <sub>4</sub>	O <sub>3</sub>	O <sub>2</sub>	O <sub>1</sub>	O <sub>0</sub>
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

# Decoder – ENABLE inputs

- Some decoders have one or more ENABLE inputs to control the operation of the decoder
- *Example: 74138*



$\bar{E}_1$	$\bar{E}_2$	$E_3$	Outputs
0	0	1	Respond to input code $A_2A_1A_0$
1	X	X	Disabled – all HIGH
X	1	X	Disabled – all HIGH
X	X	0	Disabled – all HIGH

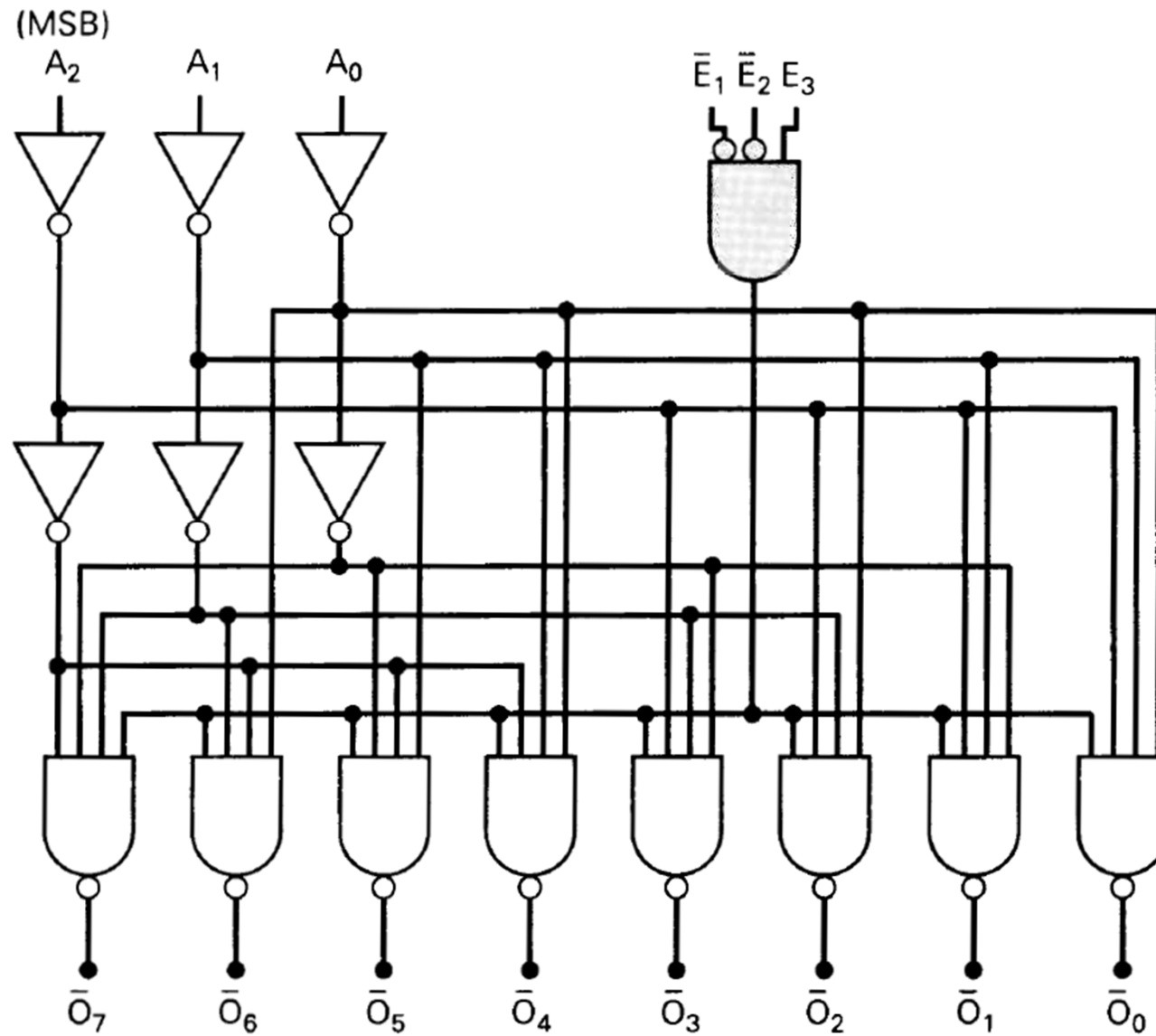
# Exercises

- Determine outputs of 74xx138

$$- E_3 = E_2 = E_1 = 0 \quad A_2 = A_1 = 0 \quad A_0 = 0$$

$$- E_3 = 1, E_2 = E_1 = 0 \quad A_2 = 0 \quad A_1 = A_0 = 1$$

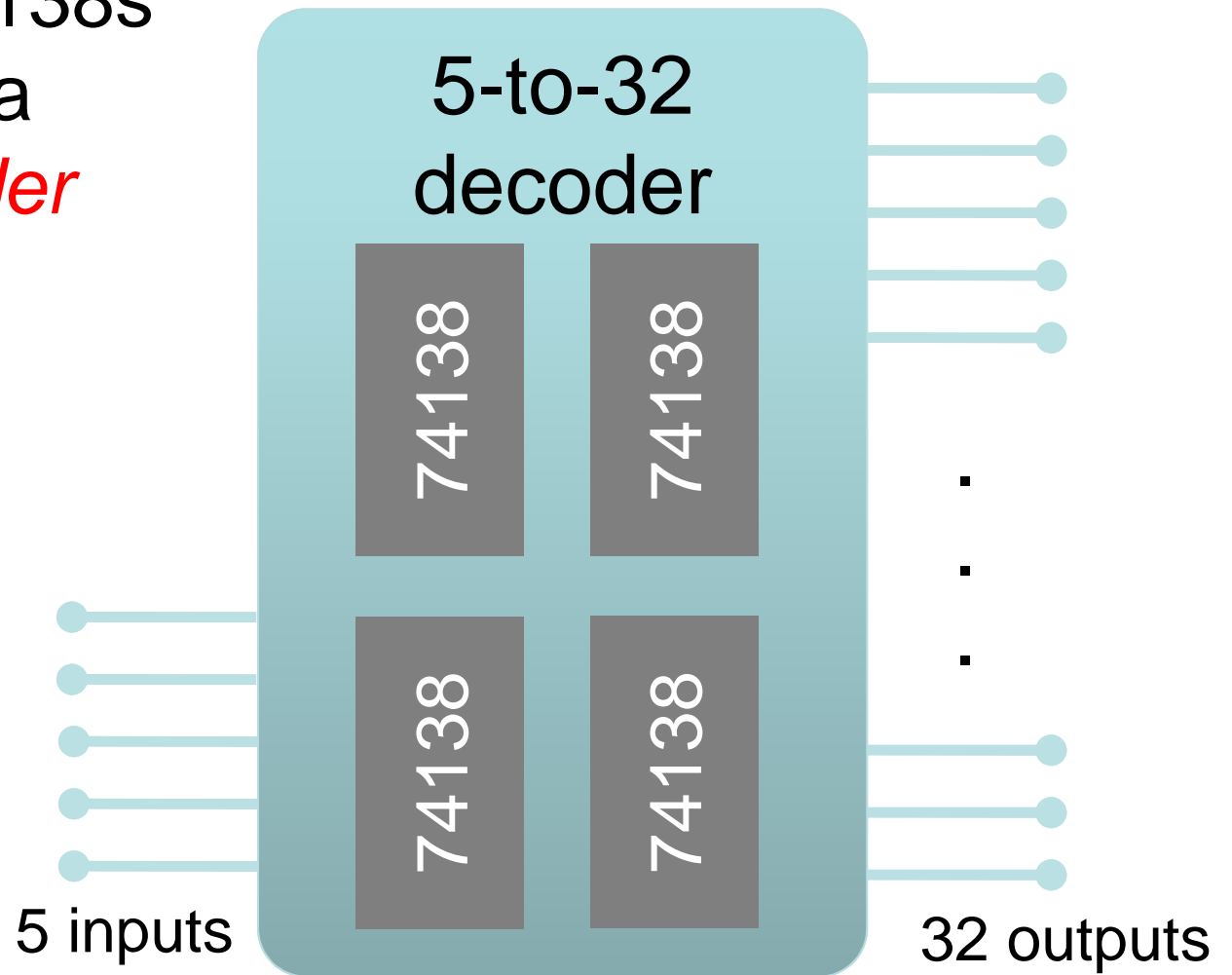
# 74LS138 Circuit





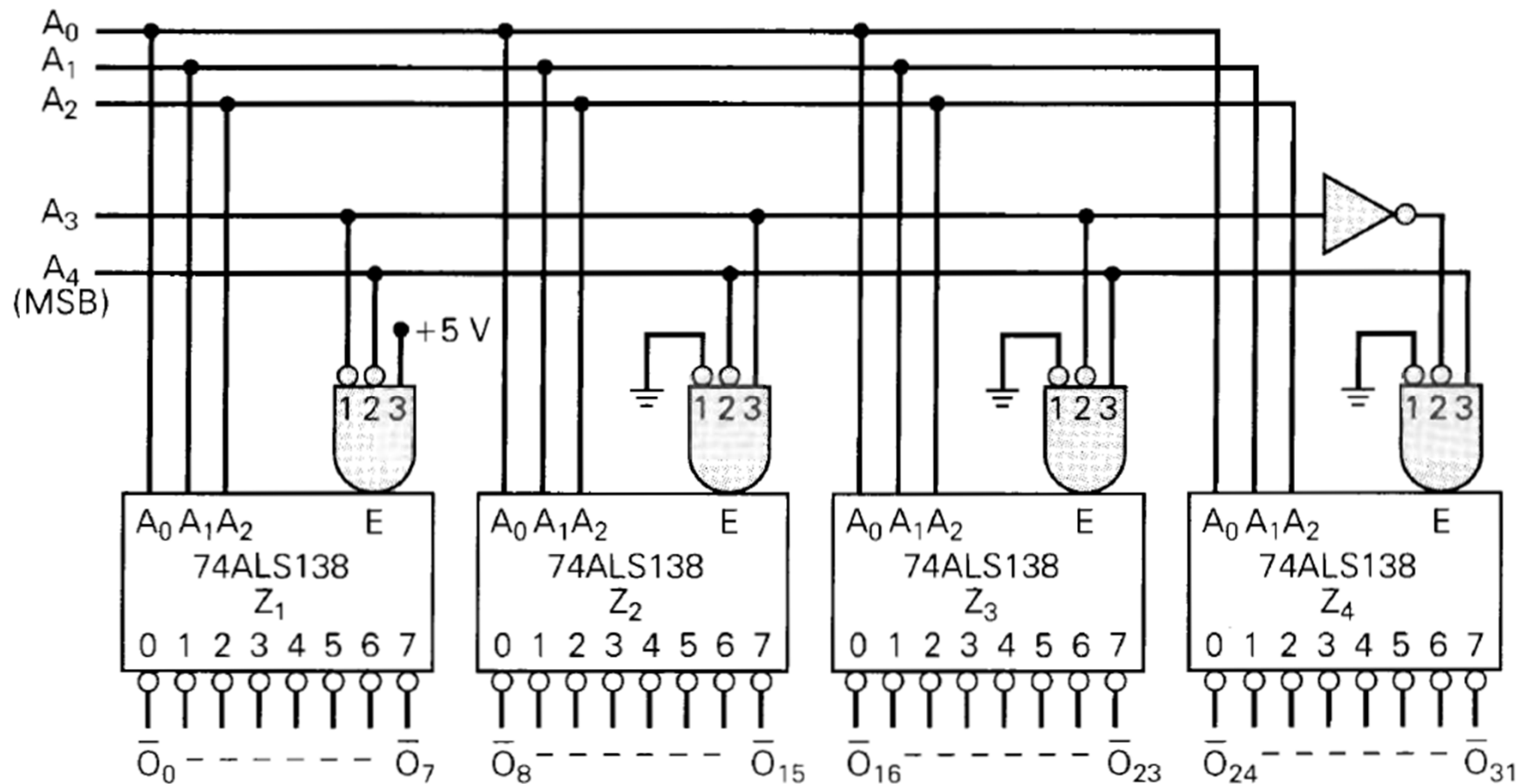
# Problem 1

- Using **four** 74138s to implement a **5-to-32 decoder**



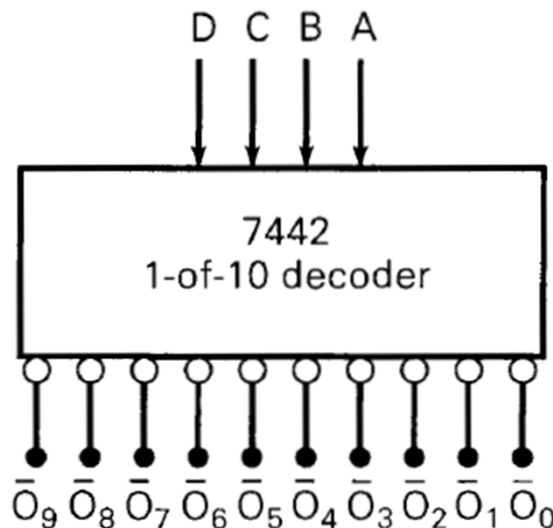
# Decoder - Combination

- Four 74ALS138s and an INVERTER can be arranged to function as a 1-of-32 decoder



# BCD-to-Decimal Decoder (7442)

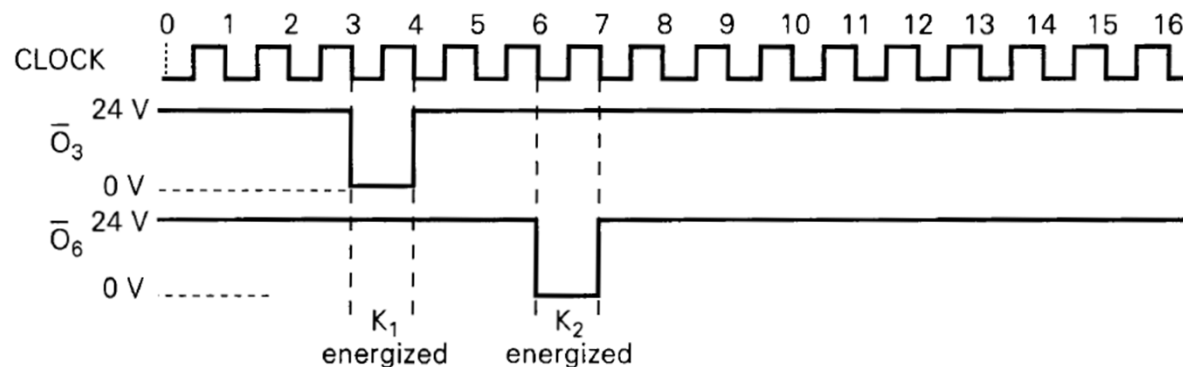
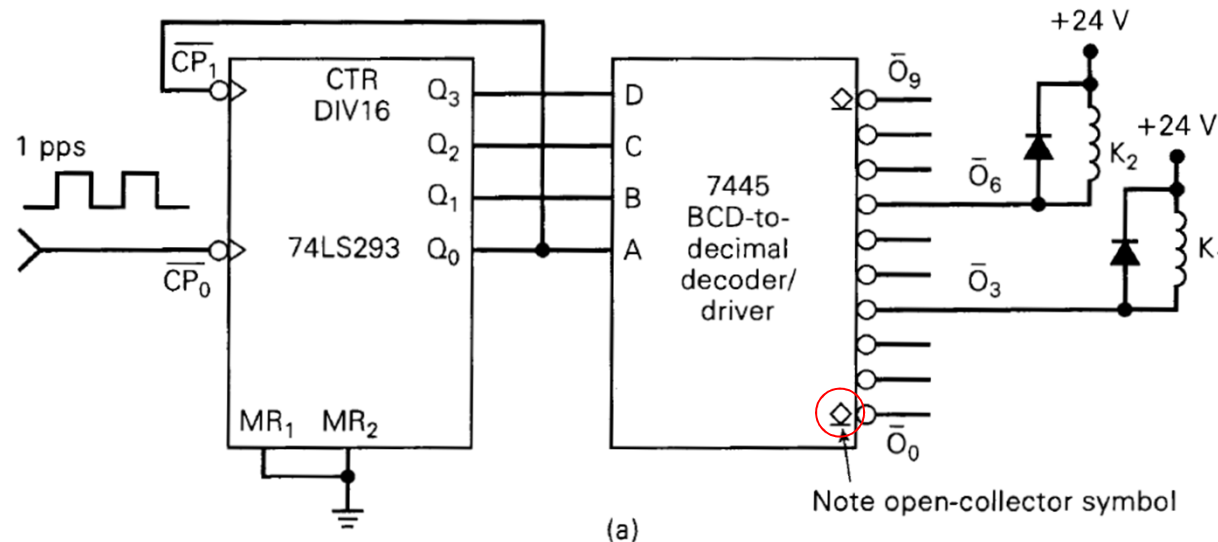
- 4-to-10 decoder (1-of-10 decoder)
  - Activate one output that corresponds to BCD input
  - **Invalid inputs** (1010  $\rightarrow$  1111): *none of outputs will be activated*



Inputs				Active Output
D	C	B	A	
L	L	L	L	$\bar{O}_0$
L	L	L	H	$\bar{O}_1$
L	L	H	L	$\bar{O}_2$
L	L	H	H	$\bar{O}_3$
L	H	L	L	$\bar{O}_4$
L	H	L	H	$\bar{O}_5$
L	H	H	L	$\bar{O}_6$
L	H	H	H	$\bar{O}_7$
H	L	L	L	$\bar{O}_8$
H	L	L	H	$\bar{O}_9$
H	L	H	L	None
H	L	H	H	None
H	H	L	L	None
H	H	L	H	None
H	H	H	L	None
H	H	H	H	None

# Decoder Applications

- Counter/Decoder combination used to provide timing and sequencing operations

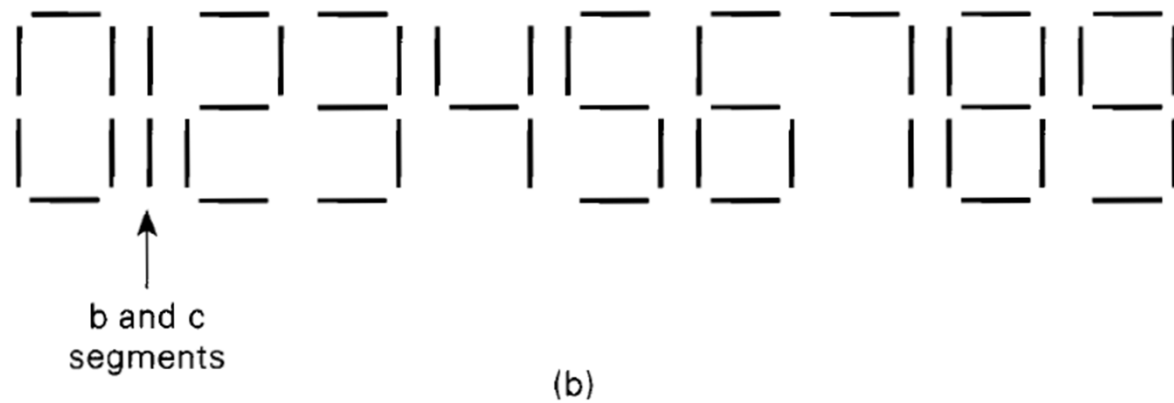
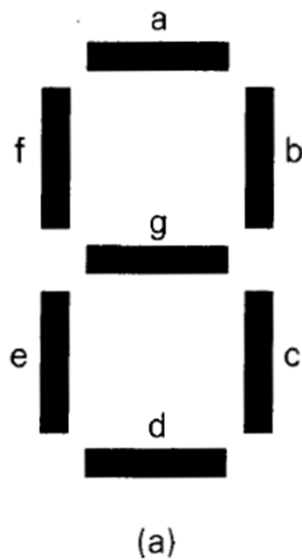


# BCD-to-Decimal Decoder/Driver

- Open-collector outputs can operate at higher current and voltage limits than a normal TTL outputs
- Directly driving loads: LEDs, lamps, relays, dc motors, ...
- *Example:* 7445 outputs can sink up to 80mA in LOW state

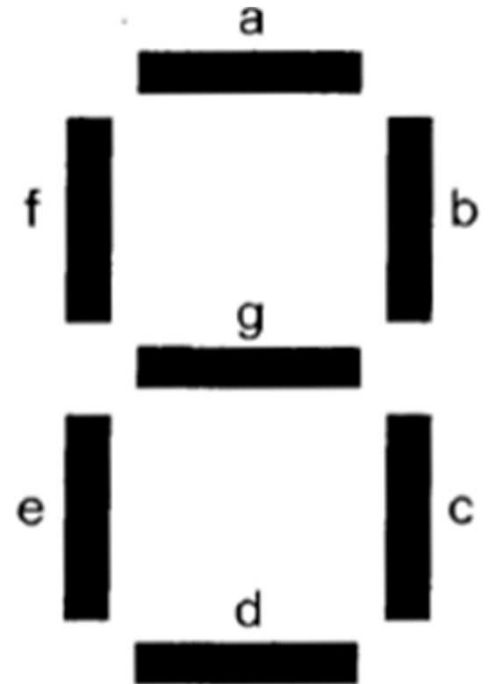
# 7-Segment LED

- Form the decimal characters 0  $\rightarrow$  9, sometimes the hex characters A  $\rightarrow$  F
- Use light-emitting diodes (LEDs) for each segment
- Normal brightness: 10mA, 2.7V



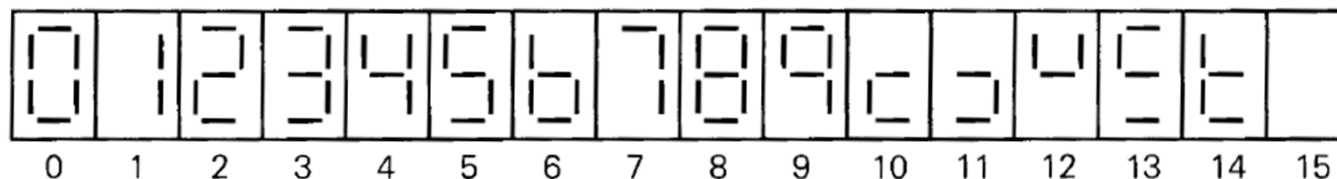
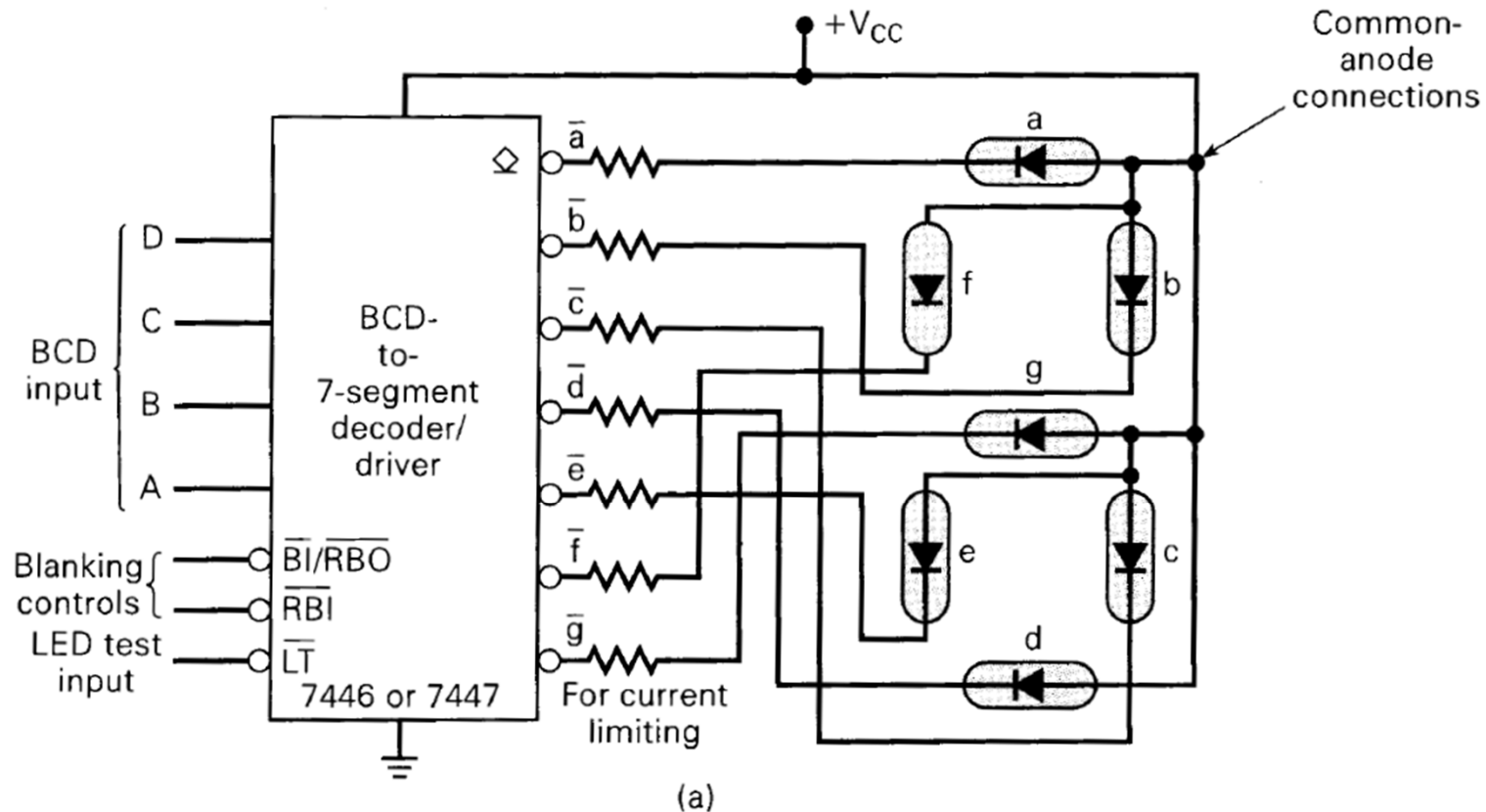
# Exercises

- Determines outputs for each case
  - Inputs DCBA = 1001
  - Inputs DCBA = 0110
  - Inputs DCBA = 0011
  - Inputs DCBA = 0000
- D is the most significant bit
- 4-bit BCD input, each output is activated more than one combination of inputs



# BCD-to-7-segment Decoder/Driver

- Common-Anode, Common-Cathode LEDs





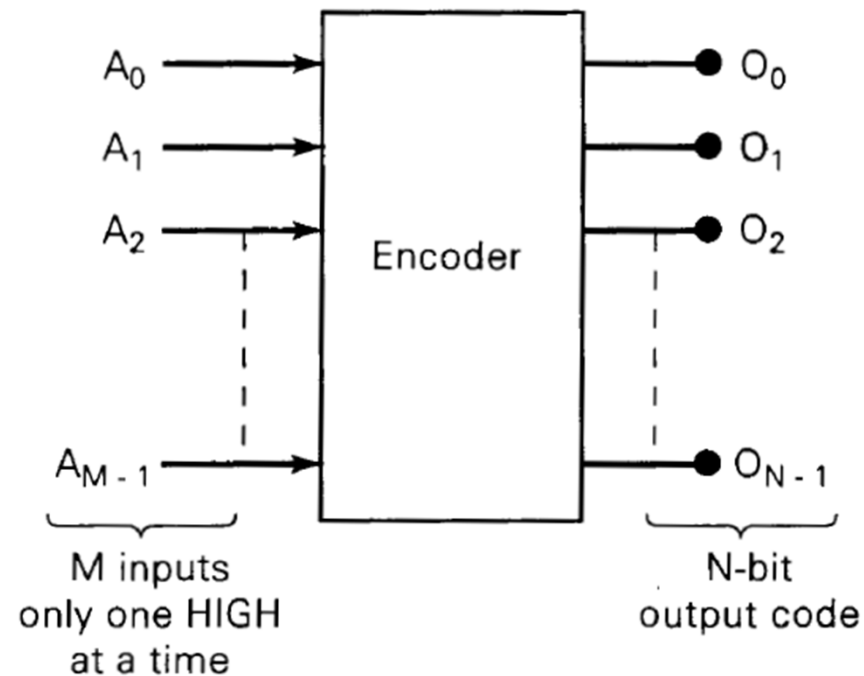
# BCD-to-7-segment Decoder/Driver

TRUTH TABLE

INPUTS							OUTPUTS								NOTE
DECIMAL OR FUNCTION	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	D	C	B	A	$\overline{\text{BI/RBO}}$	$\overline{\text{a}}$	$\overline{\text{b}}$	$\overline{\text{c}}$	$\overline{\text{d}}$	$\overline{\text{e}}$	$\overline{\text{f}}$	$\overline{\text{g}}$	
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	A
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	A
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	B
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	C
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	D

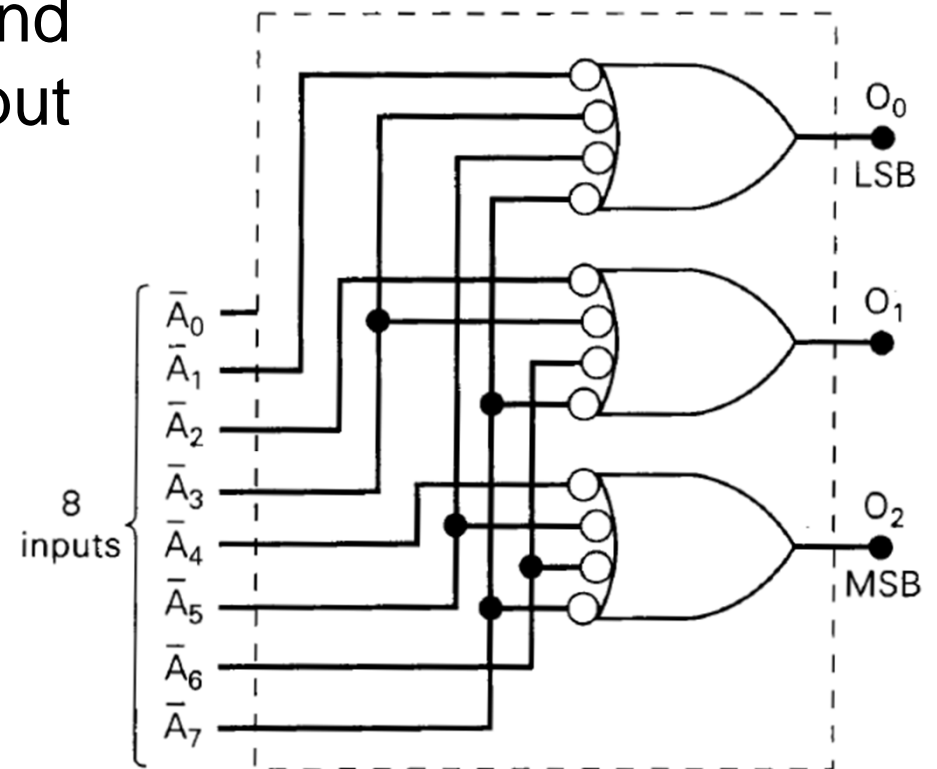
# Encoder

- The opposite of decoding process
- Encoder has a number of input lines, only one of which is activated at a given time, and produces a N-bit output code



# Encoder - Example

- Octal-to-binary encoder  
(*8-line-to-3-line encoder*)
  - Accept 8 inputs lines and produce a 3-bit output code



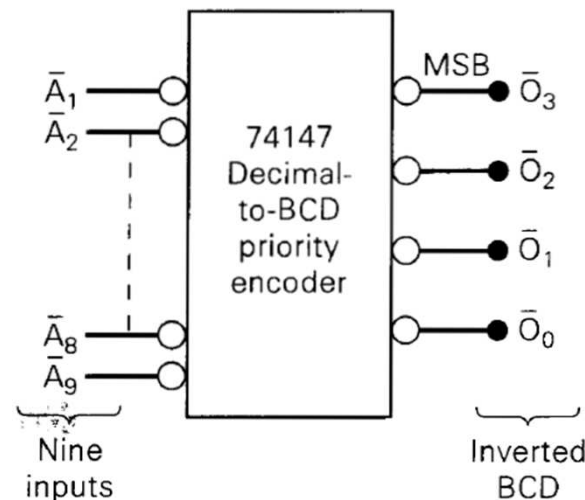
\*Only one  
LOW input  
at a time

# Encoder - Example

Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$O_2$	$O_1$	$O_0$
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

# Priority Encoder

- When *more than one input is activated* at one time, the output code will respond to the *highest-numbered input*
  - Example: A6, A2, A0 are activated → output code is 110 (6)
- 74147 Decimal-to-BCD Priority Encoder



# Decimal-to-BCD Priority Encoder

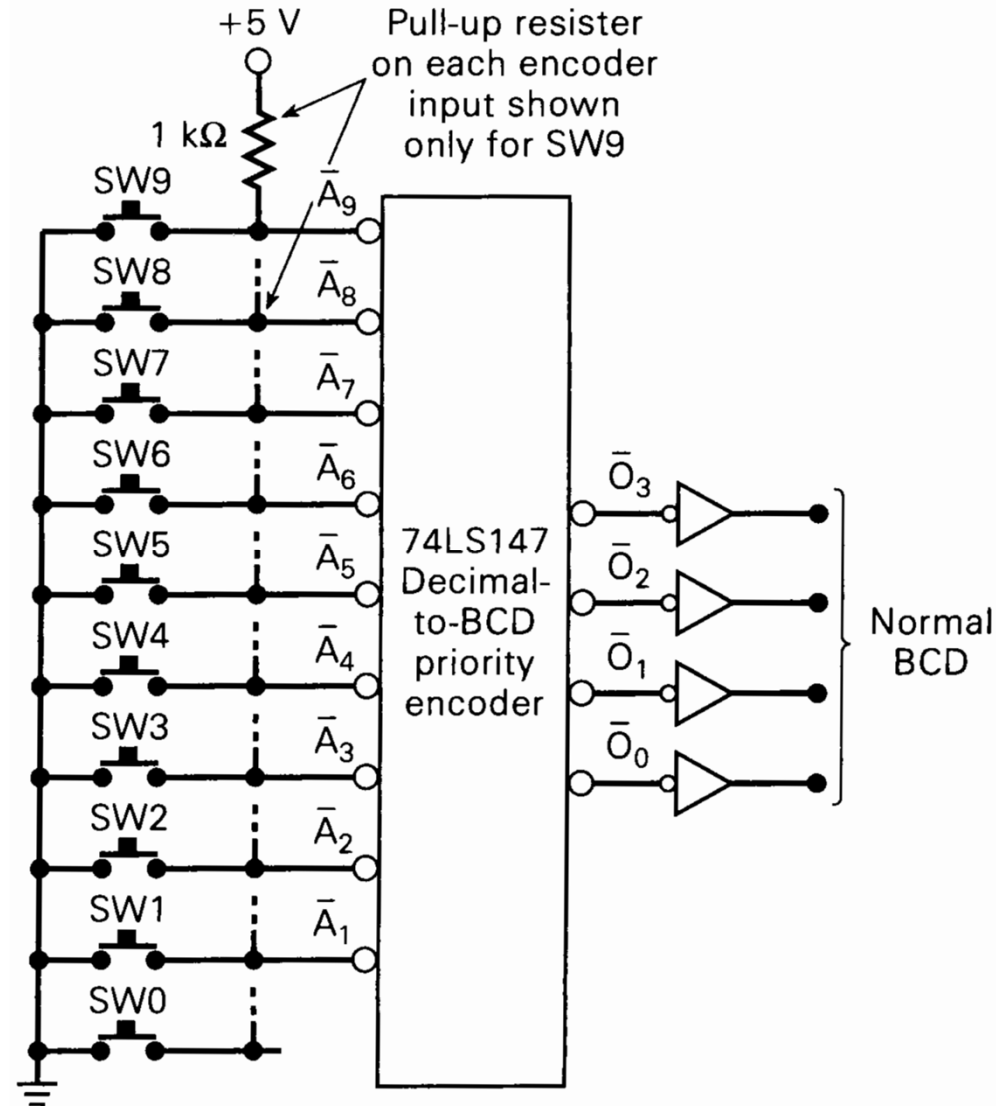
$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$\bar{A}_8$	$\bar{A}_9$					$\bar{O}_3$	$\bar{O}_2$	$\bar{O}_1$	$\bar{O}_0$
1	1	1	1	1	1	1	1	1					1	1	1	1
X	X	X	X	X	X	X	X	0					0	1	1	0
X	X	X	X	X	X	X	0	1					0	1	1	1
X	X	X	X	X	X	0	1	1					1	0	0	0
X	X	X	X	X	0	1	1	1					1	0	0	1
X	X	X	X	0	1	1	1	1					1	0	1	0
X	X	X	0	1	1	1	1	1					1	0	1	1
X	X	0	1	1	1	1	1	1					1	1	0	0
X	0	1	1	1	1	1	1	1					1	1	0	1
0	1	1	1	1	1	1	1	1					1	1	1	0

X = either 0 or 1

# Exercises

- Determine outputs of 74147 if  $A_9$ - $A_0$  inputs are
  - 111011111
  - All high except  $A_7$ ,  $A_5$ ,  $A_3$
  - All low except  $A_9$ ,  $A_1$ ,  $A_0$

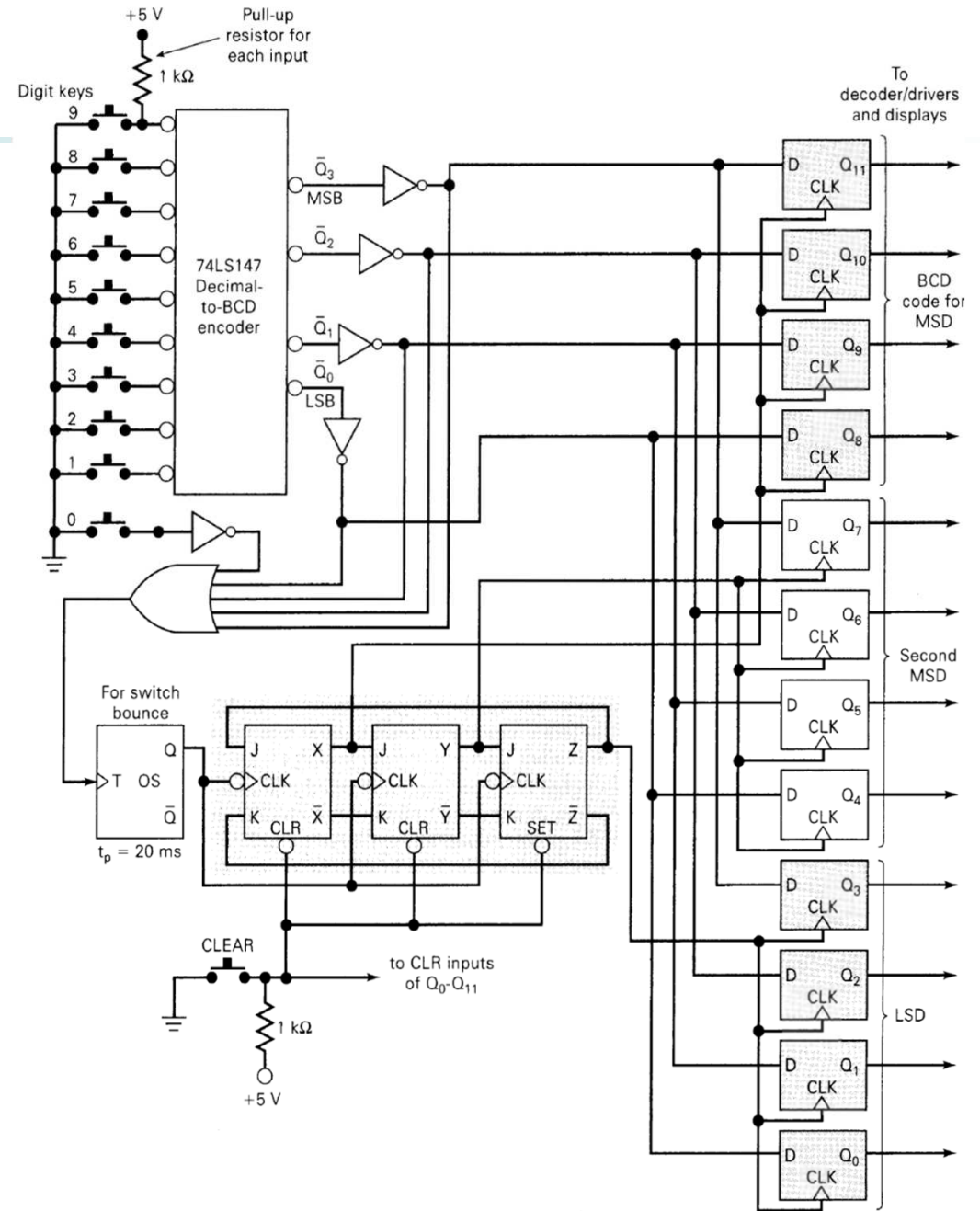
# Switch Encoder

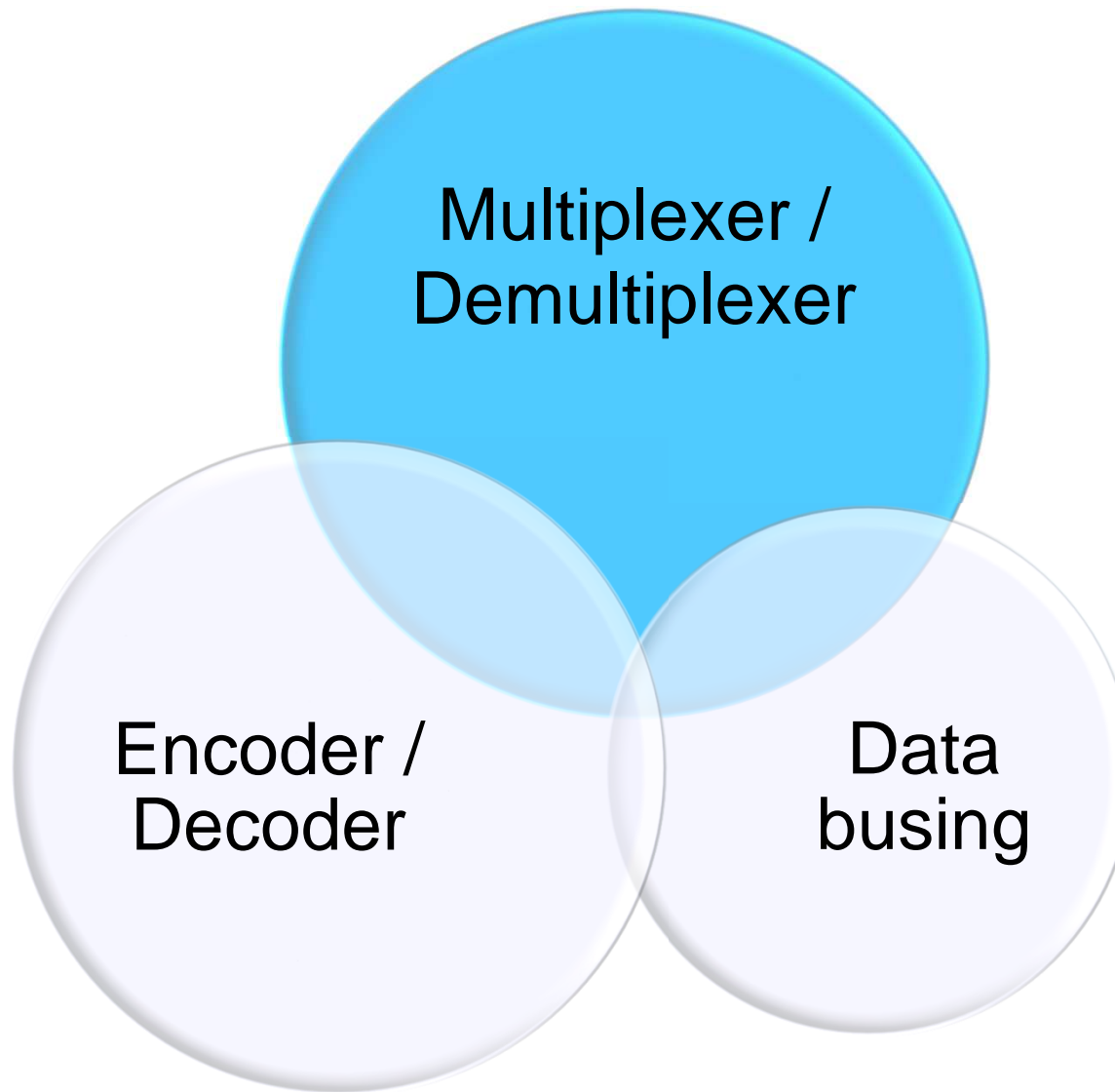




# Application

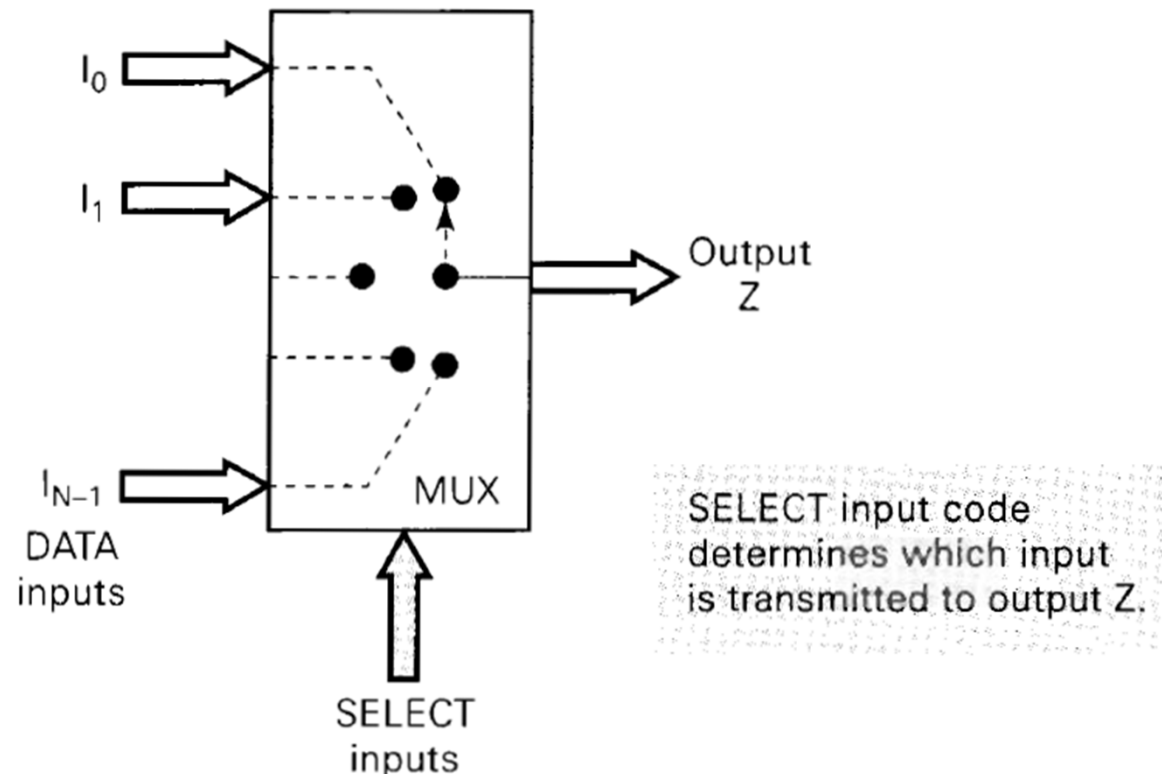
- 12 D-FF: three 4-bit FF registers
- JK-FF: control the transfer of data to the appropriate register
- T-FF: One-shot





# Multiplexers (MUX - Data Selectors)

- Accept several digital data inputs
- Select *one of them* to pass on to the output
- Desired input is controlled by *SELECT inputs*



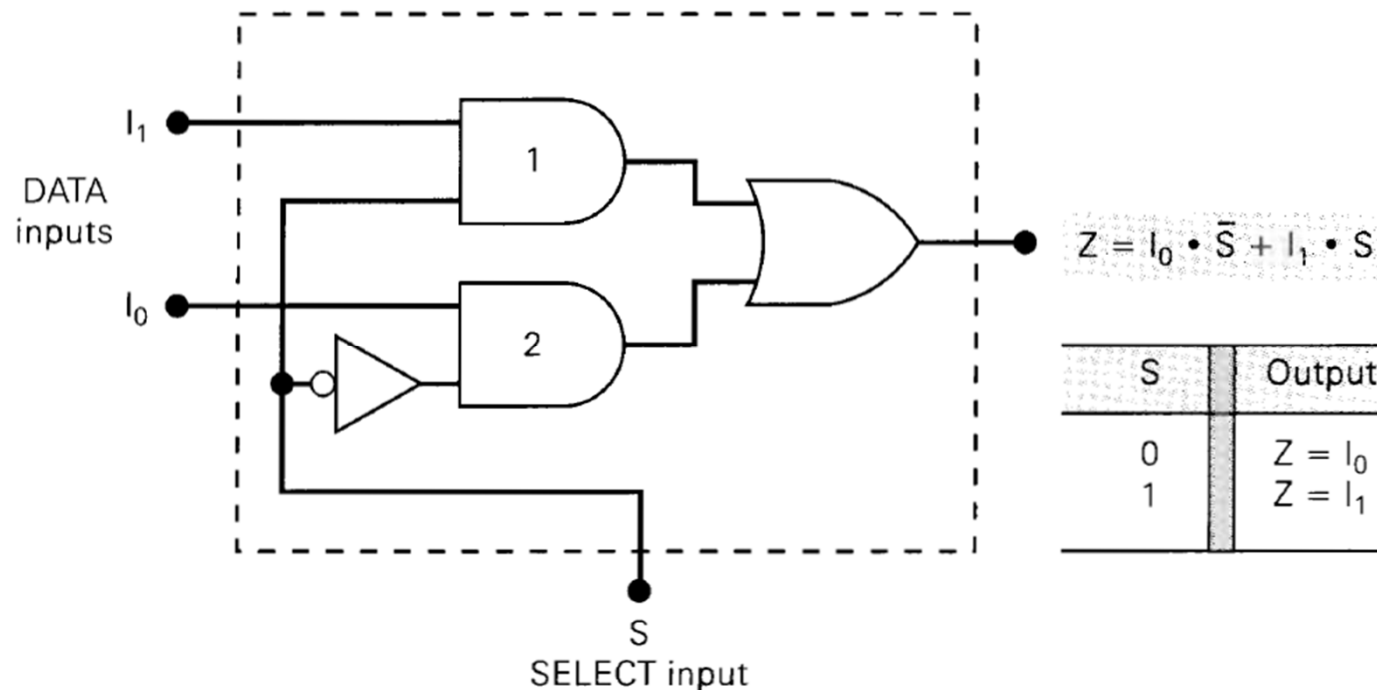
# Basic Two-Input Multiplexer

- Boolean expression for the output

$$Z = I_0S' + I_1S$$

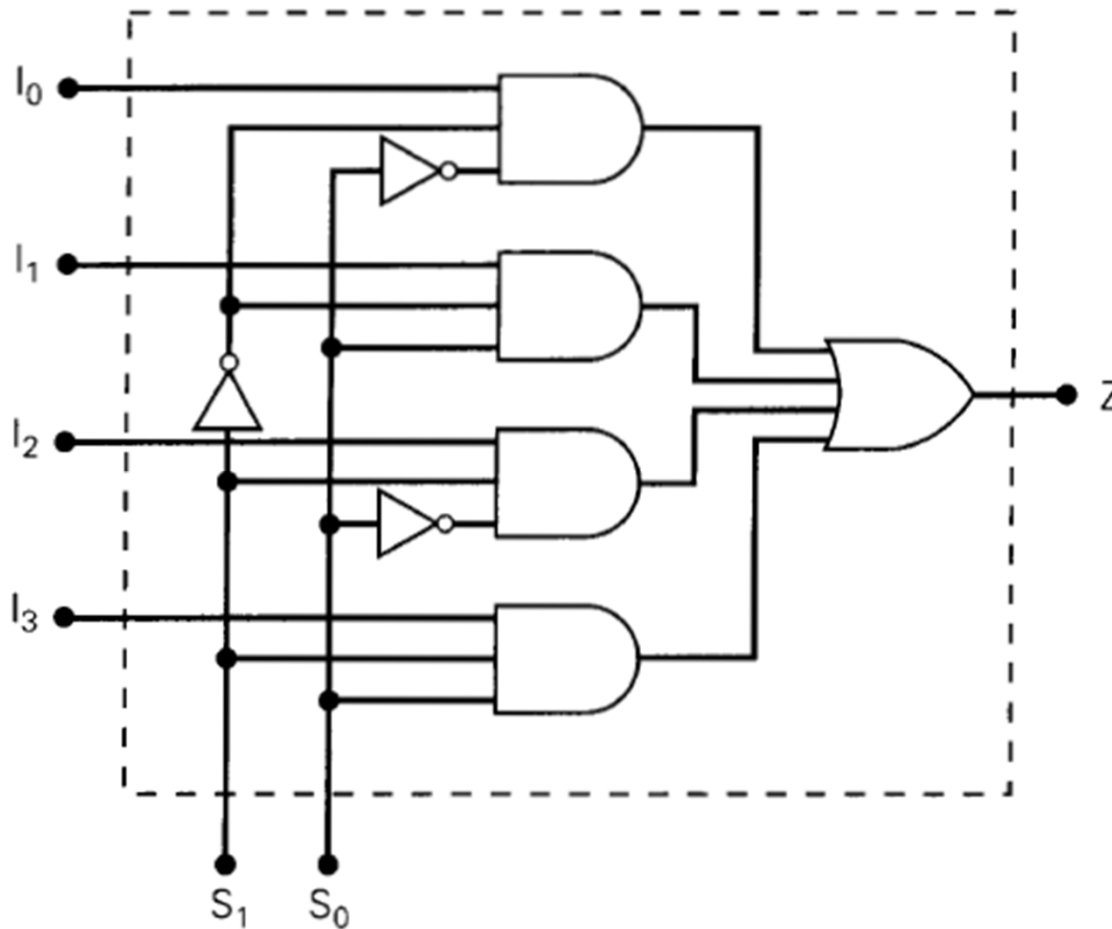
–  $S = 0$ :  $Z = I_0 \cdot 1 + I_1 \cdot 0 = I_0$  (gate 2 enabled)

–  $S = 1$ :  $Z = I_0 \cdot 0 + I_1 \cdot 1 = I_1$  (gate 1 enabled)



# Other Multiplexer (1)

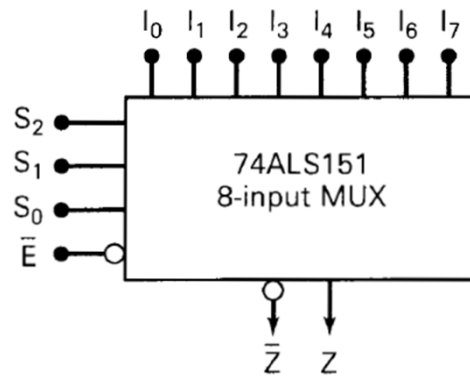
- Four-Input Multiplexer



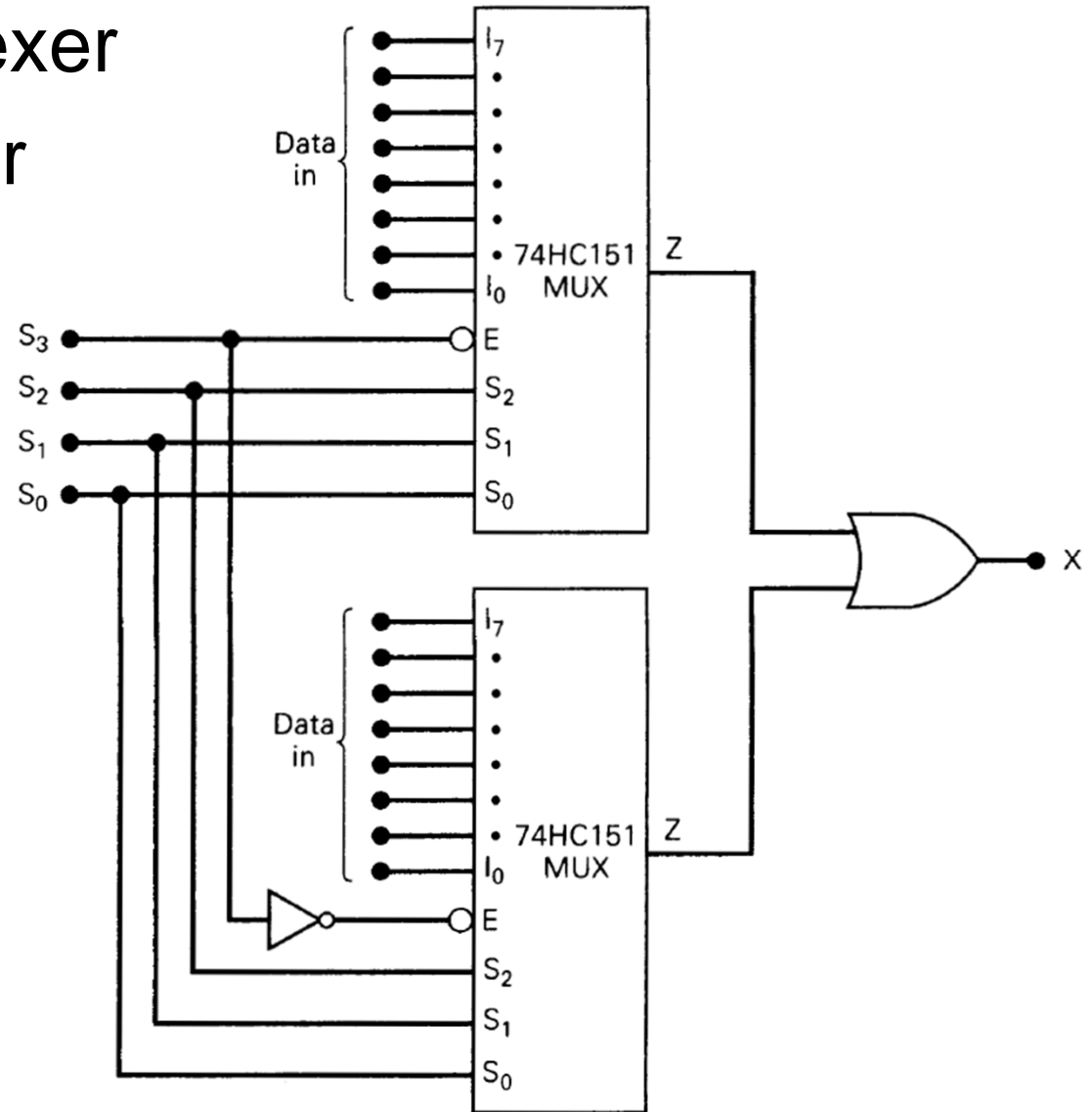
$S_1$	$S_0$	Output
0	0	$Z = I_0$
0	1	$Z = I_1$
1	0	$Z = I_2$
1	1	$Z = I_3$

# Other Multiplexer (2)

- Eight-Input Multiplexer
- 16-Input Multiplexer

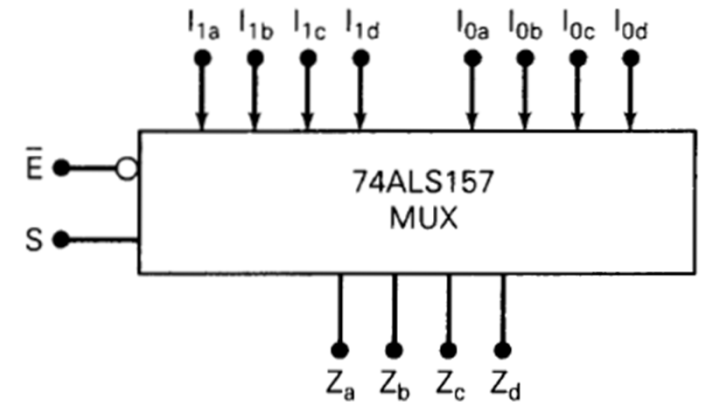
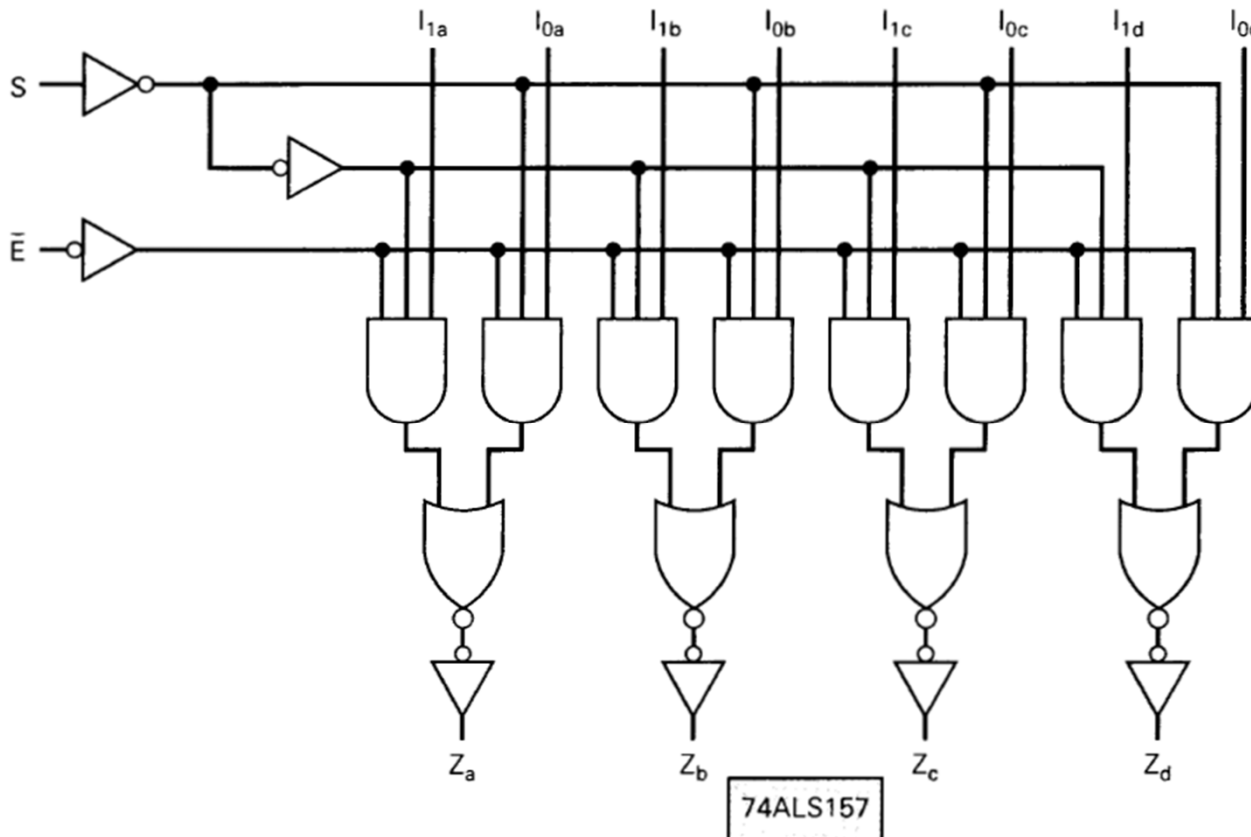


Inputs				Outputs	
$\bar{E}$	$S_2$	$S_1$	$S_0$	$\bar{Z}$	$Z$
H	X	X	X	H	L
L	L	L	L	$\bar{I}_0$	$I_0$
L	L	L	H	$\bar{I}_1$	$I_1$
L	L	H	L	$\bar{I}_2$	$I_2$
L	L	H	H	$\bar{I}_3$	$I_3$
L	H	L	L	$\bar{I}_4$	$I_4$
L	H	L	H	$\bar{I}_5$	$I_5$
L	H	H	L	$\bar{I}_6$	$I_6$
L	H	H	H	$\bar{I}_7$	$I_7$



# Other Multiplexer (3)

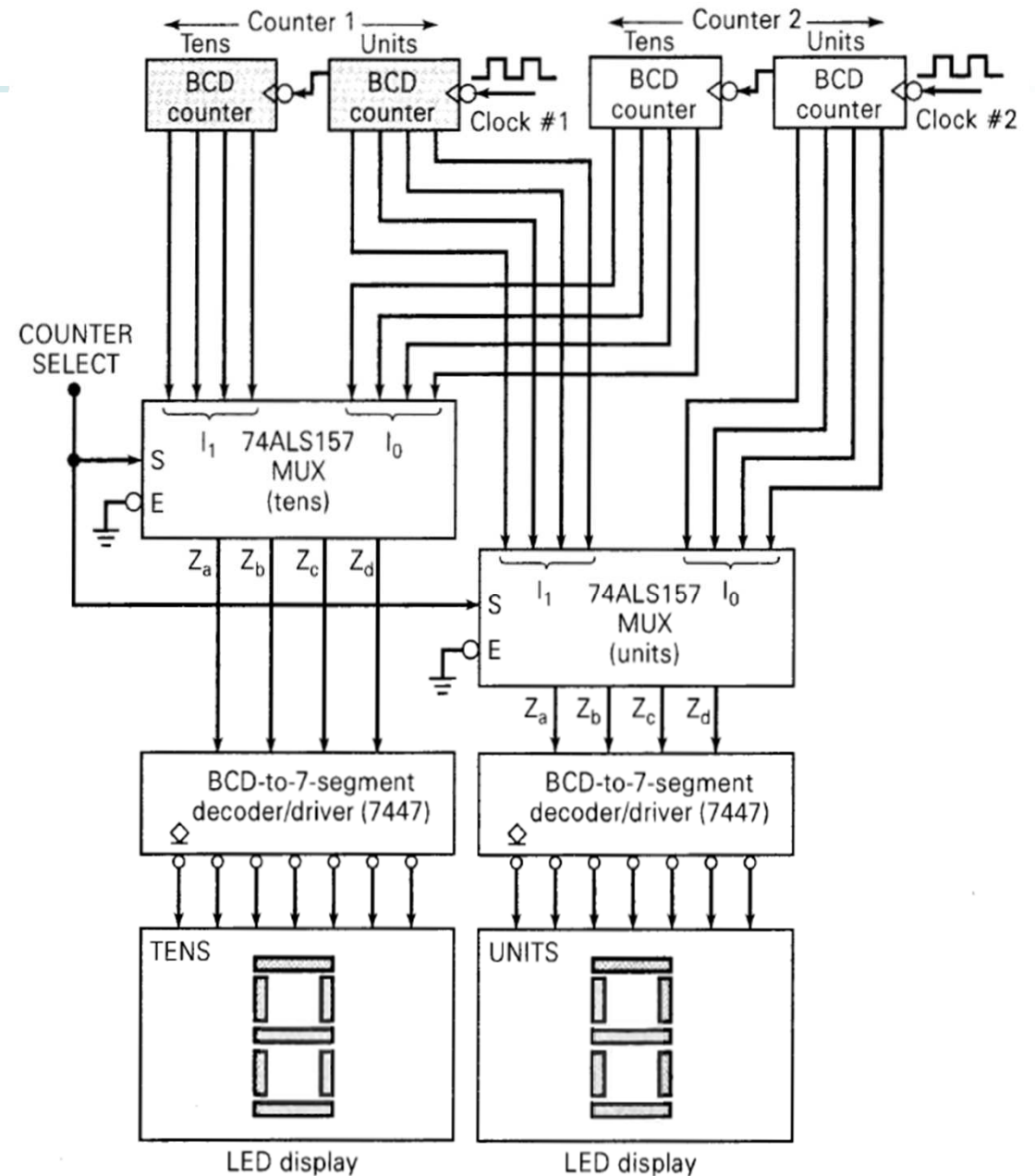
- Quad Two-Input MUX (74157)



$\bar{E}$	$S$	$Z_a$	$Z_b$	$Z_c$	$Z_d$
H	X	L	L	L	L
L	L	$I_{0a}$	$I_{0b}$	$I_{0c}$	$I_{0d}$
L	H	$I_{1a}$	$I_{1b}$	$I_{1c}$	$I_{1d}$

# Applications (1)

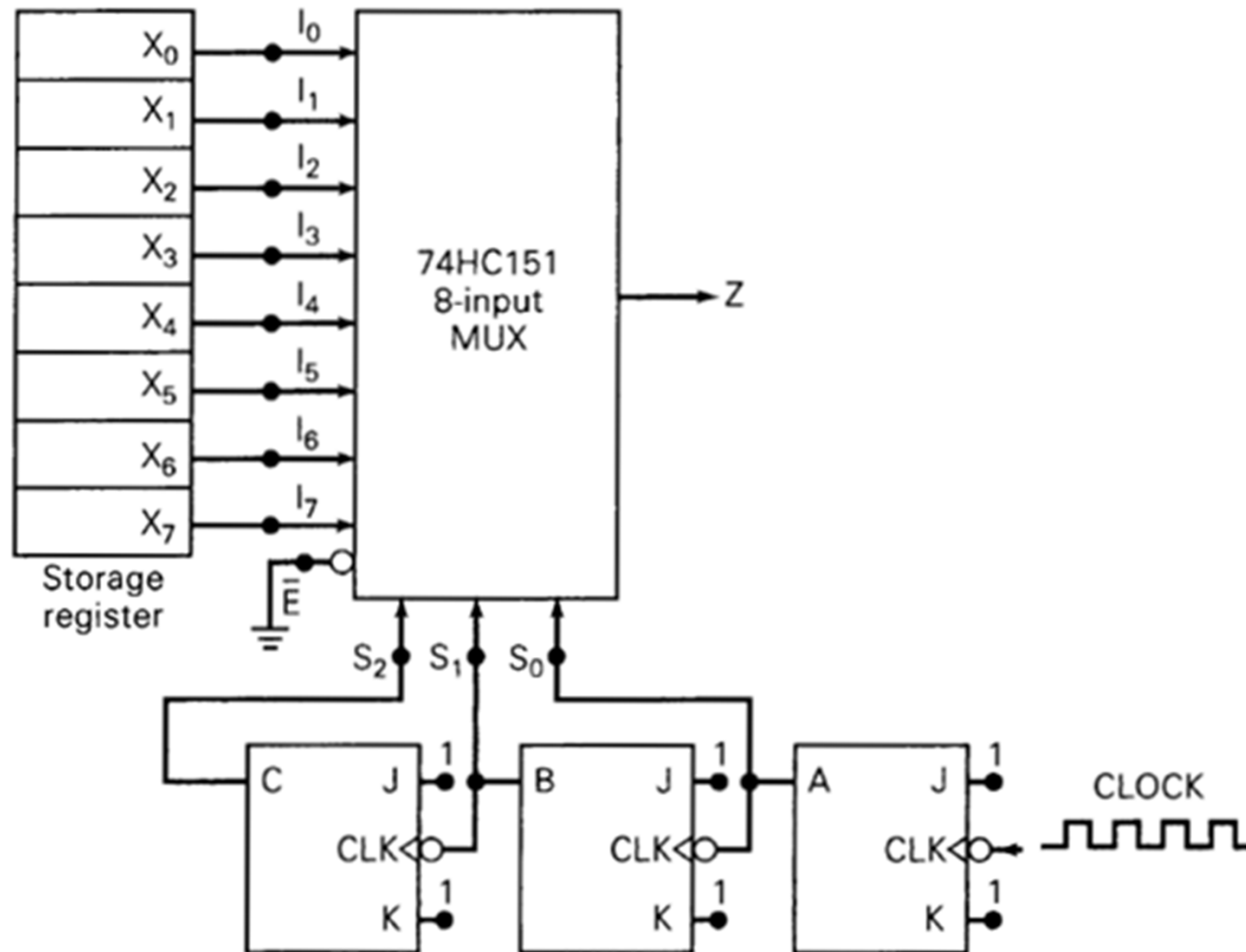
- **Data routing:**
  - **SELECT = 1:** ???
  - **SELECT = 0:** ???
  - *time-share*





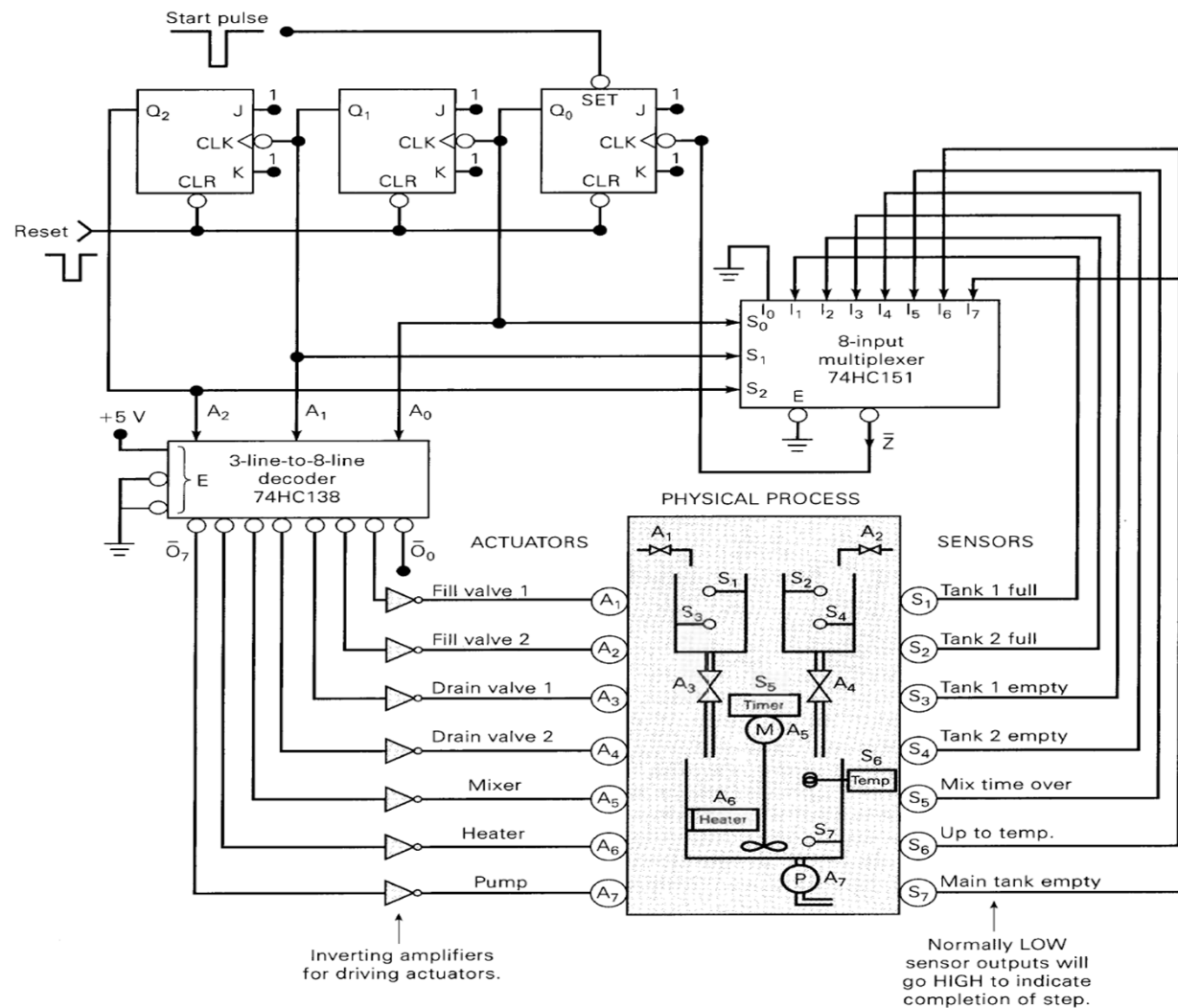
# Applications (2)

- Parallel-to-serial conversion



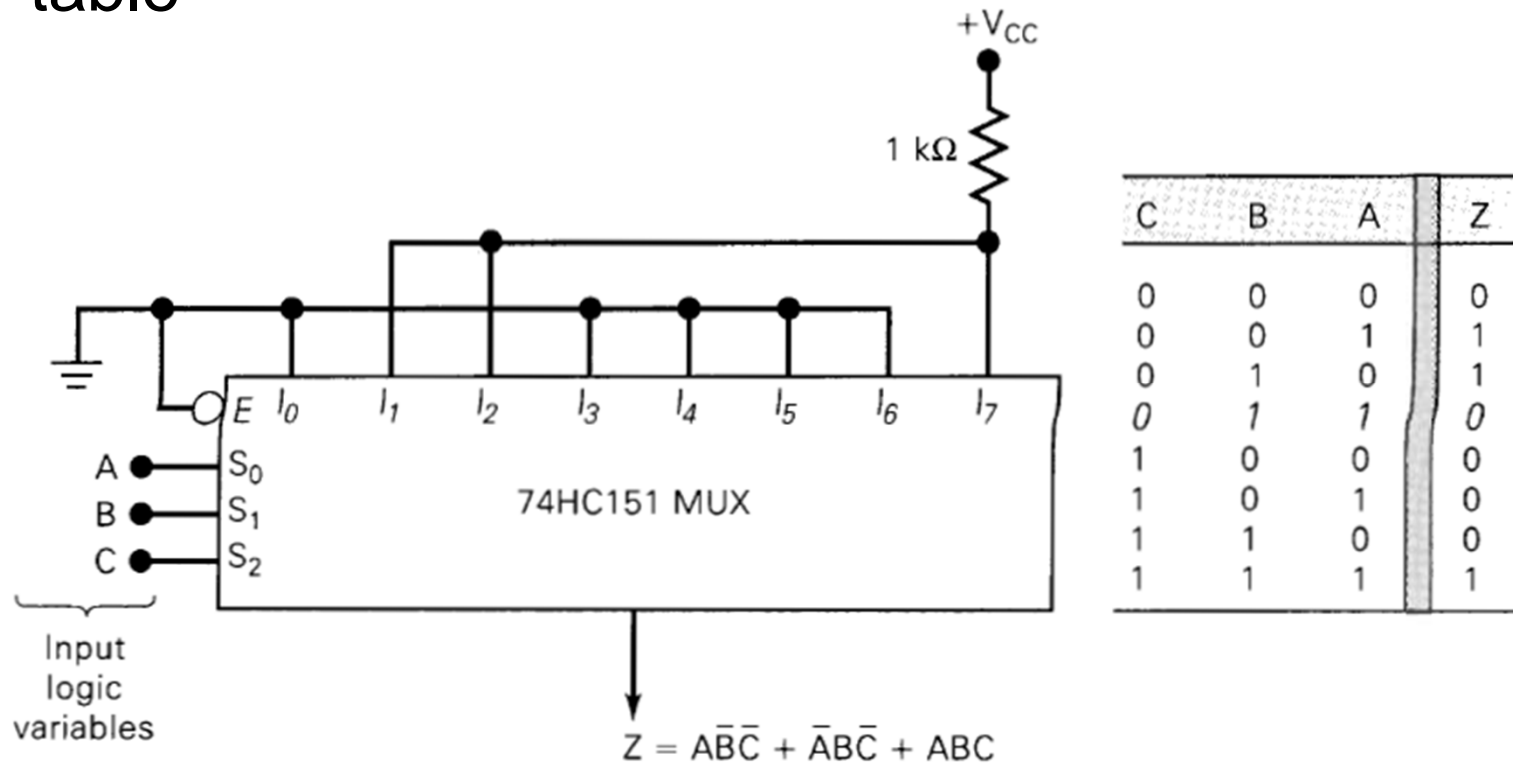
# Applications (3)

- **Operation sequencing:** in a control sequencer



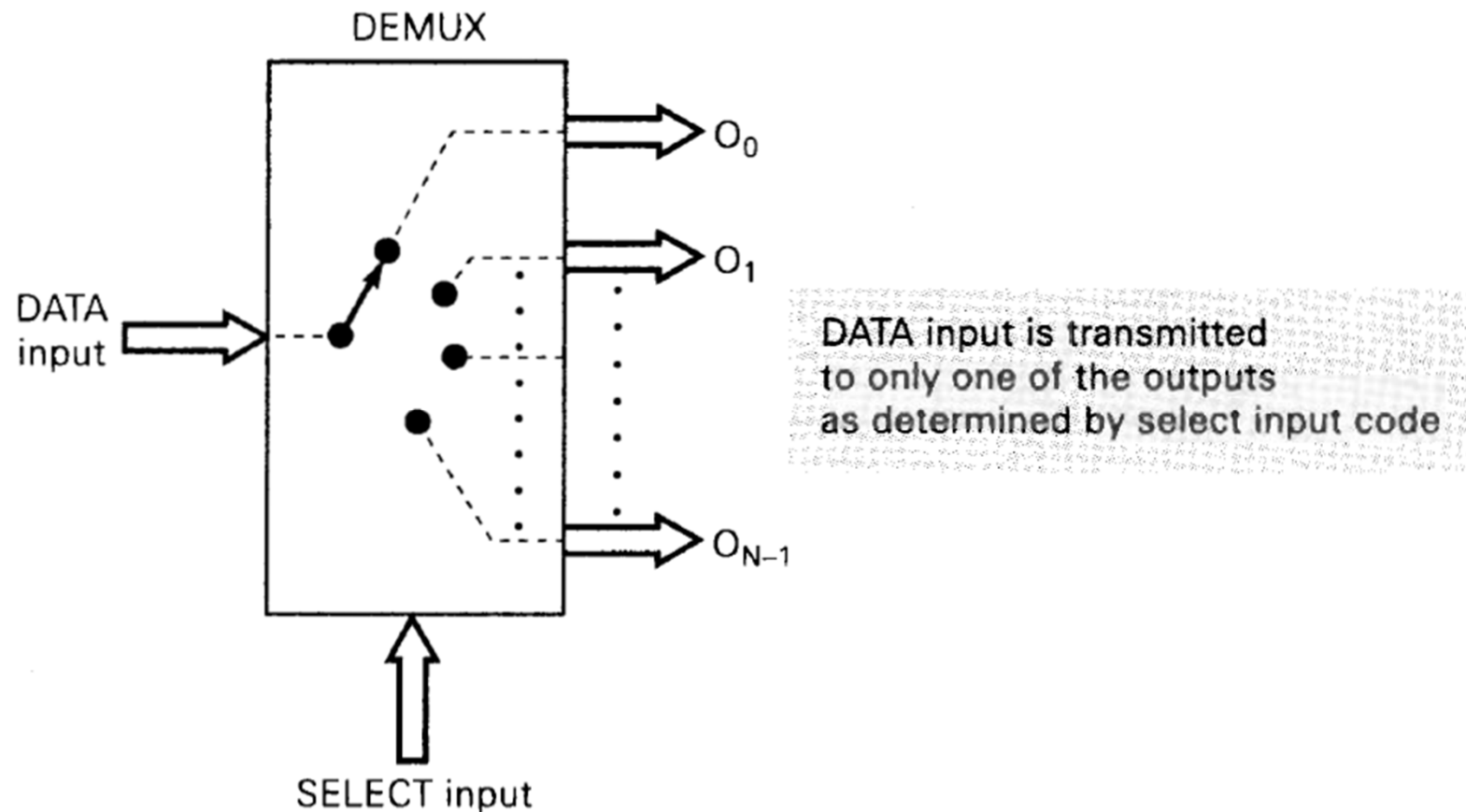
# Applications (4)

- **Logic-function generation**
  - Select inputs: logic variables
  - Data inputs: connected to 0 or 1 based on the truth table

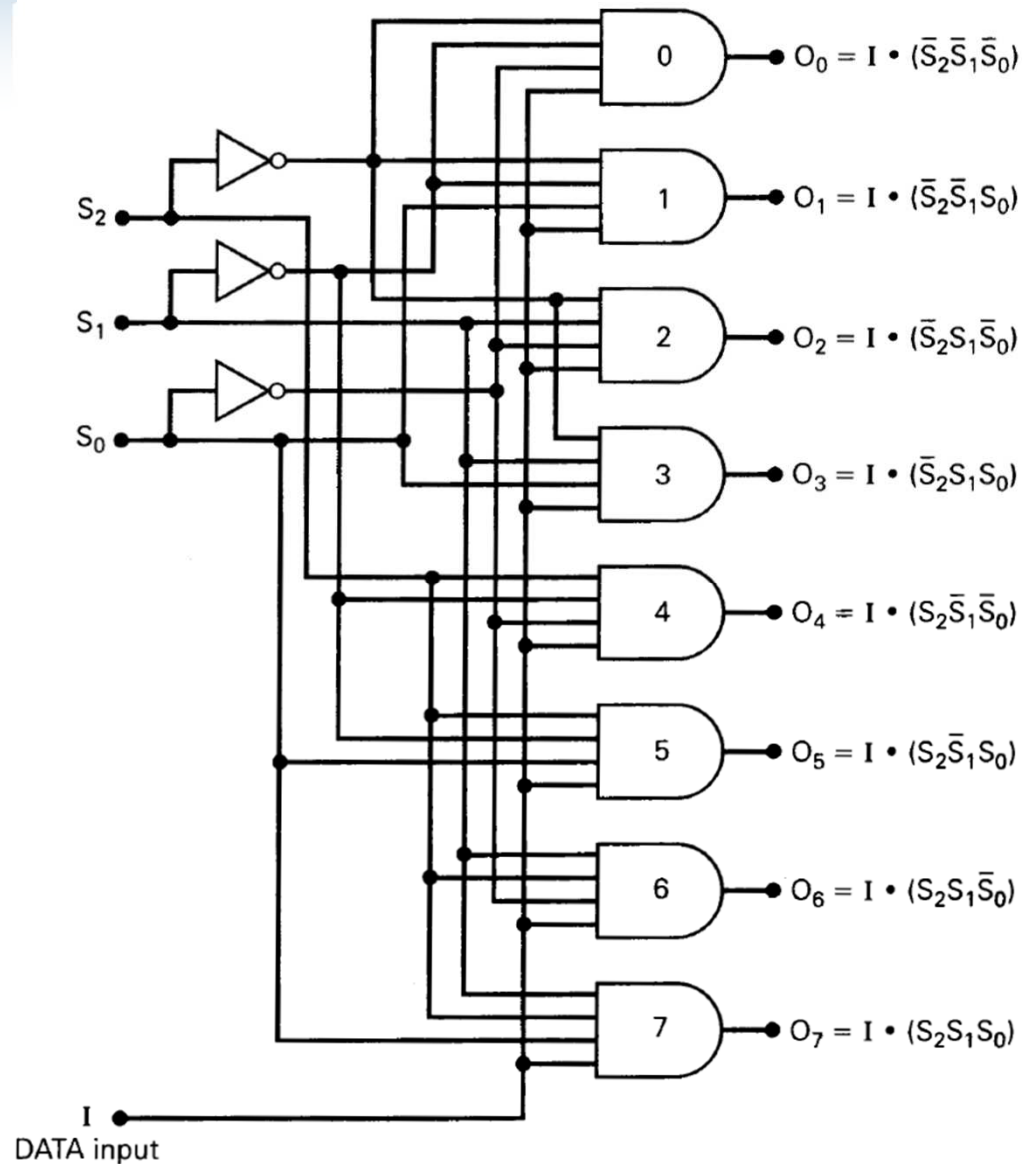


# Demultiplexers (Data Distributors)

- **Reverse operations** with multiplexer
- Like a multi-position switch



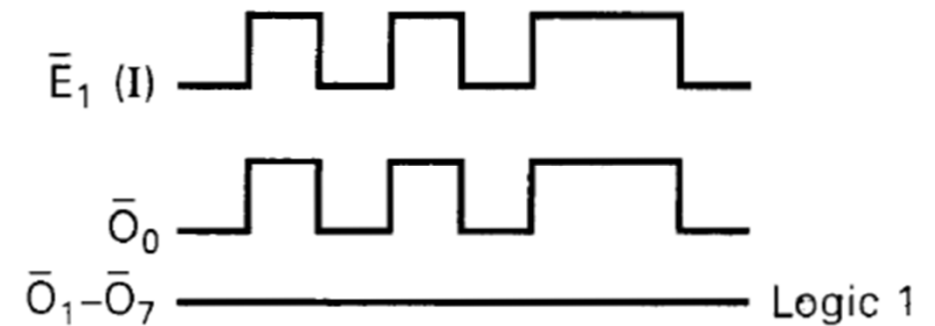
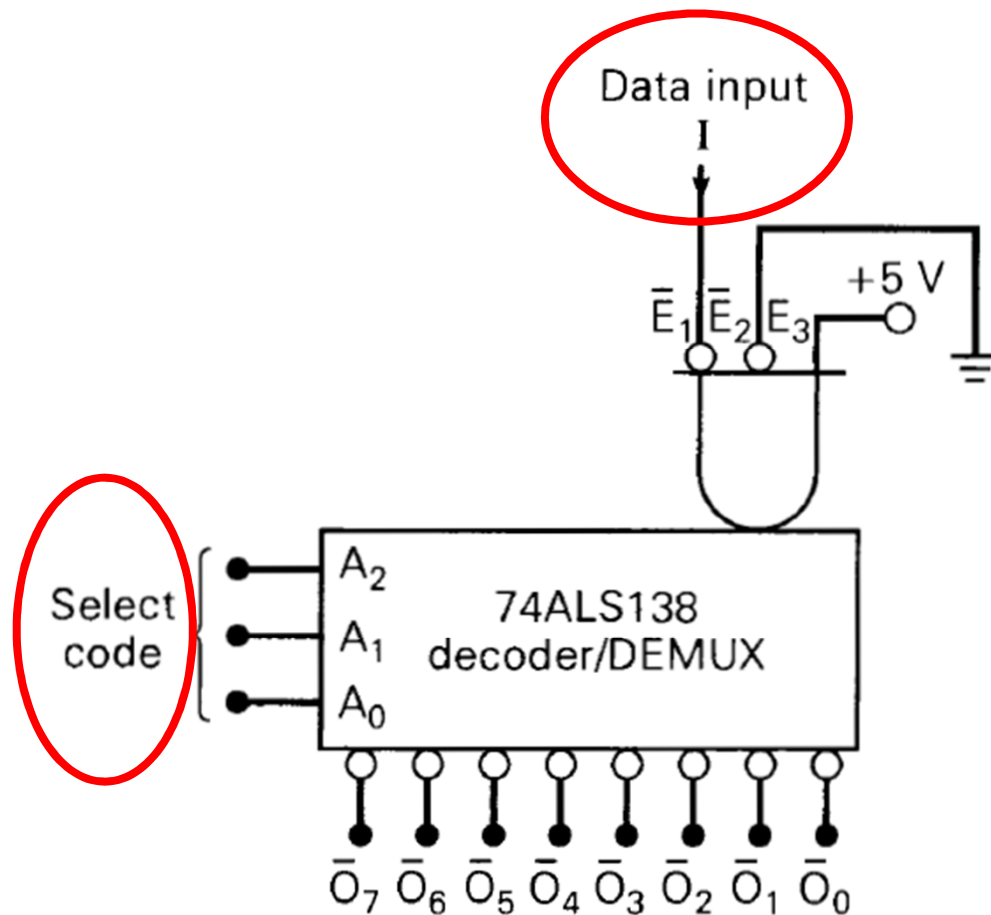
# 1-Line-to-8-Line Demultiplexer



SELECT code			OUTPUTS							
$S_2$	$S_1$	$S_0$	$O_7$	$O_6$	$O_5$	$O_4$	$O_3$	$O_2$	$O_1$	$O_0$
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

I: Data input

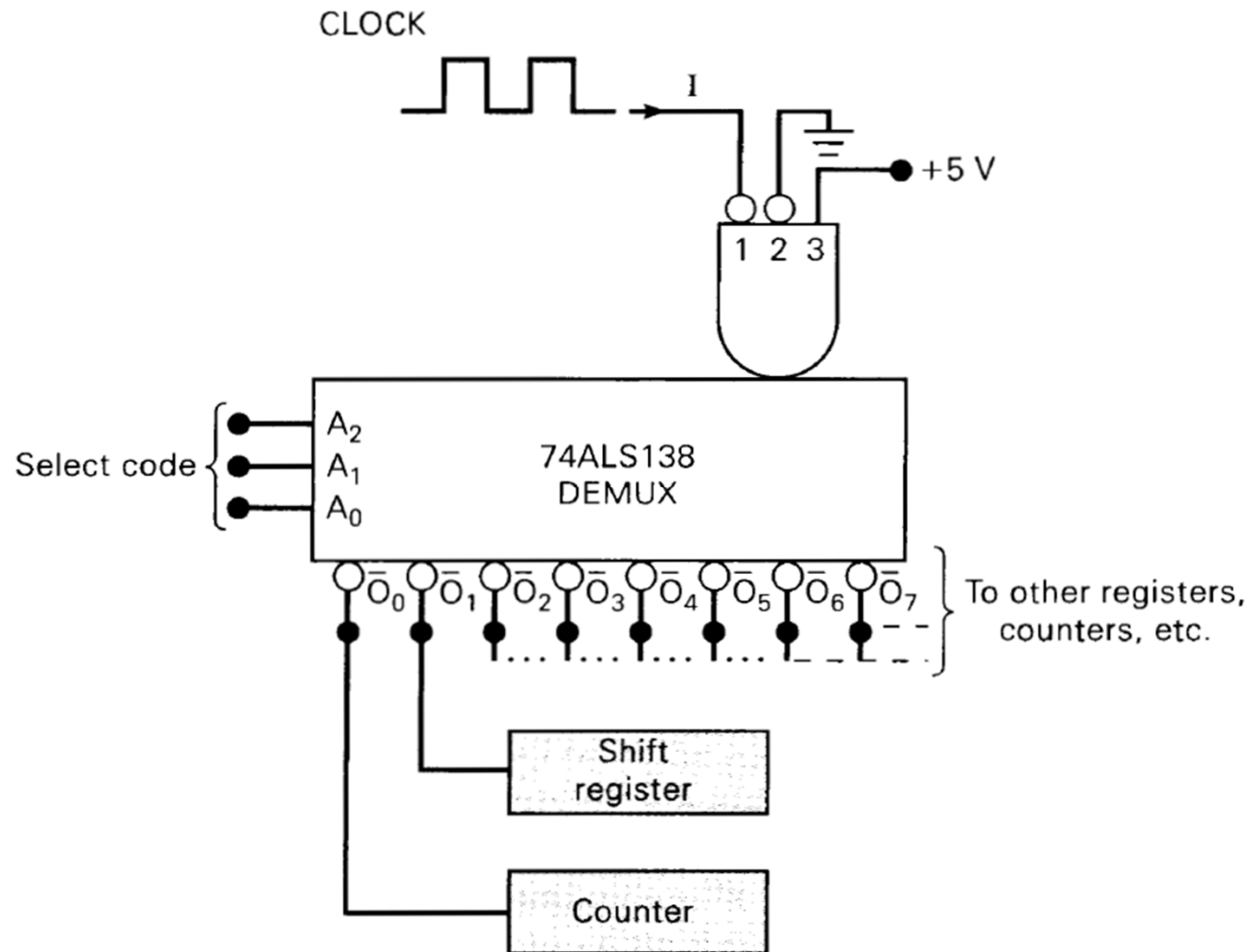
# DEMUX vs. 1-to-N Decoder



Waveforms for  $A_2A_1A_0 = 000$

# Application (1)

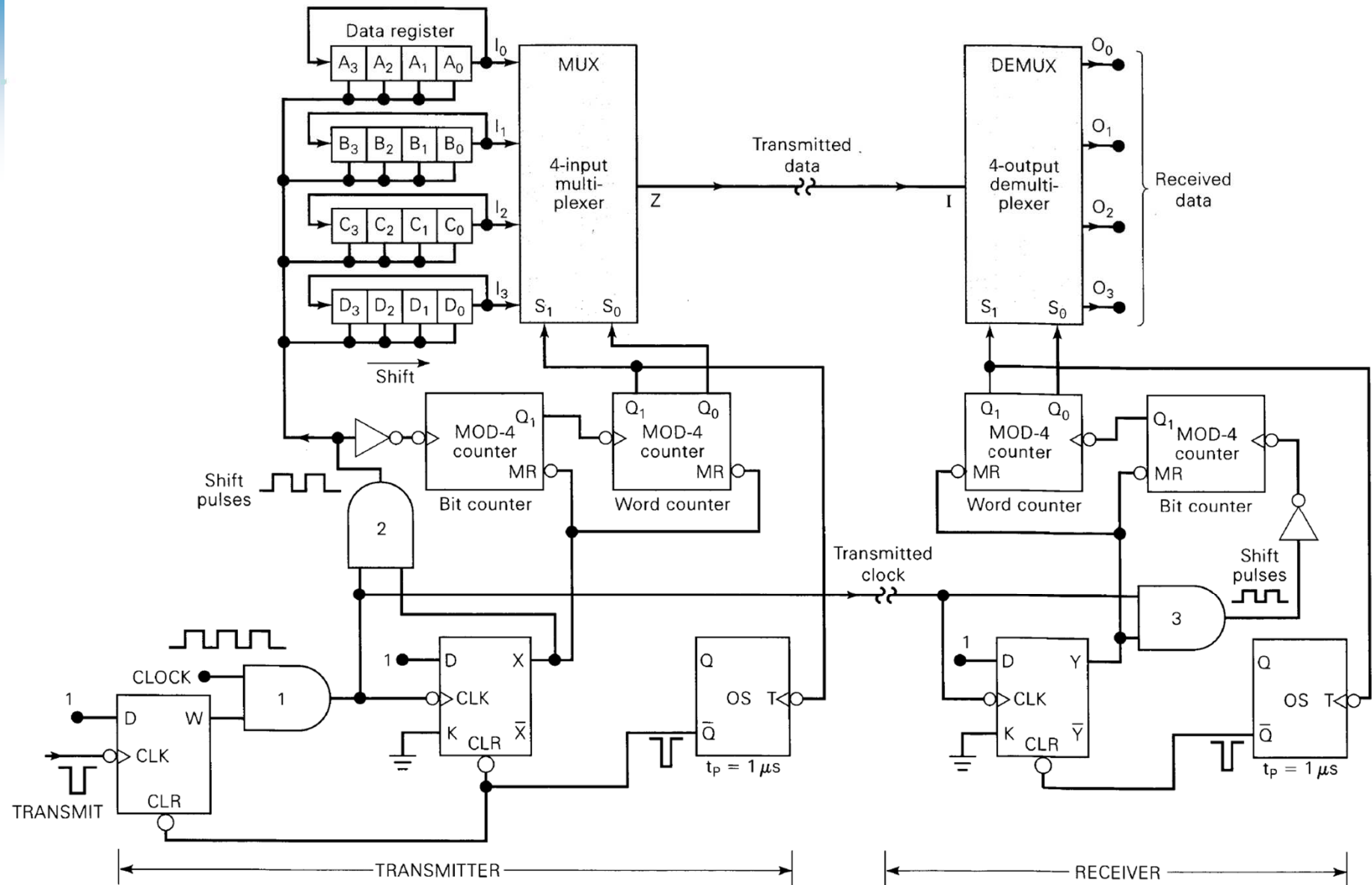
- **Clock Demultiplexer**



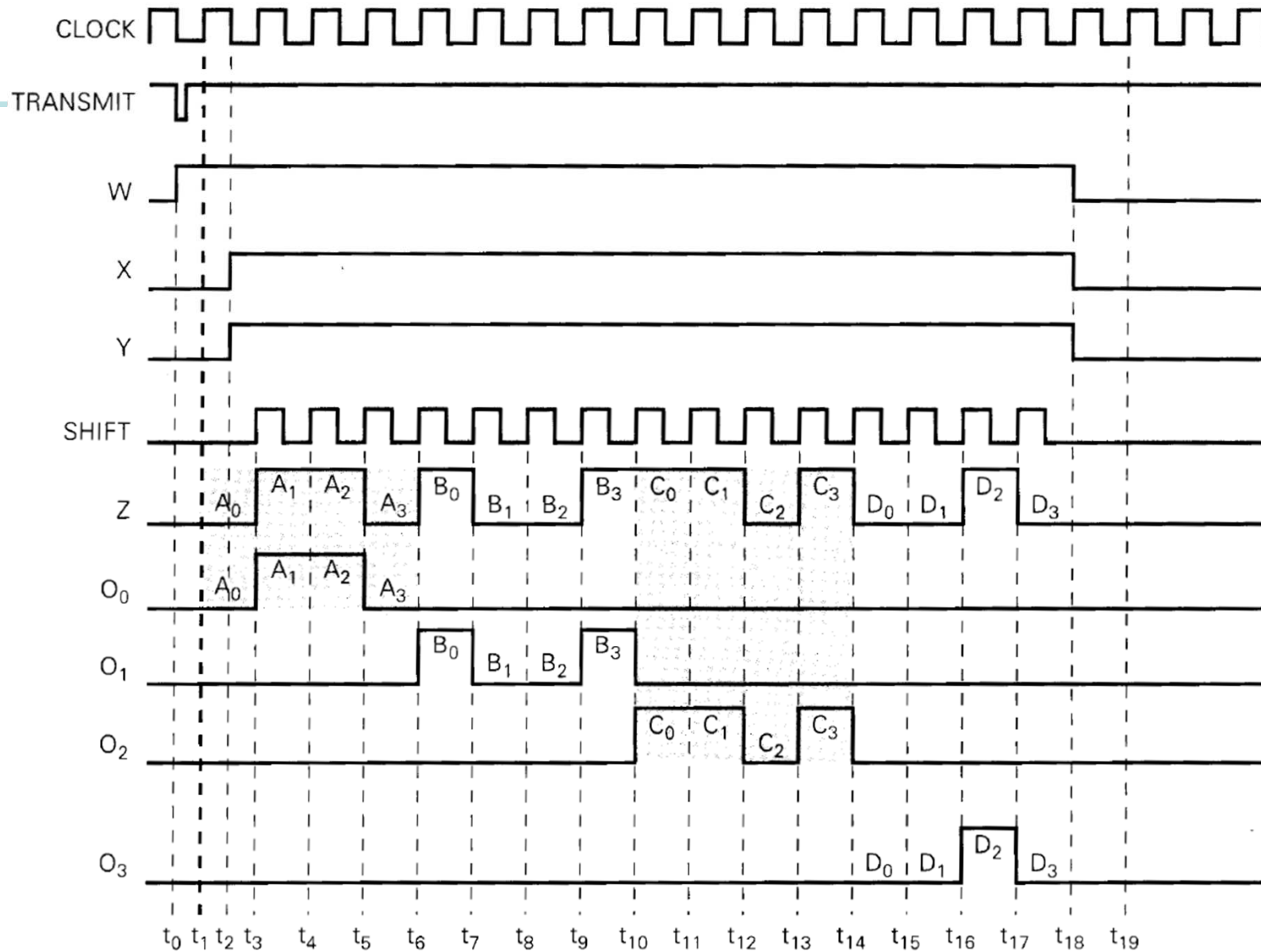
## Application (2)

- Synchronous Data Transmission System
  - Used to serially transmit  $M$   $n$ -bit data words from a transmitter to a remote receiver
  - LSB of each register is connected as a data input to the  $M$ -input multiplexer
    - The data are said to be *time-division-multiplexed*
  - Receiver: contains a 1-to- $M$  demultiplexer





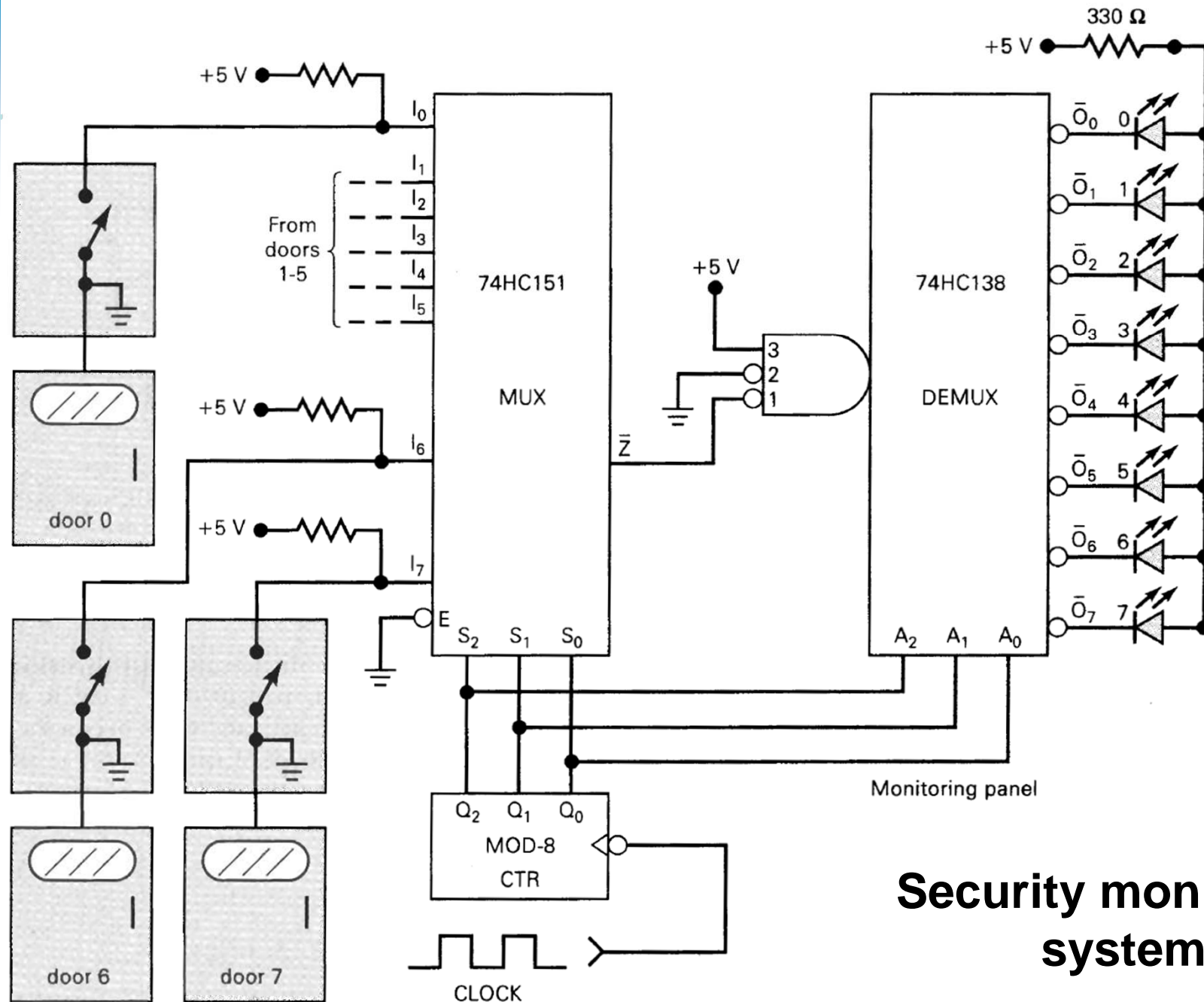
## Synchronous data transmission system



**Waveforms during one complete transmission cycle**

## Application (3)

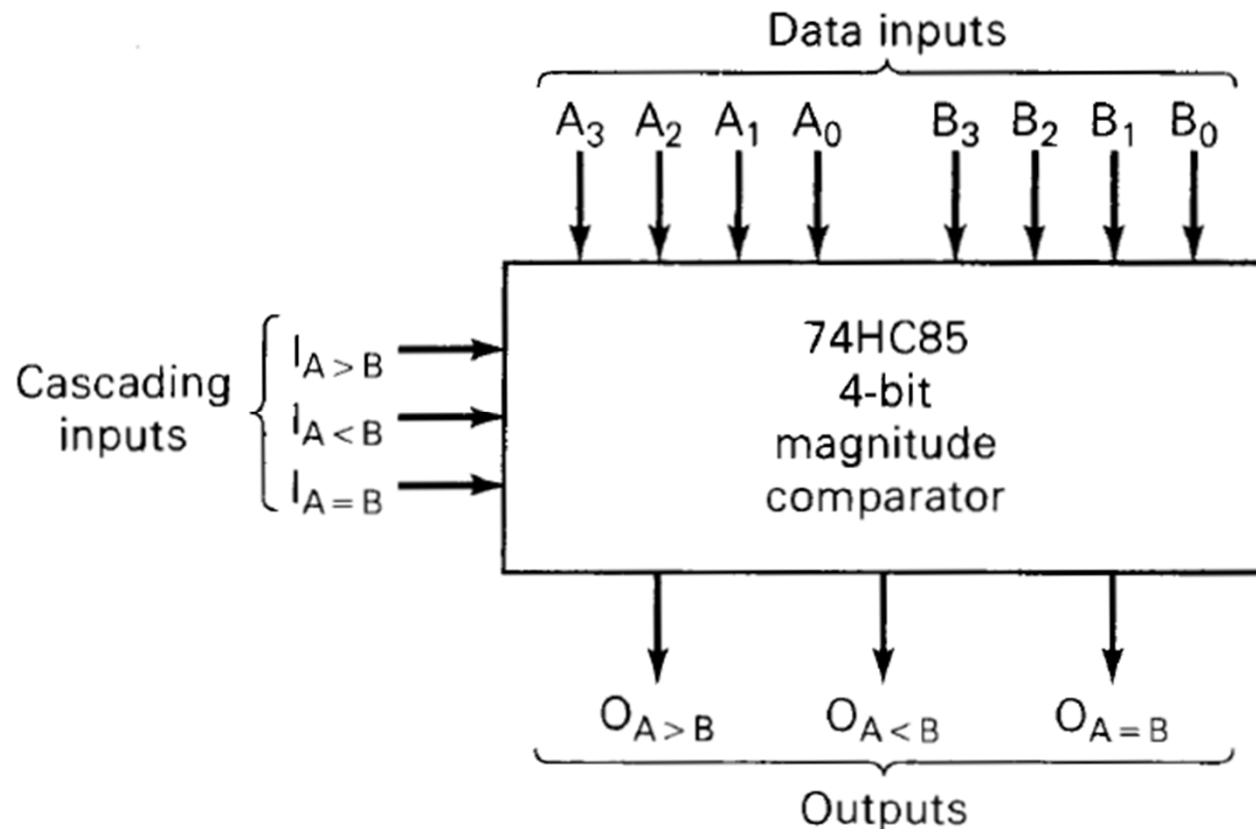
- **Security Monitoring System:** open/close status of many access doors is to be monitored.
  - Each door controls state of a switch
  - To reduce the amount of wiring to the monitoring panel → use a multiplexer/demultiplexer combination
  - State of each switch is displayed on LEDs



## Security monitoring system

# Magnitude Comparator

- Compare two input binary quantities
- Generate outputs to indicate *which one is greater*
- 74HC85



# Magnitude Comparator

TRUTH TABLE

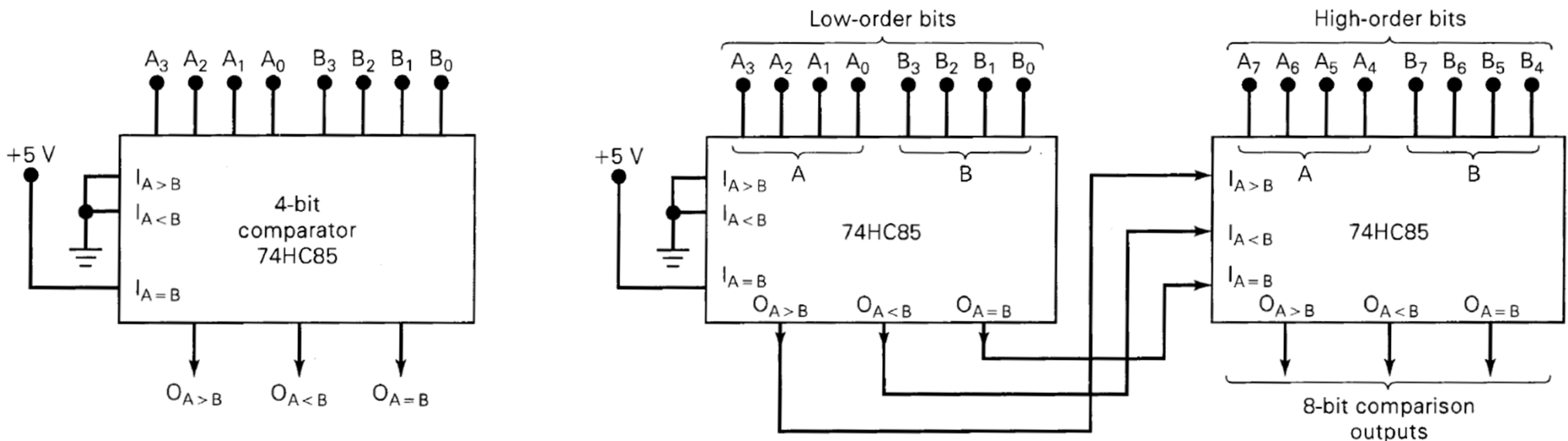
COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
$A_3, B_3$	$A_2, B_2$	$A_1, B_1$	$A_0, B_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$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

# Magnitude Comparator

- Cascading Inputs
  - Provide a means for expanding the comparison operation to more than 4 bits by cascading 2 or more 4-bit comparators



# Exercise

- Describe operation of the 8-bit comparator for the following cases

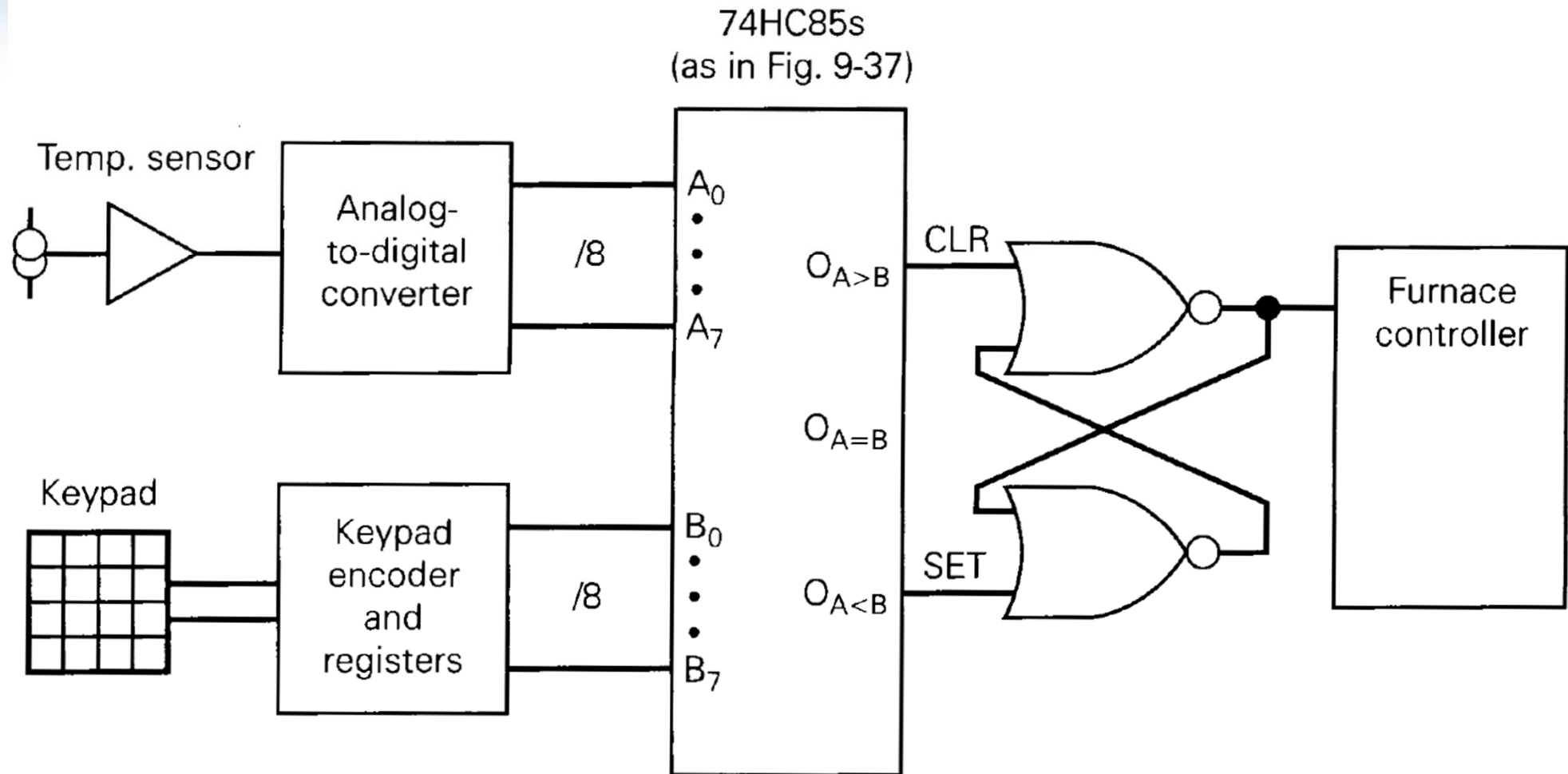
$$\begin{array}{lcl} - A_7A_6A_5A_4A_3A_2A_1A_0 & = & 10101111 \\ B_7B_6B_5B_4B_3B_2B_1B_0 & = & 10110001 \end{array}$$

$$\begin{array}{lcl} - A_7A_6A_5A_4A_3A_2A_1A_0 & = & 10101111 \\ B_7B_6B_5B_4B_3B_2B_1B_0 & = & 10101111 \end{array}$$

$$\begin{array}{lcl} - A_7A_6A_5A_4A_3A_2A_1A_0 & = & 10101111 \\ B_7B_6B_5B_4B_3B_2B_1B_0 & = & 10101001 \end{array}$$



# Magnitude Comparator - Application



**Magnitude comparator used in a digital thermostat**

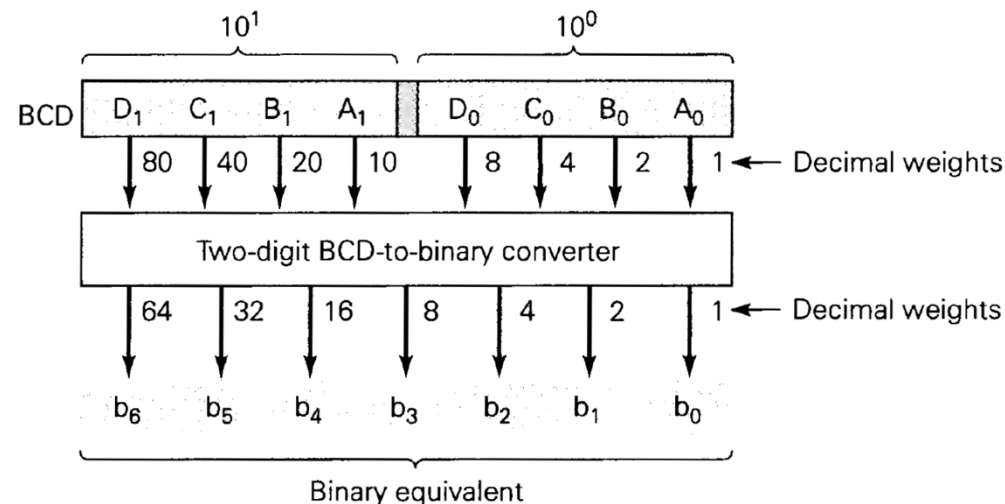
# Code Converters

- Change data presented in one type of binary code to another type of binary code
  - BCD-to-7-segment decoder/driver is a code converter
  - Some of the more common code conversion
    - BCD to 7-segment
    - BCD to binary
    - Binary to BCD
    - Binary to Gray code
    - Gray code to binary
    - ASCII to EBCDIC
    - EBCDIC to ASCII

*(EBCDIC is an alphanumeric code developed by IBM similar to ASCII)*

# BCD-to-Binary Converter

- Basic idea
  - Inputs to the converter are the two 4-bit code groups of the decimal value
    - $D_0C_0B_0A_0$ : represent the  $10^0$  or units digit
    - $D_1C_1B_1A_1$ : represent the  $10^1$  or tens digit
  - Outputs:  $b_6b_5b_4b_3b_2b_1b_0$  – 7 bits of the binary equivalent of the same decimal value



# BCD-to-Binary Converter

- BCD-to-binary converter can be accomplished with either hardware or software
  - Hardware method: generally faster but require extra circuit
  - Software method: no extra circuit but take more time (step by step)
  - The method chosen in a particular application depends on whether or not conversion time is an important consideration

# BCD-to-Binary Converter

- Conversion Process
  - The bits have decimal weights which are 8, 4, 2, 1 but differ by a factor of 10 from one code group to the next
  - Decimal weight of each bit can be converted to its binary equivalent
  - *Compute the binary sum of the binary equivalents of all bits in the BCD representation that are 1s*

# BCD-to-Binary Converter

- Example: Convert 01010010 ( $52_{\text{BCD}}$ ) to binary

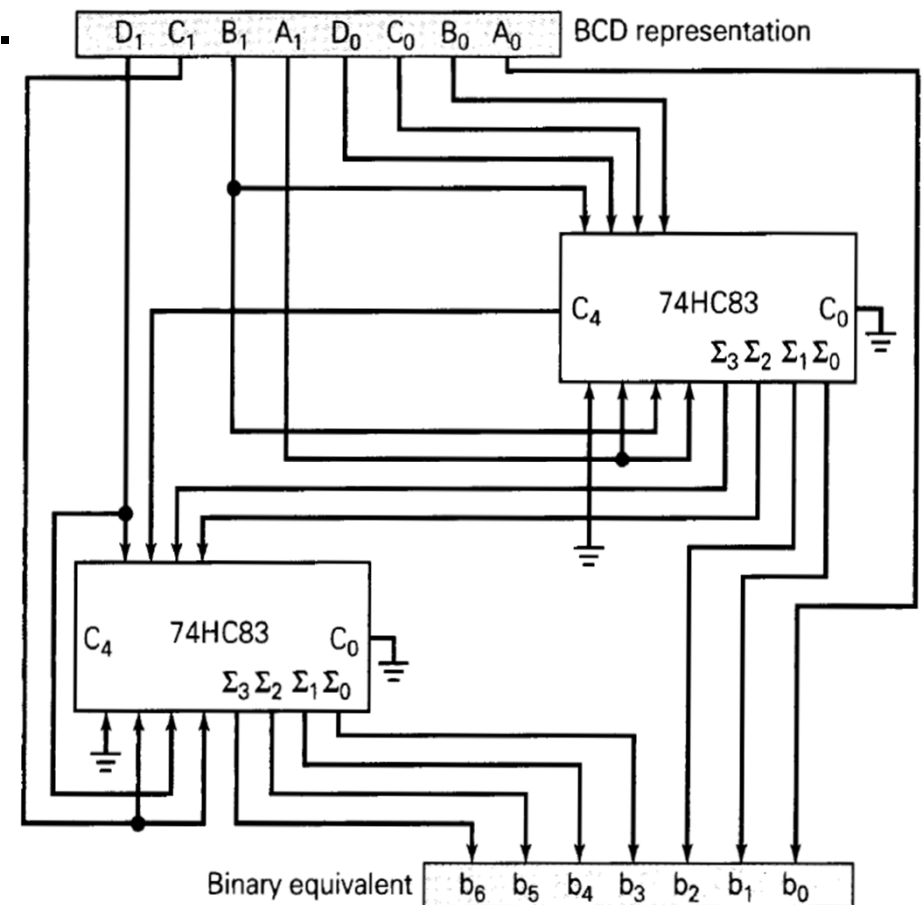
BCD Bit	Decimal Weight	Binary Equivalent						
		$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
$A_0$	1	0	0	0	0	0	0	1
$B_0$	2	0	0	0	0	0	1	0
$C_0$	4	0	0	0	0	1	0	0
$D_0$	8	0	0	0	1	0	0	0
$A_1$	10	0	0	0	1	0	1	0
$B_1$	20	0	0	1	0	1	0	0
$C_1$	40	0	1	0	1	0	0	0
$D_1$	80	1	0	1	0	0	0	0

0 1 0 1 0 0 1 0 (BCD)  
     └──────────┘→ 0000010 (binary for 2)  
     └──────────┘→ 0001010 (binary for 10)  
     └──────────┘→ + 0101000 (binary for 40)  
                     0110100 (binary for 52)

# BCD-to-Binary Converter

- Circuit Implementation: use binary adder circuits
  - A0: directly to b0
  - B0, A1 contribute to b1, ...

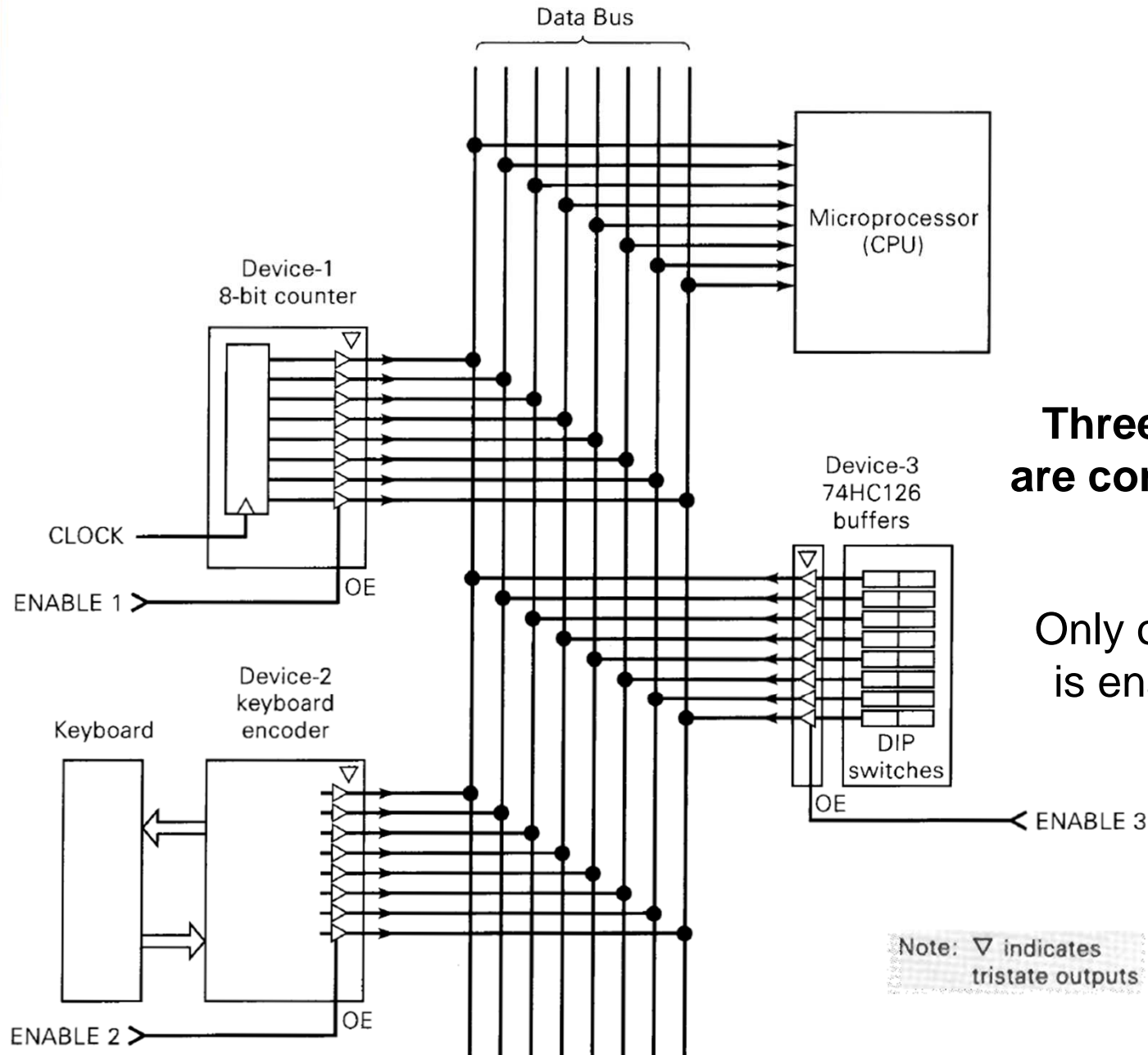
BCD Bit	Decimal Weight	Binary Equivalent						
		$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
$A_0$	1	0	0	0	0	0	0	1
$B_0$	2	0	0	0	0	0	1	0
$C_0$	4	0	0	0	0	1	0	0
$D_0$	8	0	0	0	1	0	0	0
$A_1$	10	0	0	0	1	0	1	0
$B_1$	20	0	0	1	0	1	0	0
$C_1$	40	0	1	0	1	0	0	0
$D_1$	80	1	0	1	0	0	0	0



# Data Busing

- **Data bus:** a common set of connecting lines for the transfer of data
- Many different devices can have their outputs and inputs tied to the common data bus lines
- Devices tied to the data bus will often have *tristate outputs* or tied to the data bus through *tristate buffers*
- Some of the devices that are commonly connected to a data bus:
  - Microprocessor, semiconductor memory chips, Digital-to-analog/Analog-to-digital converters (DACs/ADCs)





Three different devices  
are connected to data bus

Only one device at a time  
is enabled to avoid **bus  
contention**

# 74ALS173/HC173 Tristate Register

- Tristate Register allow outputs of devices to be tied to a data bus

Inputs					FF Outputs
MR	CP	$\overline{IE}_1$	$\overline{IE}_2$	$D_n$	Q
H	X	X	X	X	L
L	L	X	X	X	$Q_0$
L	$\downarrow$	H	X	X	$Q_0$
L	$\downarrow$	X	H	X	$Q_0$
L	$\downarrow$	L	L	L	L
L	$\downarrow$	L	L	H	H

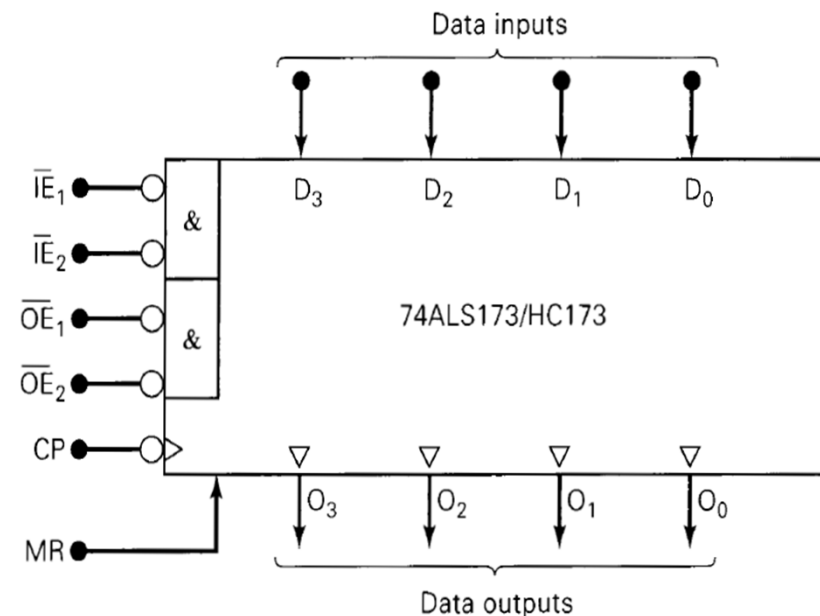
When either  $\overline{OE}_1$  or  $\overline{OE}_2$  is HIGH the output is in the OFF state (high impedance); however, this does not affect the contents or sequential operating of the register

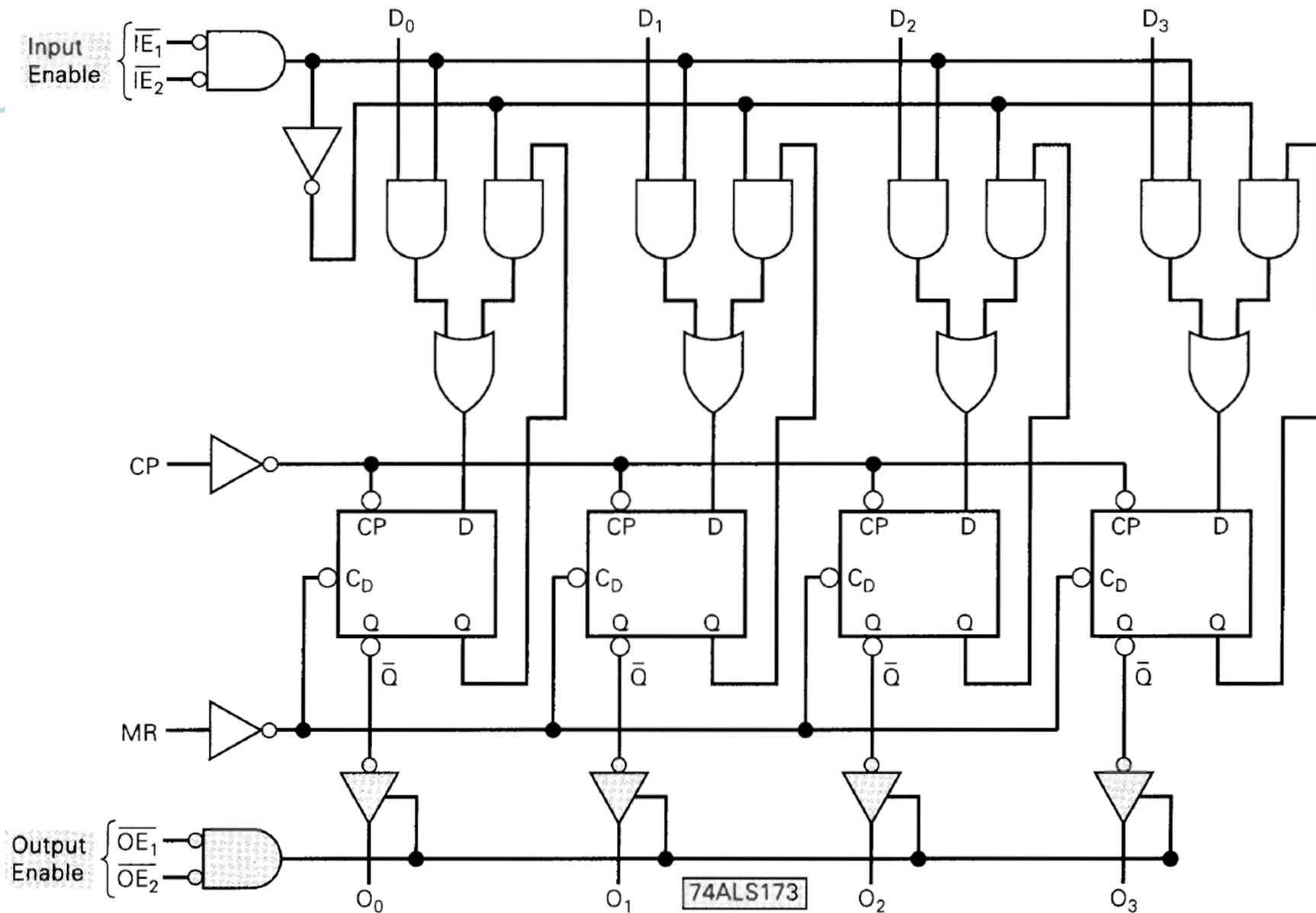
H = HIGH voltage level

L = LOW voltage level

X = immaterial

$Q_0$  = output prior to PGT

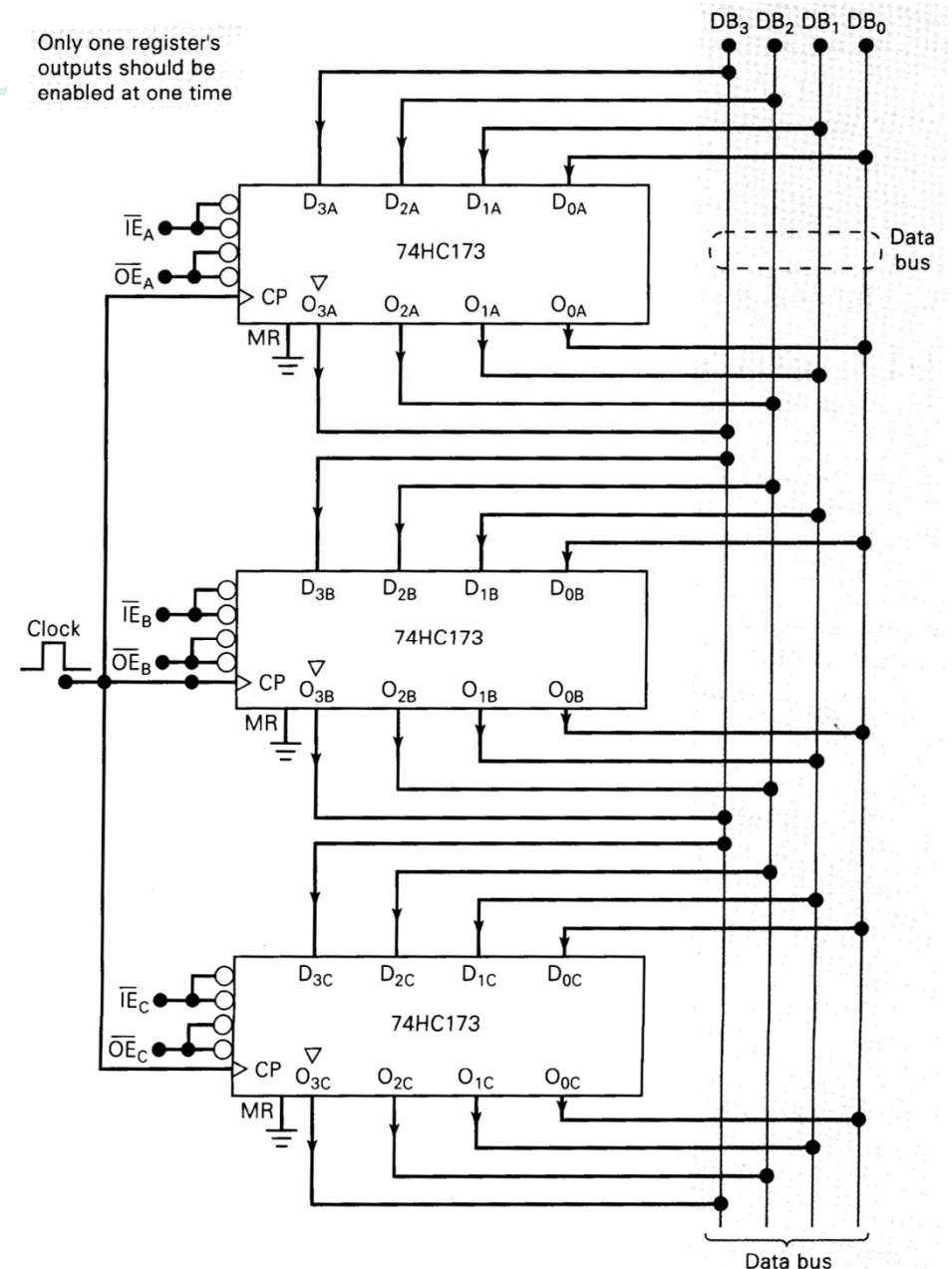




**Logic Diagram for the 74ALS173 tristate register**

# Data bus Operation

- Example: Register – register data transfer
  - Each register has its pair of OE inputs and IE inputs tied together
  - Data bus:  $DB_0 - DB_3$
  - 3 registers have their inputs and outputs connected together
  - Only one register have its output enabled, other 2 registers outputs remain in Hi-Z state

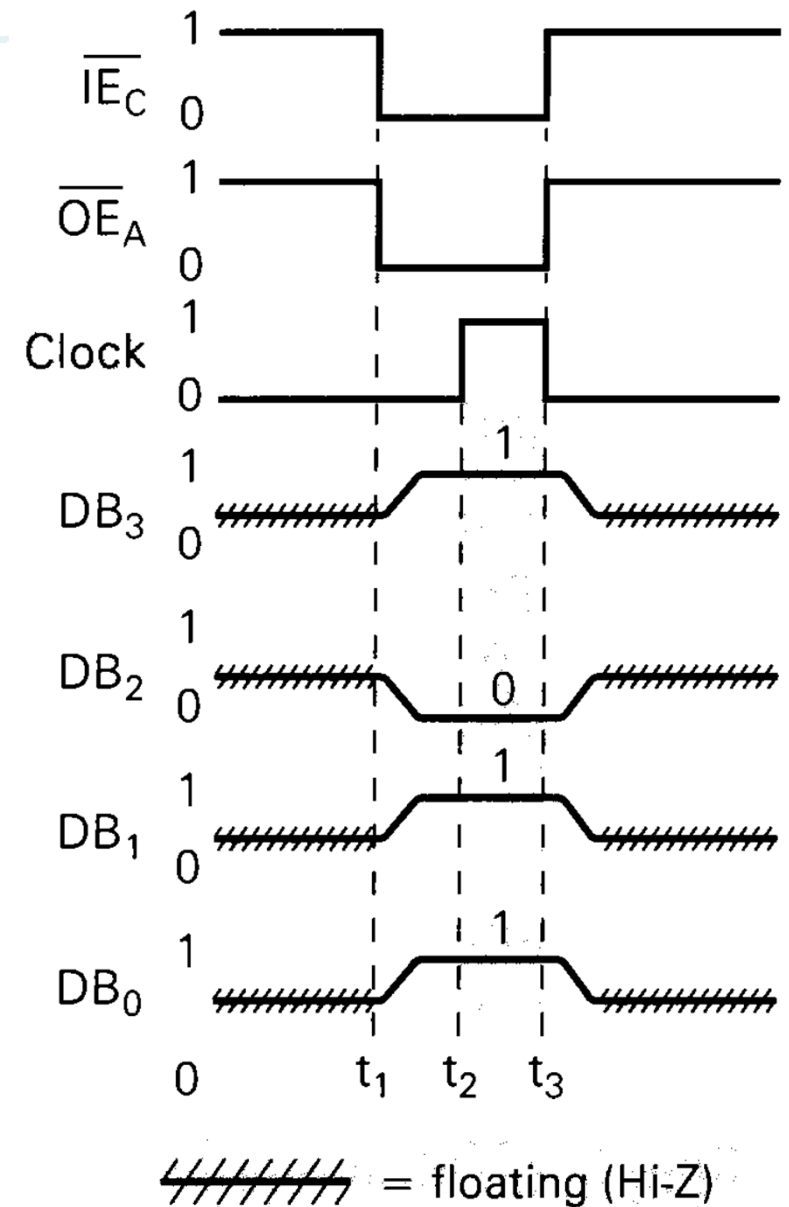


# Data bus Operation

- Data Transfer Operation
  - The control unit will generate the signals that select which register will put its data on the data bus and which one will take the data from the data bus
  - Example:
    - t1: only register A have its output enabled
$$OE_A = 0 \qquad OE_B = OE_C = 1$$
    - t2: only register C should have its inputs enabled
$$IE_C = 0 \qquad IE_A = IE_B = 1$$

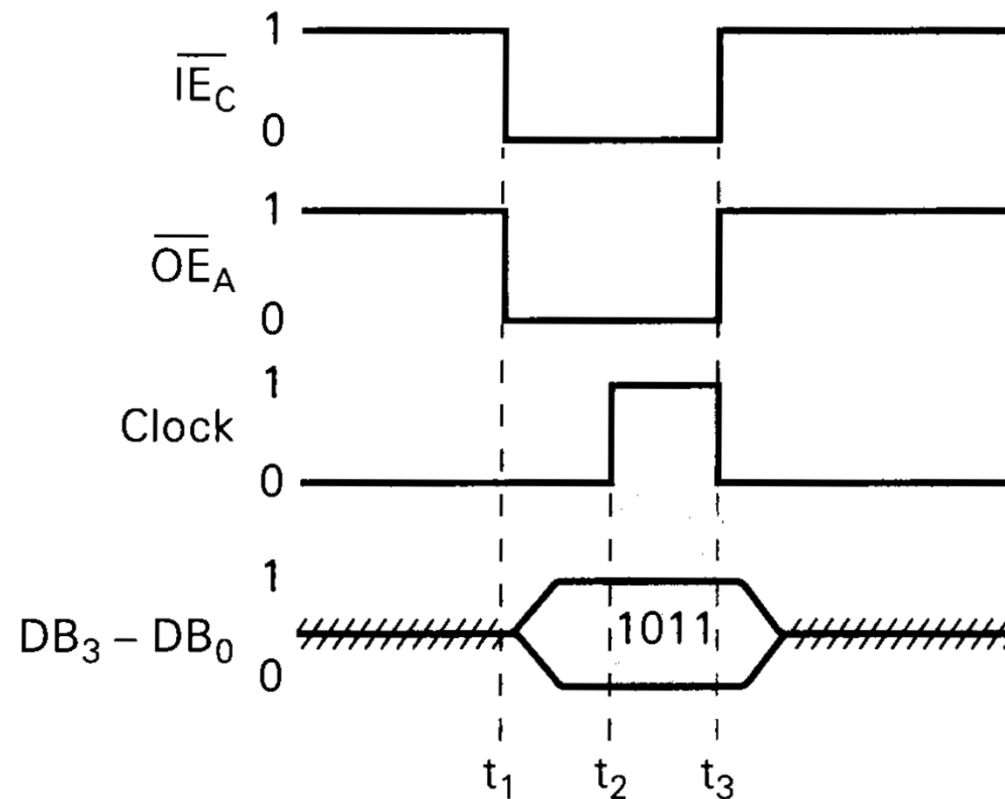
# Bus Signals

- **t1**: Register A outputs are enabled. Its data are placed on bus
- **t2**: PGT of clock transfers valid data from data bus into register C
- **t3**: Register A outputs are disabled and data bus lines return to Hi-Z state



# Simplified Bus Timing Diagram

- Use only a single timing waveform to represent the complete set of bus lines

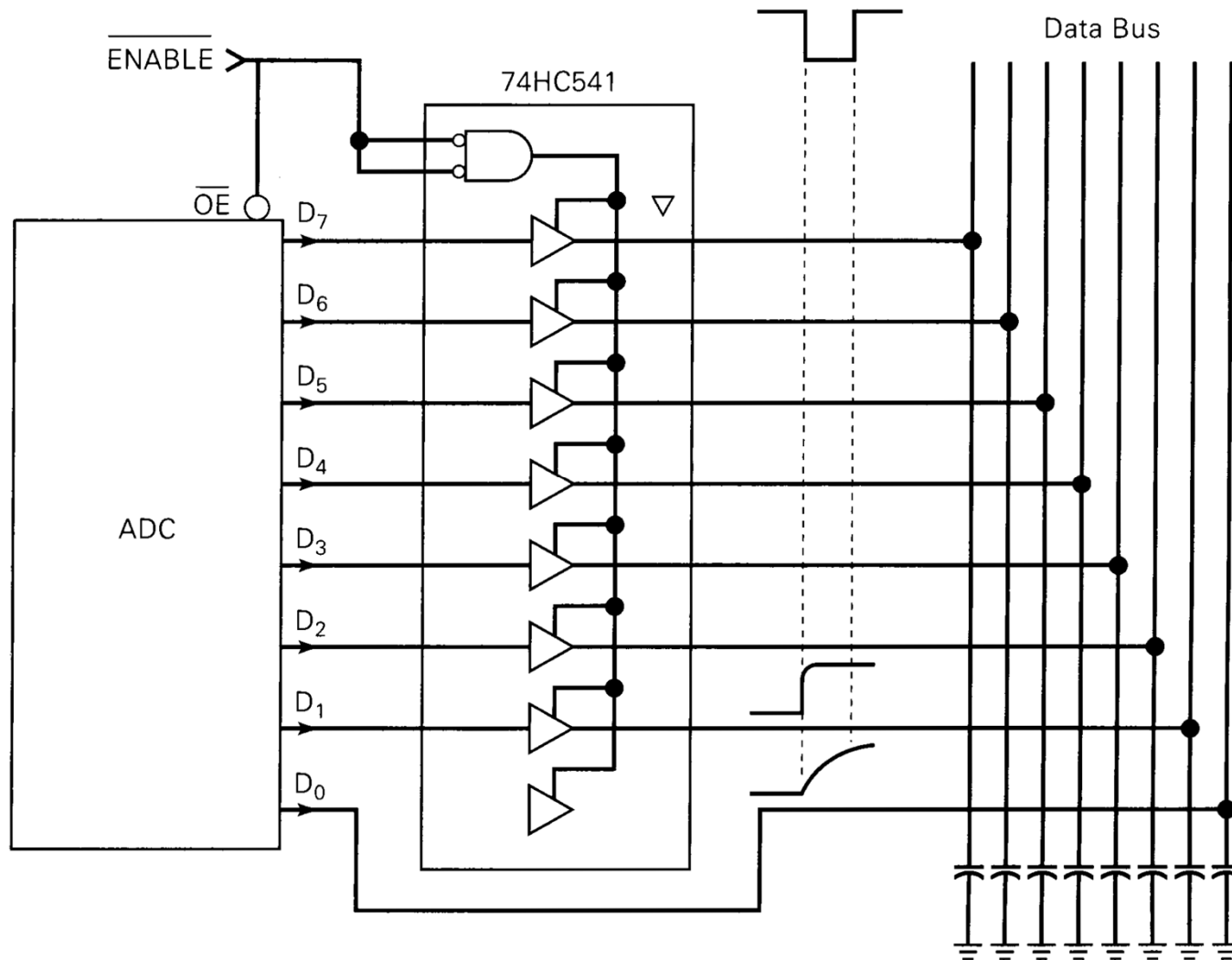


# Expanding the Bus

- The number of lines on the data bus will depend on the size of the **data word** (unit of data) that is to be transferred over the bus
  - Example: 8-bit word size → 8-line data bus
- Some devices will need to be connected to the bus through a **bus driver**
- Bus driver
  - Tristate outputs with a very low output impedance → rapidly charge and discharge the bus capacitance
  - Reduce transition time of signal



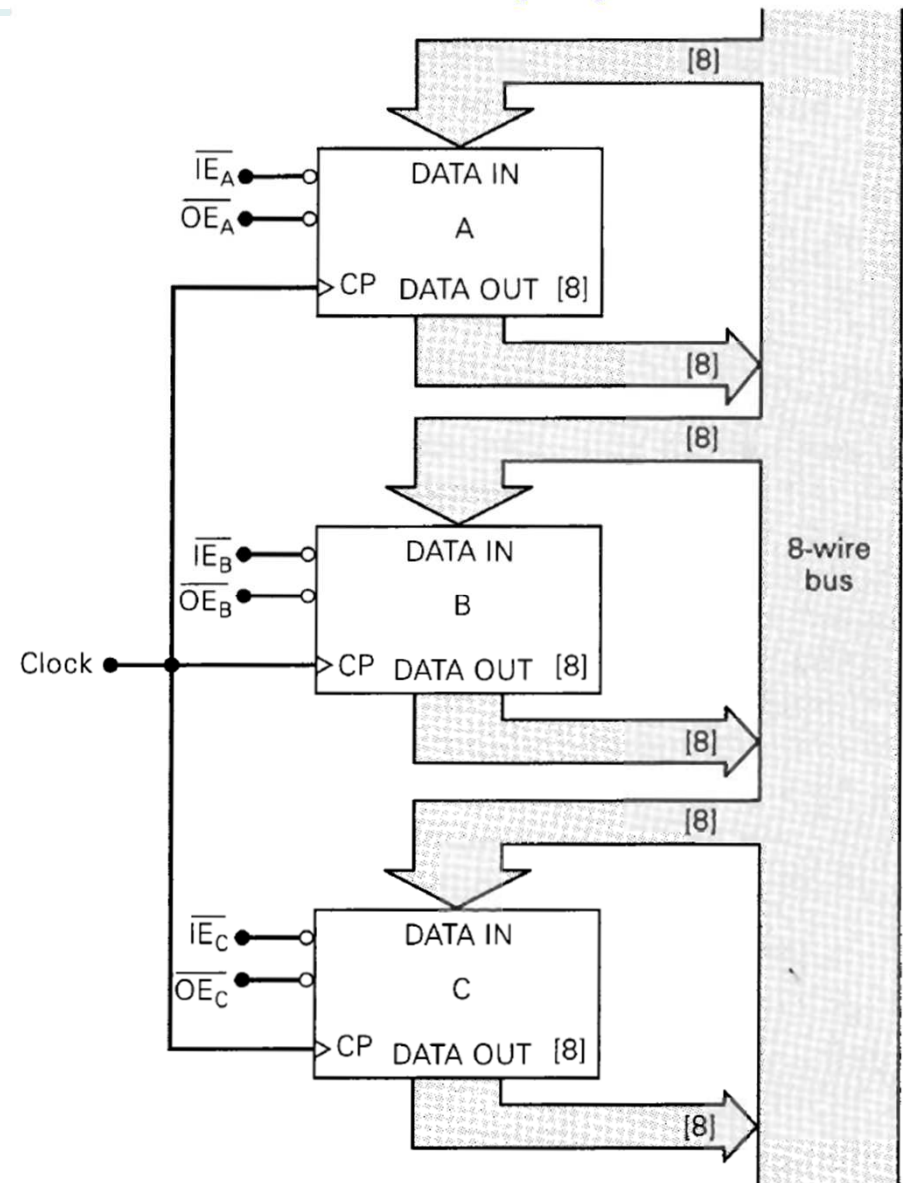
# Expanding the Bus



**74HC541 octal bus driver**

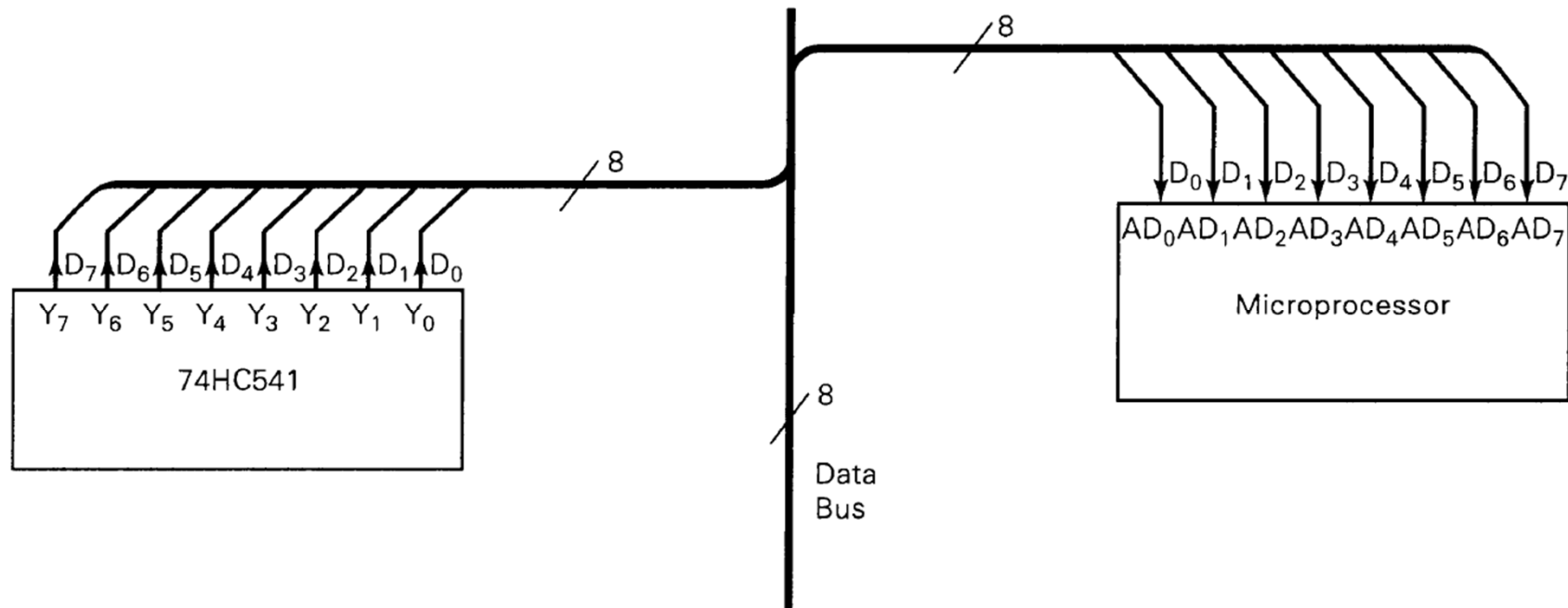
# Simplified Bus Representation (1)

- Use on block diagram and in some circuit schematics
- The connections to and from the data bus are represented by wide arrows
- The numbers in brackets ([ ]) indicate the number of bits of register and number of lines of data bus



# Simplified Bus Representation (2)

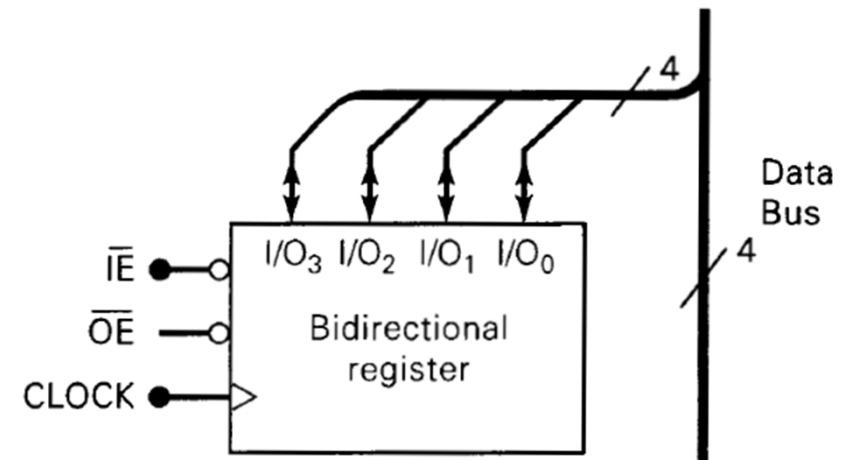
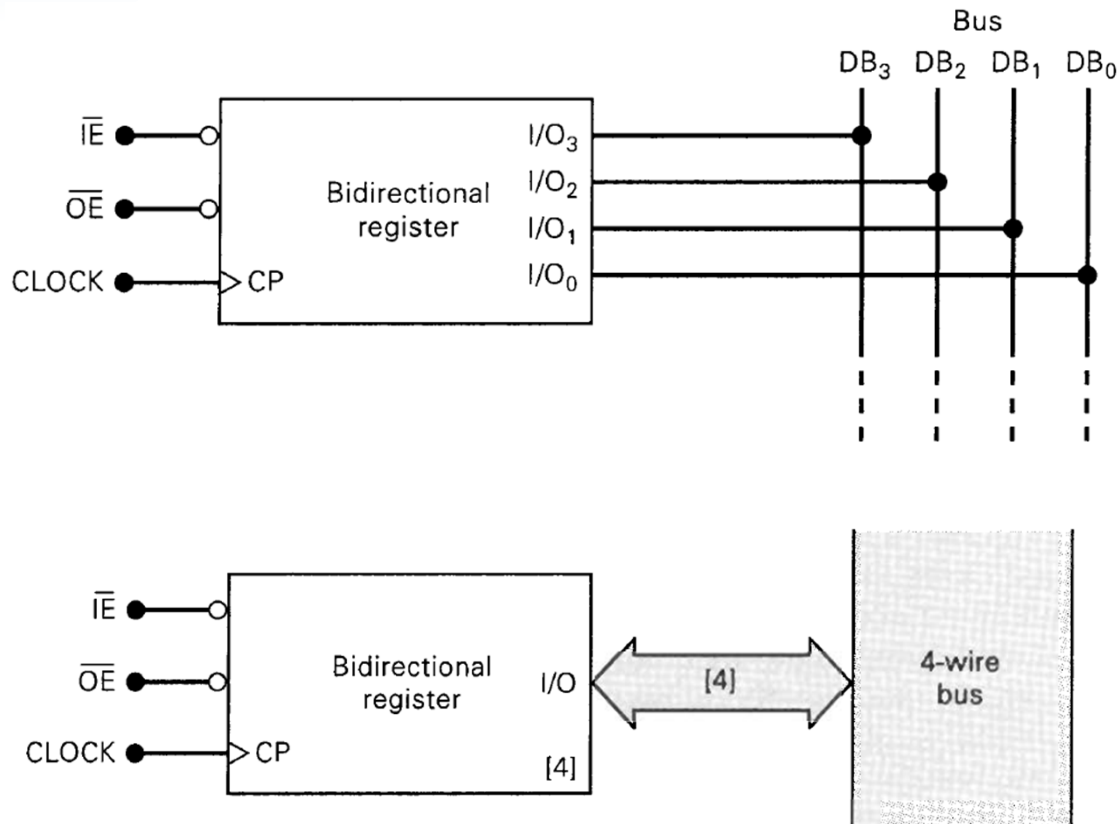
- Bundle method
  - A single line denotes the data bus
  - “/8” denotes an eight-line data bus



# Bidirectional Busing


- Some devices have both inputs and outputs connected to the data bus
  - Inputs and outputs are connected together internal to the chip
    - Reduce the number of IC pins
    - Reduce number of connections to the bus
  - Input lines ( $I_0/D_0$  to  $I_3/D_3$ ) and output lines ( $O_0$  to  $O_3$ ) have been replaced by input/output lines ( $I/O_0$  to  $I/O_3$ )
  - I/O line will function as either an input or an output depending on the state of the enable inputs
    - **Bidirectional data lines**

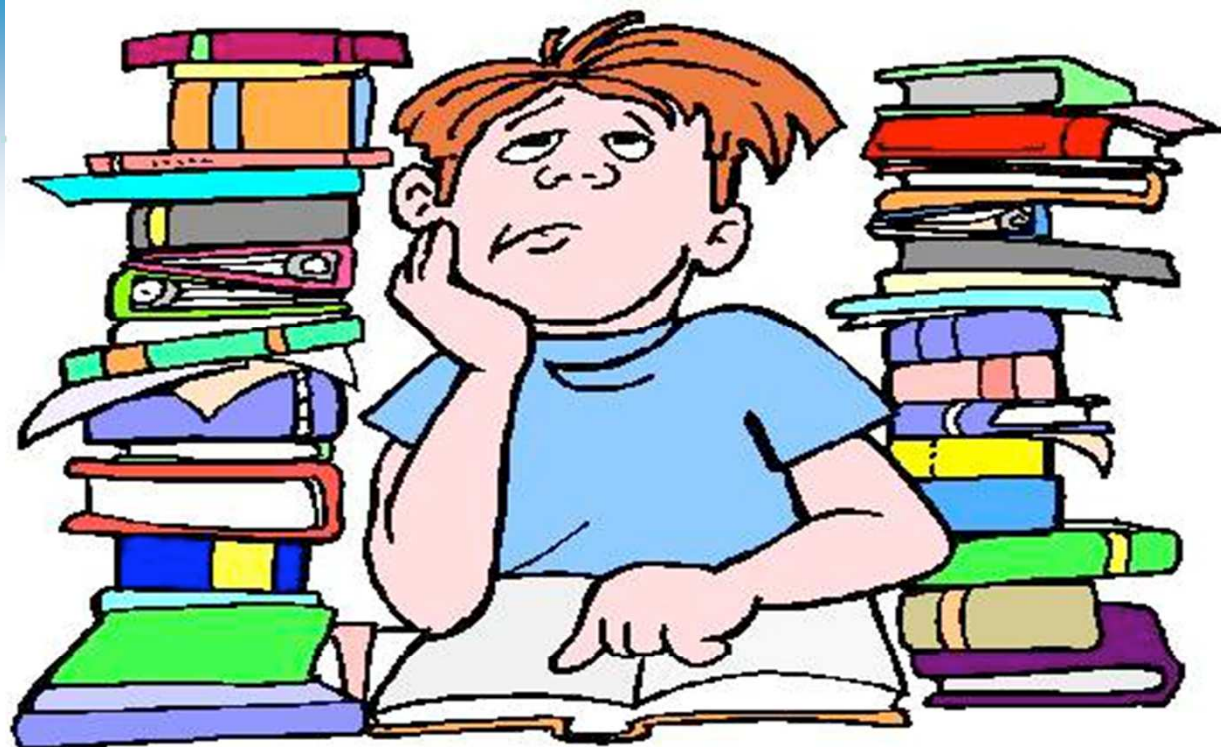
# Bidirectional Busing



**Bidirectional register connected to data bus**

# Summarize

- 
- ✓ 1-of-N encoder/decoder
  - ✓ BCD to 7-segment
  - ✓ Multiplexer/demultiplexer
  - ✓ Comparator
  - ✓ Data busing



# Review questions

- Can more than one decoder output be active at one time?
- What is the function of a decoder's enable input?
- Which LED segments will be on for a 7-segment decoder input of 1001?
- How does a priority encoder differ from an ordinary encoder?
- What are the functions of a multiplexer ?
- What are some major applications of a multiplexer?



# Reference

- Chapter 9, Digital System – Principles and Applications, Ronald J.Tocci, Neal S. Widmer