# Lecture Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Continued Multiple Output Logic Design
    - Essential prime implicants and multiple-output circuits
    - Example with both limited fan-in and multiple-output (and NAND)
  - Introduction to gate delays and timing diagrams
- Today's Lecture
  - Hazards in combinatorial logic
  - Start Multiplexers

# Handouts and Announcements

- Announcements
  - Homework Problem 8-2
    - Posted on Carmen Sunday evening (2/12)
    - Due: 11:59pm Thursday 2/16
  - Homework Reminder:
    - HW 7-4 Due: 11:59pm Tuesday 2/14
    - HW 8-1 Due: 11:59pm Thursday 2/16
  - Read for Wednesday: pages 268-271
    transistor level CMOS: 713-717(up to Fig A-8)

# Handouts and Announcements

- Announcements
  - ECE 2060 Laboratories
    - Start tomorrow for some of you, rest later this week
    - You must attend your scheduled lab session
    - There are videos you are required to view prior to attending your first lab session
    - Refer to the Carmen pages for your lab section for details
    - Lab questions should be referred to Prof. Chapman and/or the GTA for your lab section

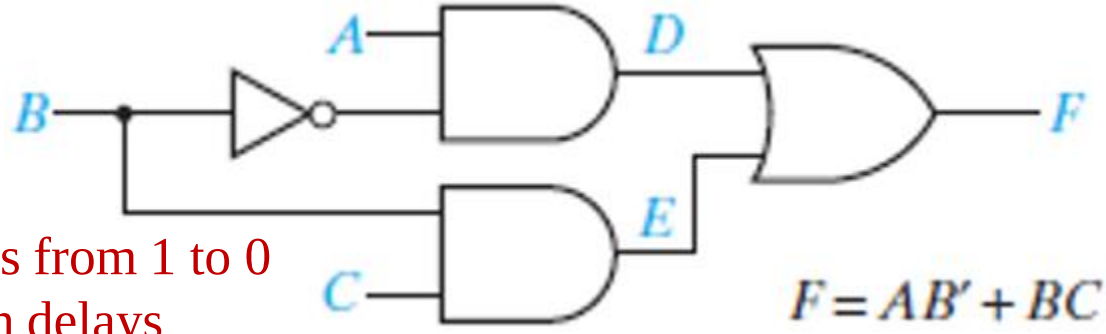
# Hazards in Combinational Logic

- Switching transients occur when different paths from input to output have different propagation delays
  - A circuit output may momentarily go to 0 when it should remain a constant 1, we say that the circuit has a static 1-hazard
  - If the output may momentarily go to 1 when it should remain a 0, we say that the circuit has a static 0-hazard
  - If, when the output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times, we say that the circuit has a dynamic hazard
    - In all 3 cases, final steady-state outputs are correct
    - Transient errors before steady-state is reached

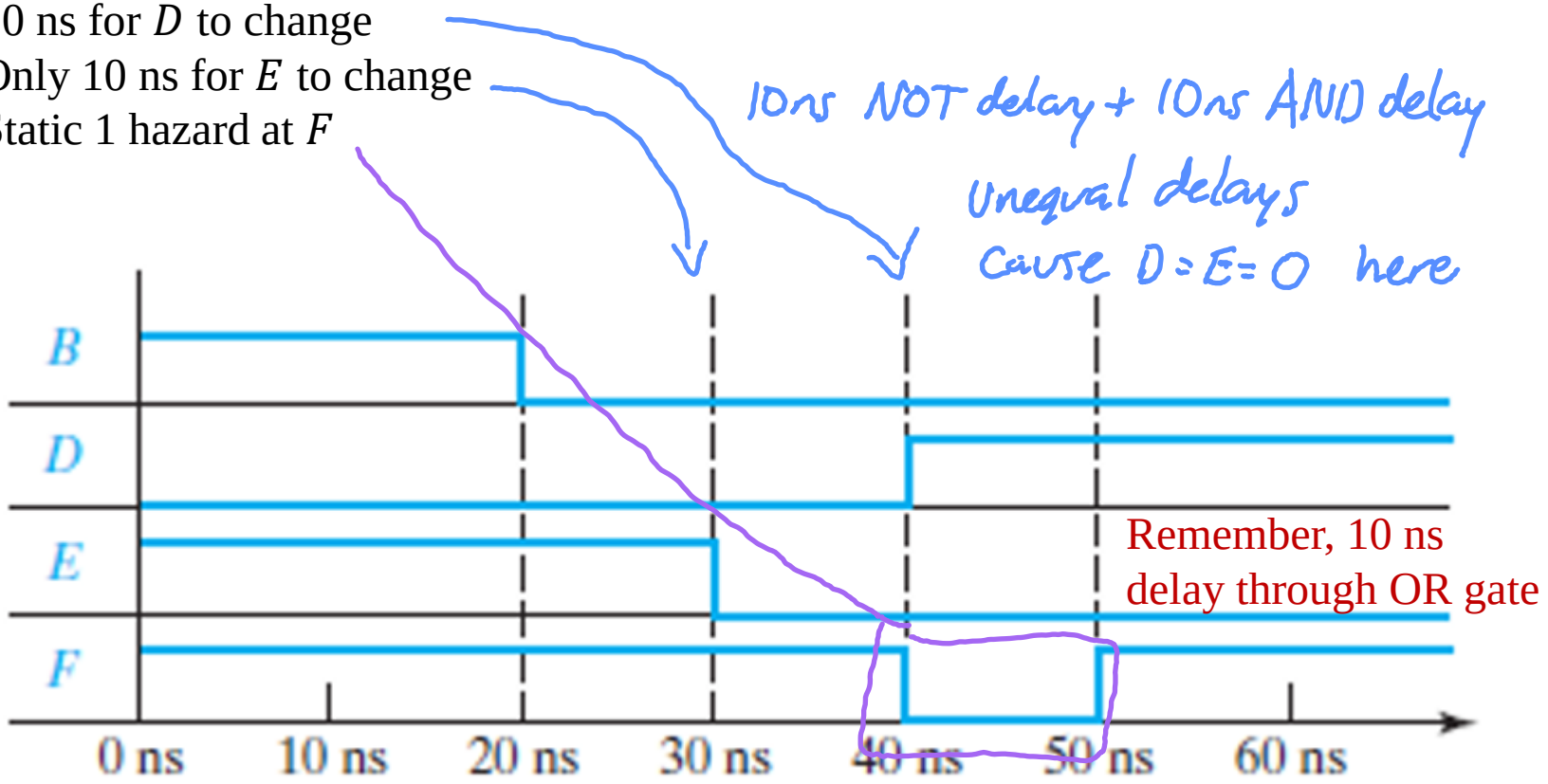

(a) Static 1-hazard (b) Static 0-hazard (c) Dynamic hazards

# Hazards in Combinational Logic

## Static 1 hazard

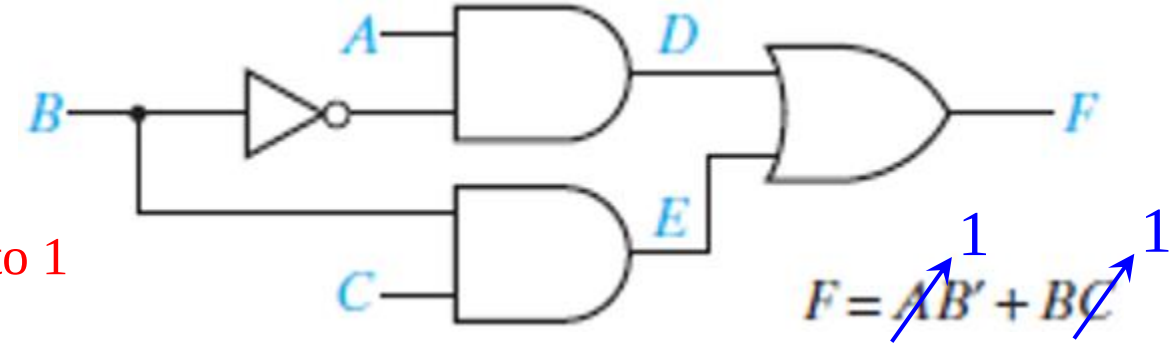



- Let $A = C = 1$
- Then $F = B' + B = 1$
- $F$ should remain 1 if $B$ changes from 1 to 0
- But consider 10 ns propagation delays
  - 20 ns for $D$ to change
  - Only 10 ns for $E$ to change
  - Static 1 hazard at $F$

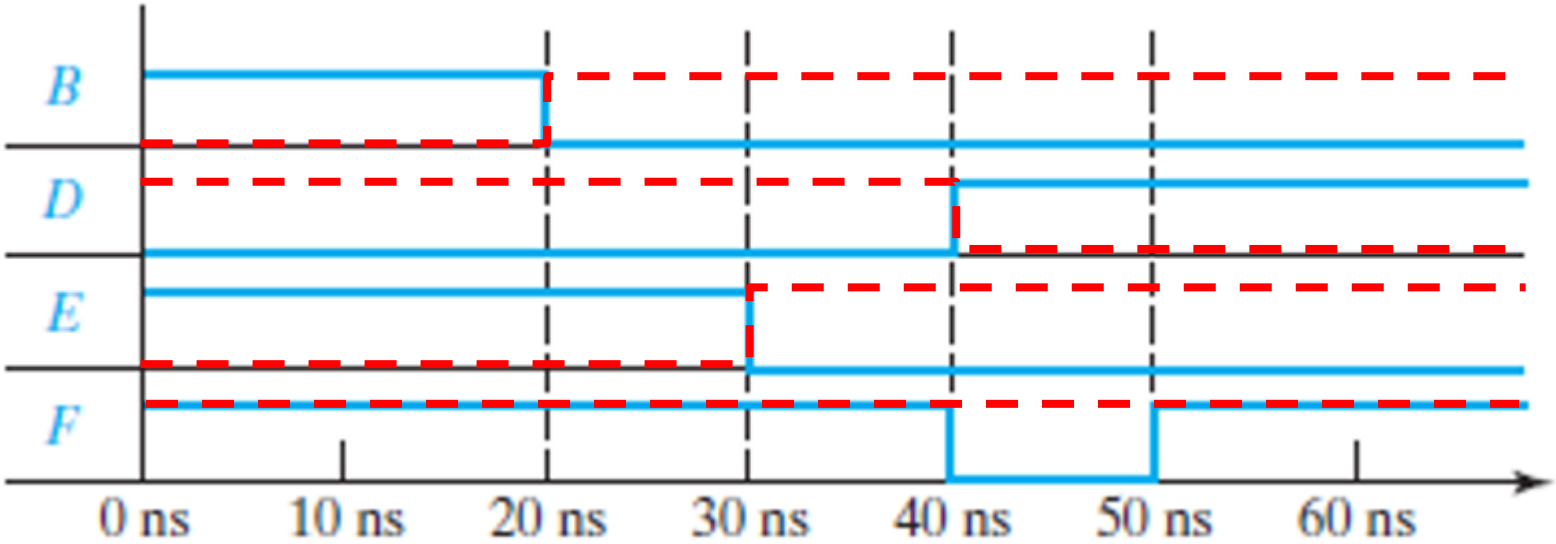


# Hazards in Combinational Logic

## Static 1 hazard



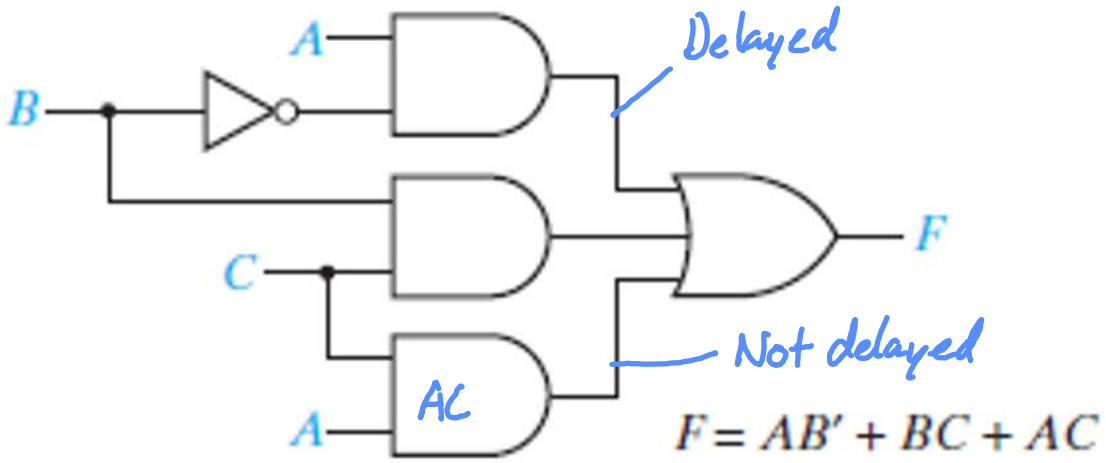

- Let $A = C = 1$
- Then $F = B' + B = 1$
- Note that changing $B$ from 0 to 1 doesn't cause a similar glitch
  - In this case the timing difference causes an overlap of when D and E = 1
  - Since they are inputs to an OR gate output stays 1
- No glitch for one direction of change doesn't imply no glitch for the other direction of change

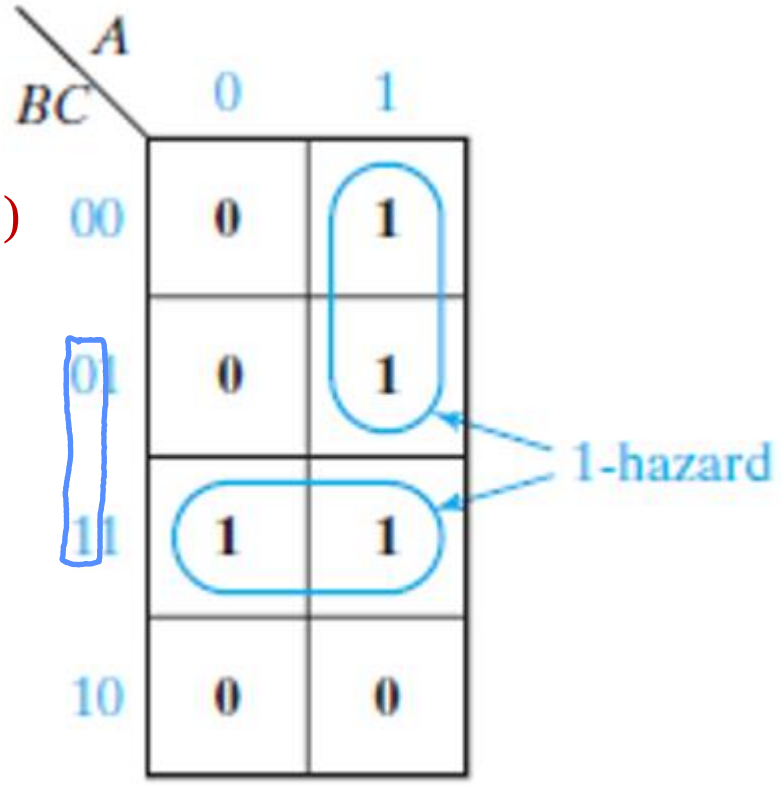


# Hazards in Combinational Logic

## Static 1 hazard

- $F = AB' + BC$
- Hazard happened when b changed
- Any "1" not in the same "Prime Implicant" (loop ) leaves open a potential static 1 hazard
- To fix: add a loop that spans the hazard
- $F = AB' + BC$ + AC



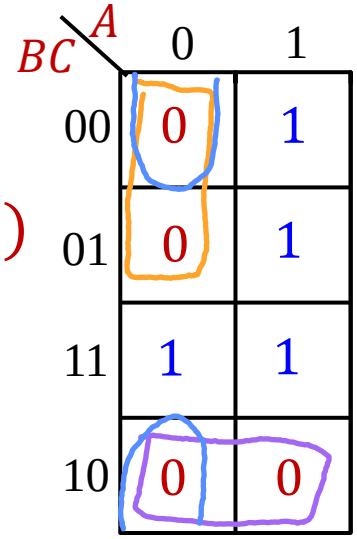




- Hazard removed by adding redundant logic
- Still SOP
- But no longer minimum SOP

# Hazards in Combinational Logic

## Static 0 hazard

- POS: circle the zeros
- $\bar{F} = A'B' + BC'$
- $F = \overline{A'B' + BC'} = (\overline{A'B'})(\overline{BC'}) = (A + B)(B' + C)$
- Potential static zero hazard for change of $B$
- Add loop for $A'C'$ to remove the static zero risk
- $F = (A + B)(B' + C)(A + C)$



# Hazards in Combinational Logic

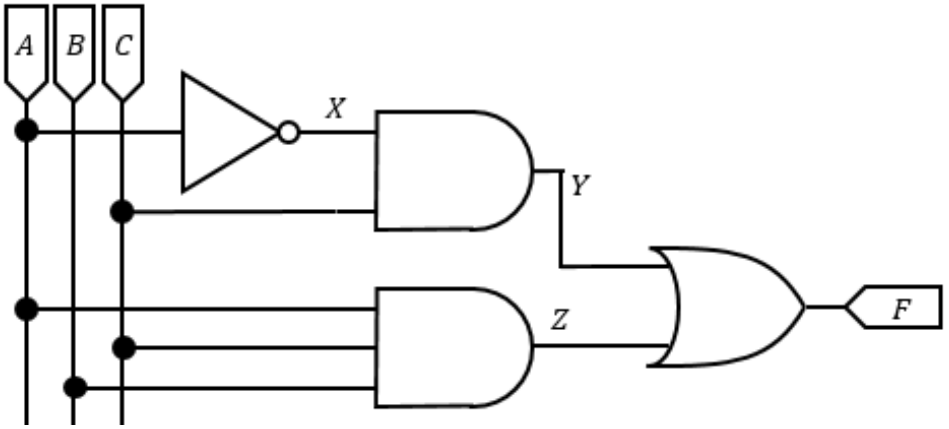
## Dynamic hazards

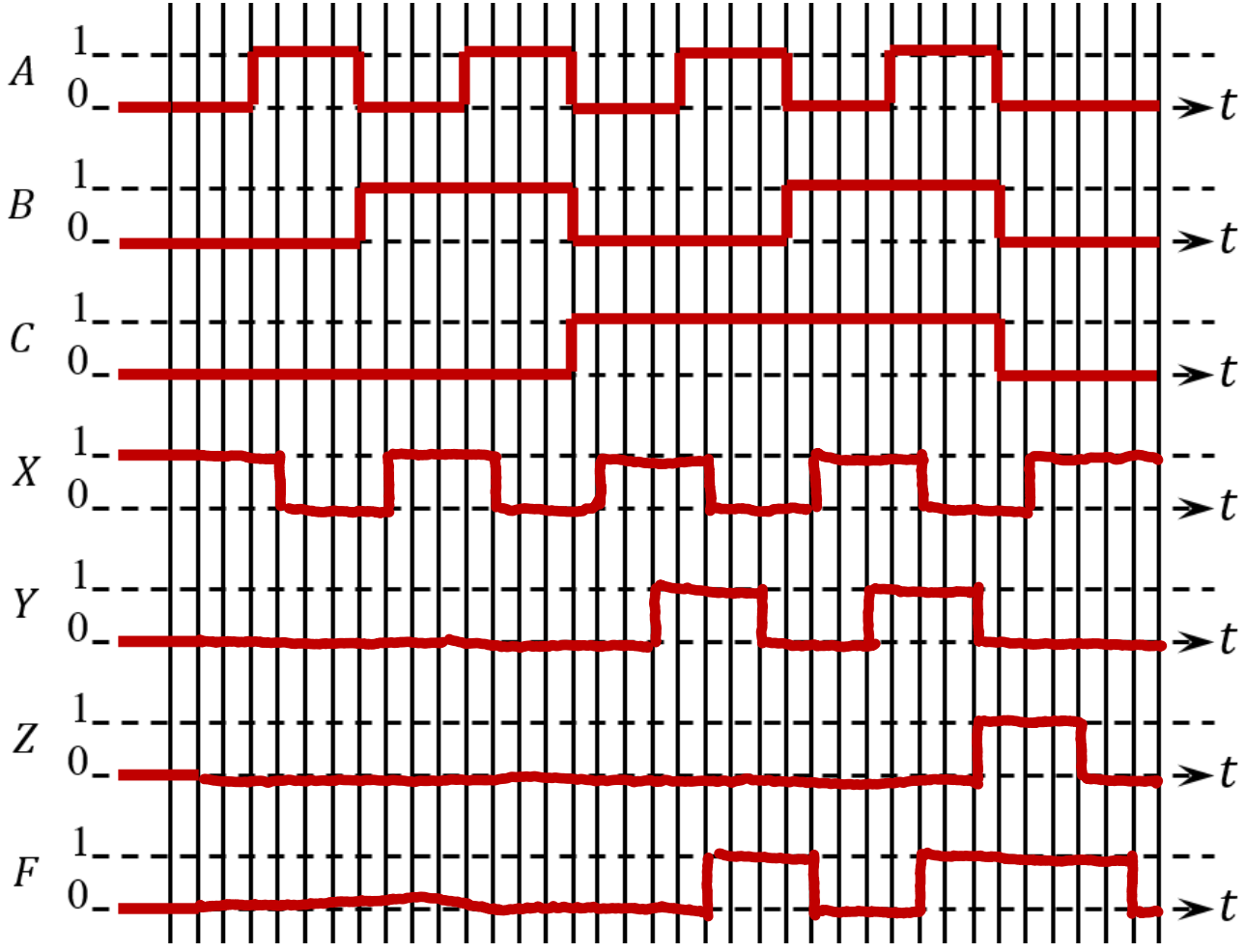
- Harder to identify
- Occur when multiple paths exist from input ⇨ output with different delays
- If ALL static hazards are resolved, then Dynamic Hazards no longer exist (but see the next slide)

# Hazards in Combinational Logic

- In our work on hazards we only considered 1 input changing at a time
- This is not always true
- Almost all circuits will have hazards if more than one input changes at a time
- They cannot all be eliminated by simply adding redundancy
- Glitches are of most importance in asynchronous sequential circuits
- The internal construction of latches and flip-flops we will learn later this semester are important examples of asynchronous sequential circuits
- But we will then see how to employ flip-flops in synchronous circuits
  - Included a clock signal to keep things synchronized
  - Take outputs to be valid only at specific times in clock cycle
  - Design circuit to ignore glitches from hazards at other times

# ECE2060 Timing Chart Example




Propagation delays for this example:

- Inverters: 5 ns
- 2-input gates: 10 ns
- 3-input gates: 15 ns

Vertical lines in timing diagram indicate 5 ns increments in time

# Chapter 9 Introduction

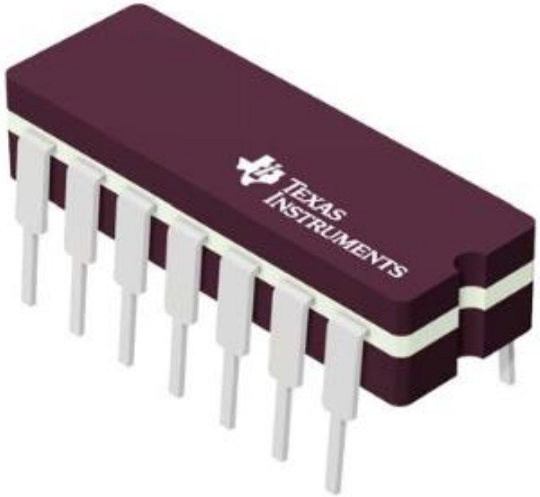
Switching to Chapter 9 now

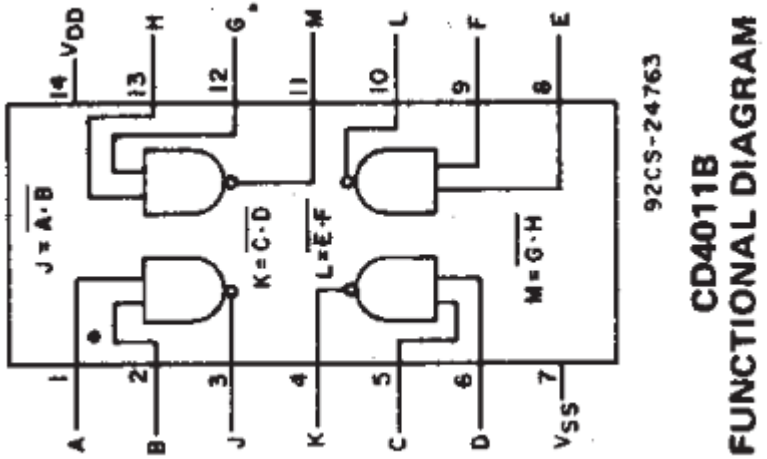
# Levels of Integration

- Chapter 9 begins to introduce more complex Integrated Circuits (ICs) used in logic design
- ICs may be classified as:
  - Small-scale integration (SSI),
  - Medium-scale integration (MSI),
  - Large-scale integration (LSI), or
  - Very-large-scale integration (VLSI)
- Classification depends on:
  - Number of Gates in each integrated circuit package
  - Type of function performed

# Levels of Integration

- SSI functions include:
  - NAND, NOR, AND, and OR gates (typ. 1-4 per package)
  - Inverters (typ. 6 per package)
  - Flip-flops (typ. 2 per package) (future lectures)
- The CD4011B – Quad 2 Input CMOS NAND that I used in an example in Lecture 7 is an SSI IC




- CD 4000-family includes various SSI and MSI ICs

# Levels of Integration

- MSI functions include: (examples - not an exhaustive list)
  - Adders
  - Multiplexers
  - Decoders
  - Registers (future lectures)
  - Counters (future lectures)
- MSI ICs typically contain the equivalent of a dozen to 100 gates per package
- LSI and VLSI:
  - More complex functions such as: (examples - not an exhaustive list)
    - Memory
    - Microprocessors
  - LSI: Hundreds to a few thousand gates
  - VLSI: Several thousand gates, or more

# Multiplexers

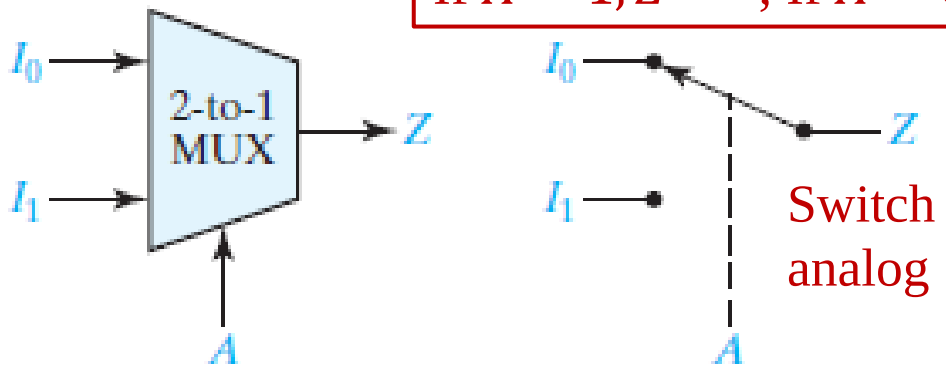
- A multiplexer (aka. data selector, abbreviated as MUX) has
  - A group of data inputs
  - A group of control inputs
- The control inputs are used to:
  - Select one of the data inputs
  - Connect it to the output terminal
  - A minterm example in Lectures 7 & 8 did something like that
- A 2-to-1 MUX and its logic equation are shown here

$$Z = A'I_0 + AI_1$$

If $A = 1, Z =$ ; If $A = 0, Z =$

FIGURE 9-1
2-to-1 Multiplexer and Switch Analog



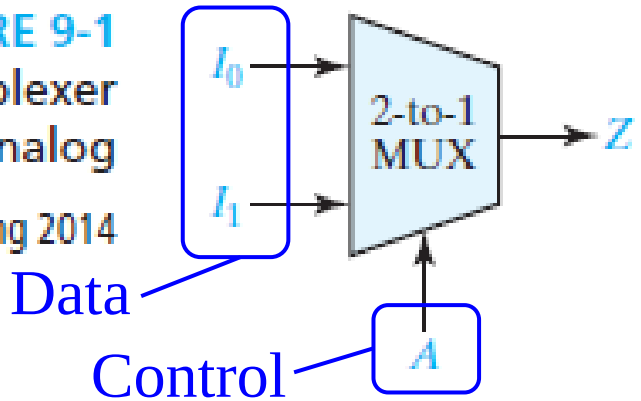


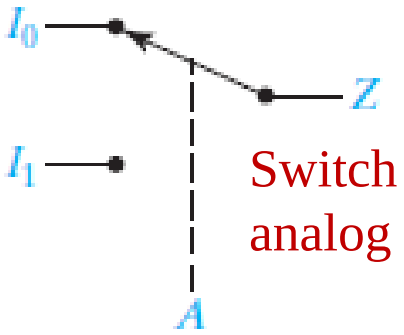
# Multiplexers

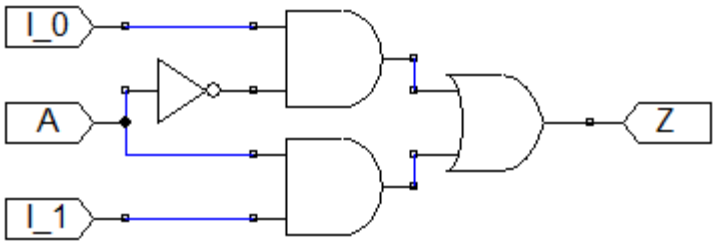
$$Z = A'I_0 + AI_1$$

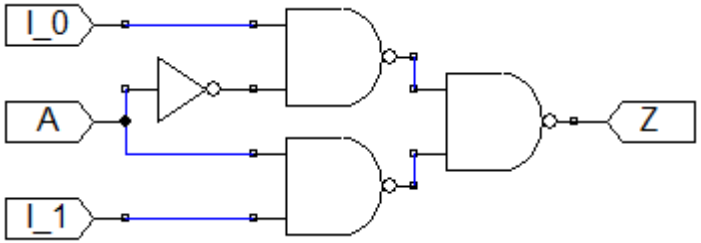
FIGURE 9-1
2-to-1 Multiplexer and Switch Analog



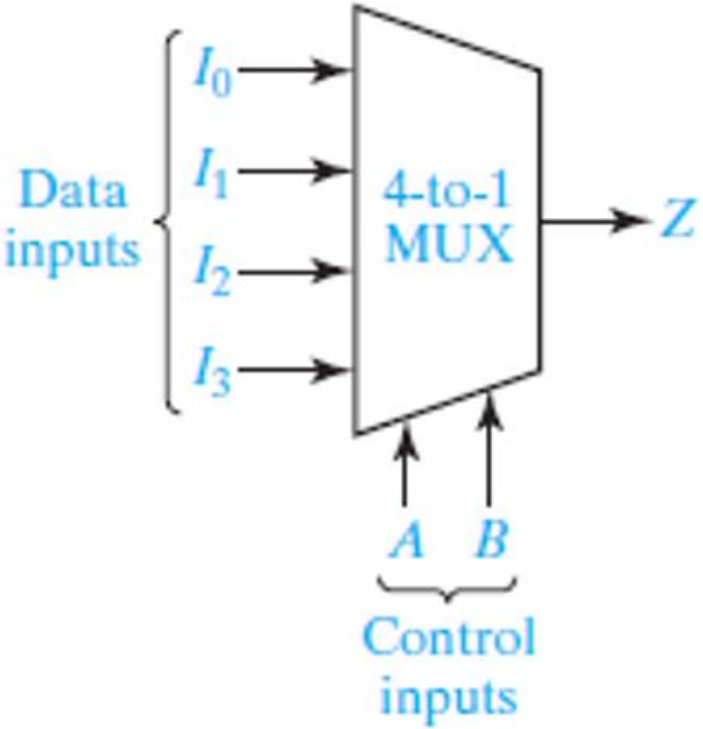


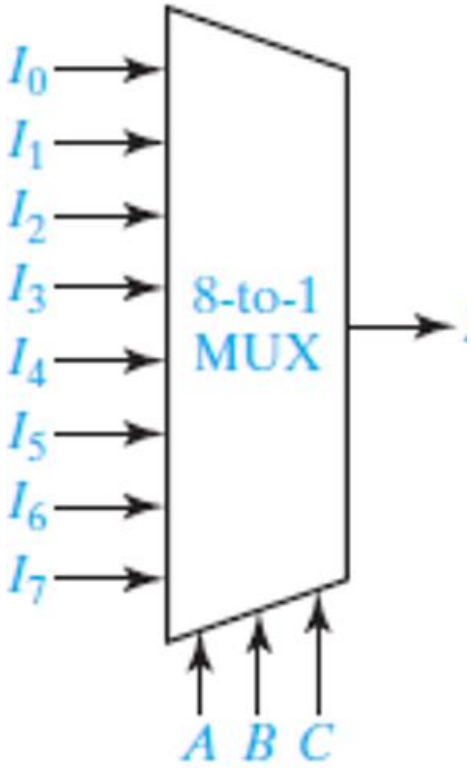



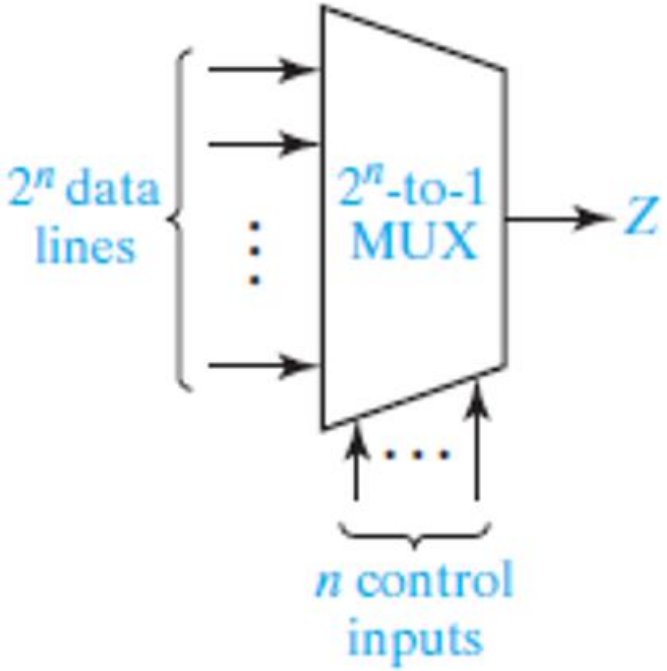



# Multiplexers

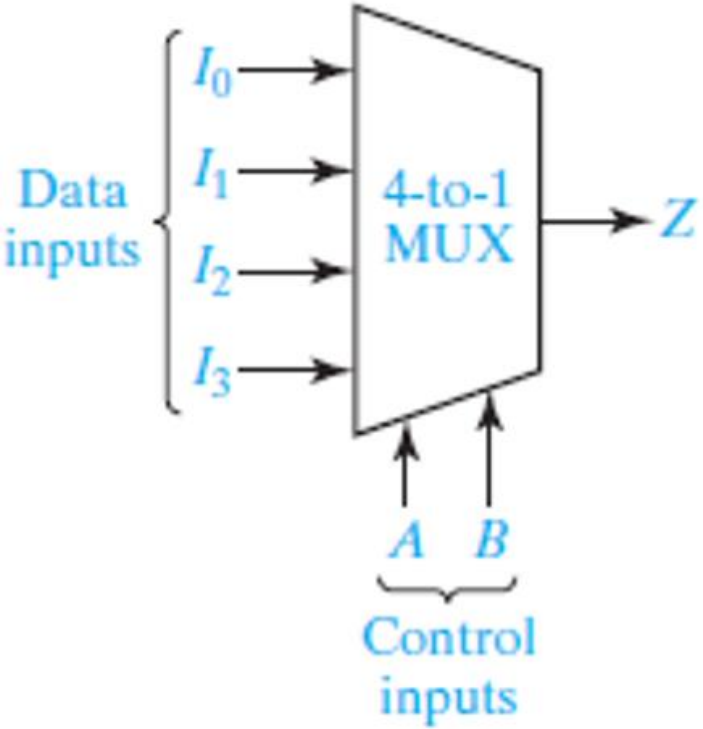



| A | B | Z |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |



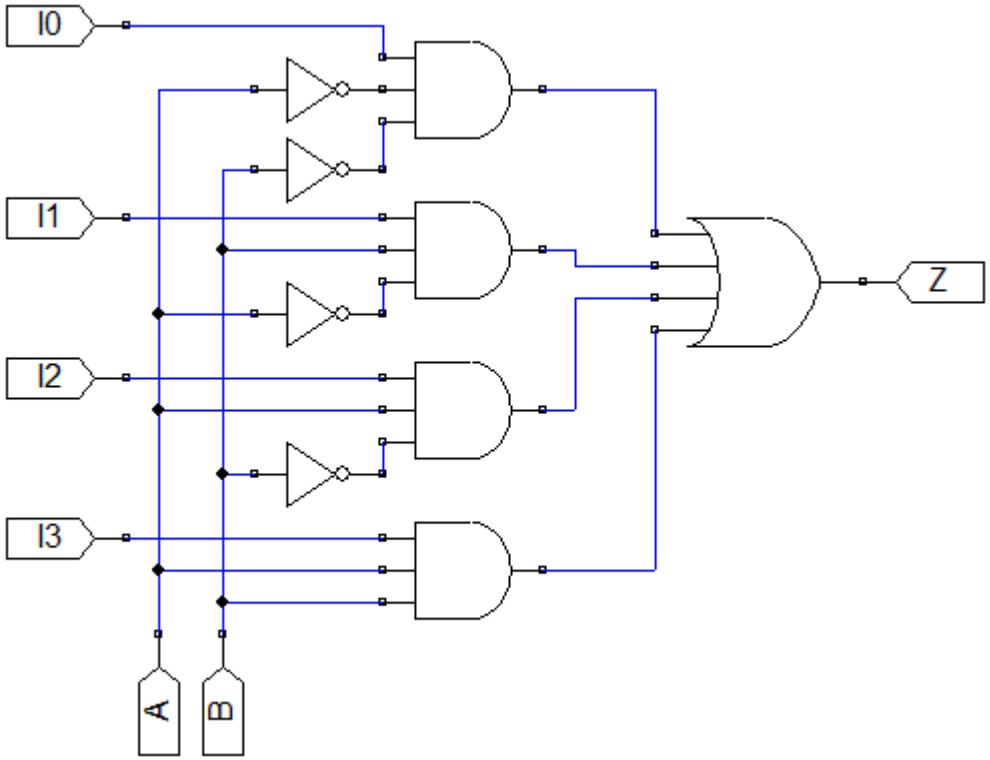



# Multiplexers

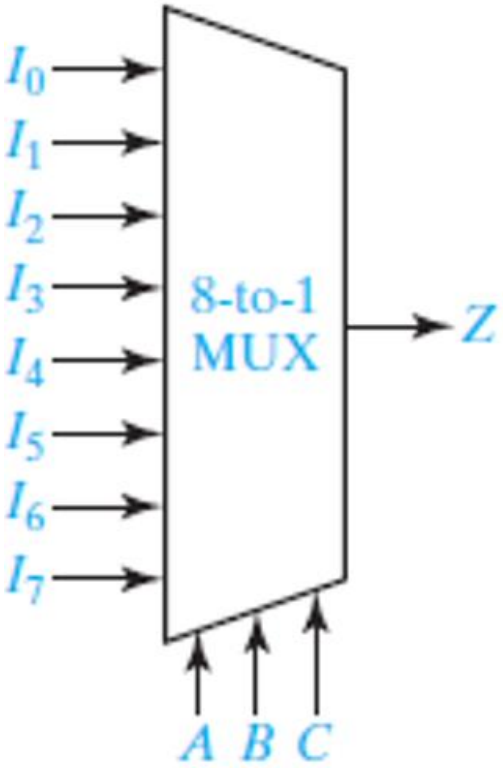


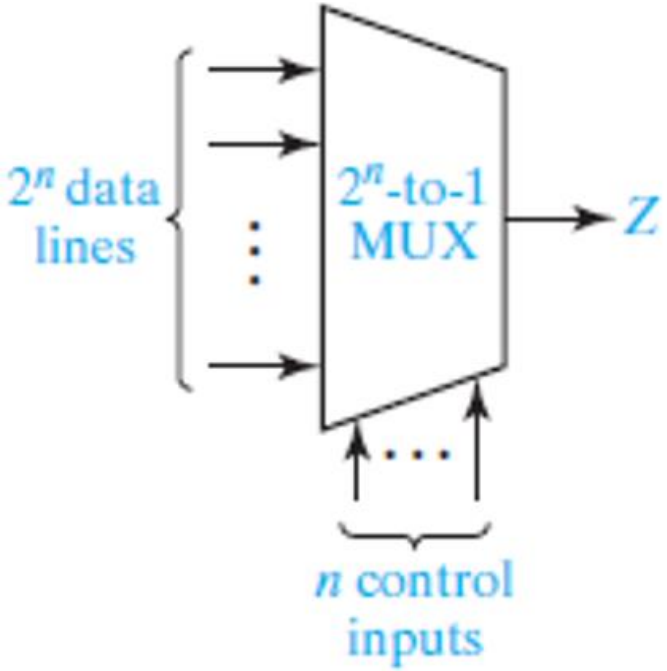

| A | B | Z |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

$$Z = \bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$$



# Multiplexers



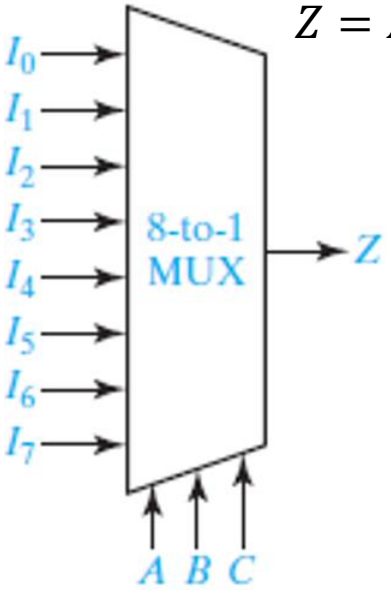




$$Z = \bar{A}\bar{B}\bar{C}I_0 + \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + A\bar{B}\bar{C}I_4 + A\bar{B}CI_5 + AB\bar{C}I_6 + ABCI_7$$

$$Z = \sum_{k=0}^{2^{n-1}} m_k I_k$$

# Multiplexers

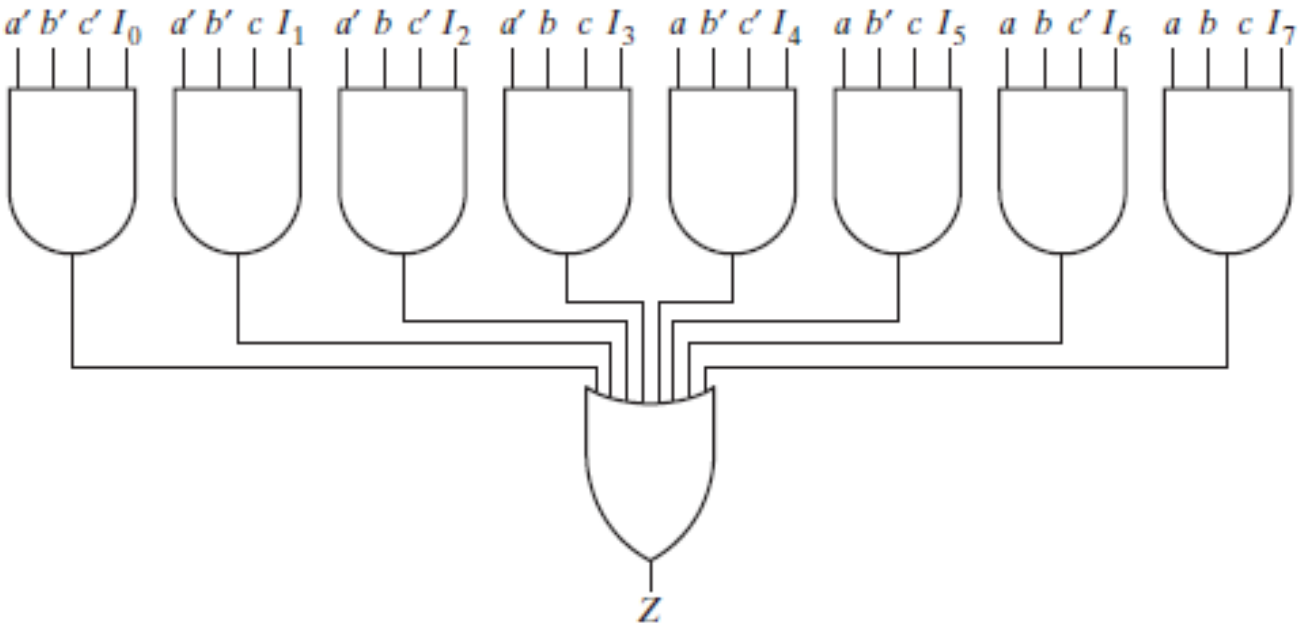
$$Z = \bar{A}\bar{B}\bar{C}I_0 + \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + A\bar{B}\bar{C}I_4 + A\bar{B}CI_5 + AB\bar{C}I_6 + ABCI_7$$




Logic Diagram for 8-to-1 MUX



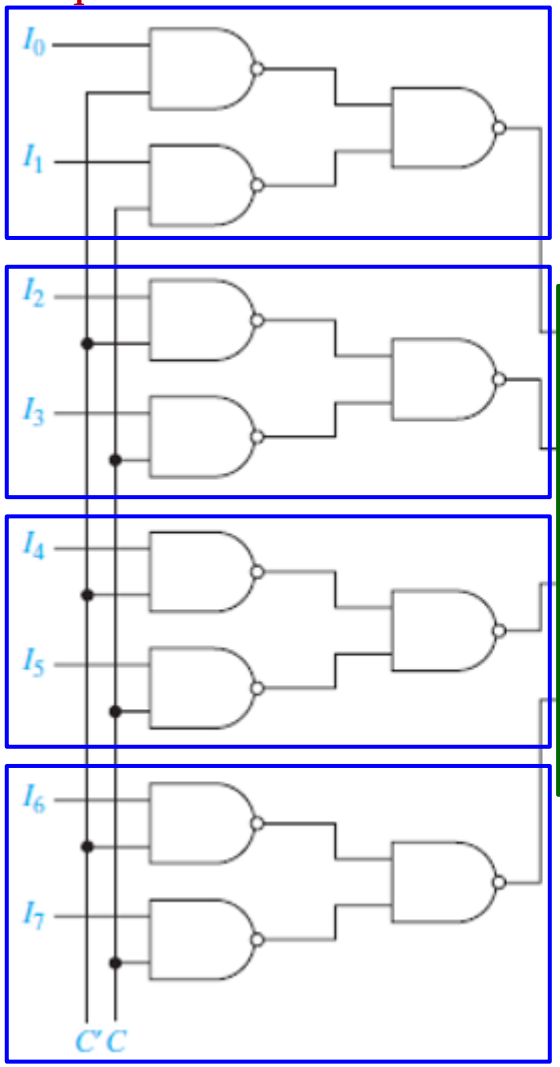
- Requires an OR with fan-in of 8
- Or an 8-NAND for NAND-NAND design
- Can design four level circuit
- But that is equivalent to
  - Four 4 to 1 MUX
  - Feeding into a 8 to 1 MUX



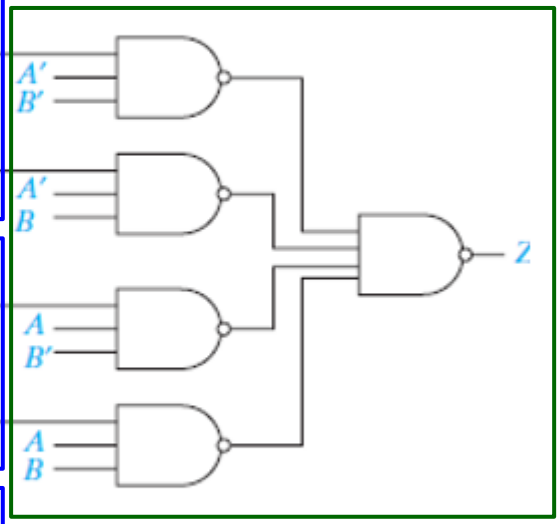
# Multiplexers

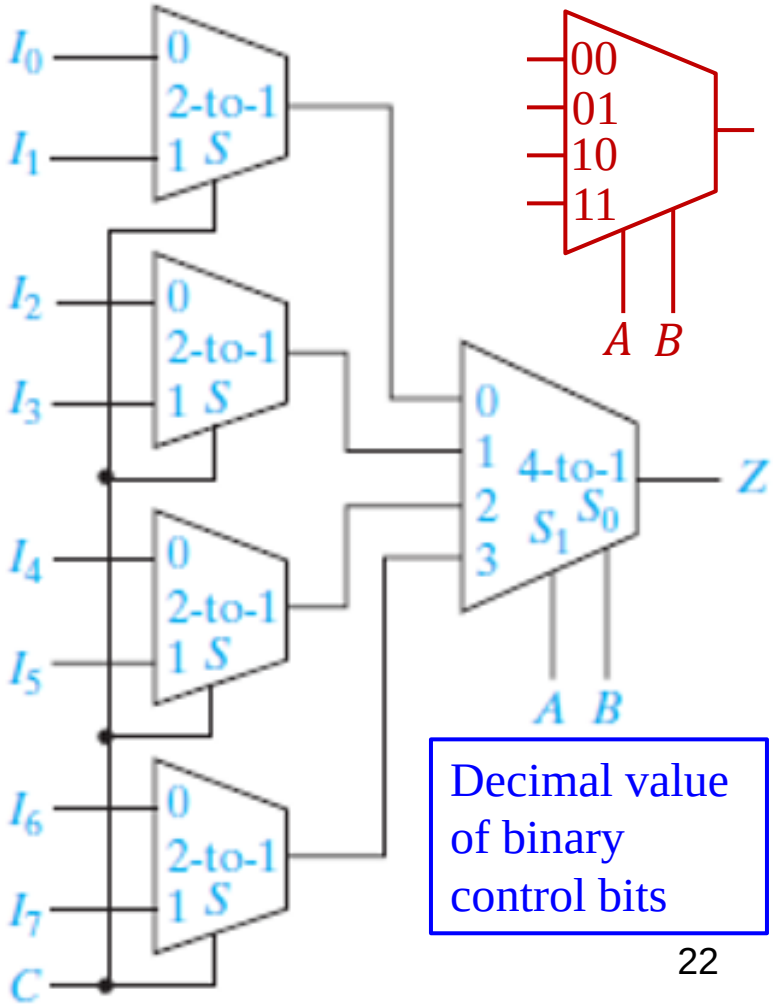
$$Z = \bar{A}\bar{B}\bar{C}I_0 + \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + A\bar{B}\bar{C}I_4 + A\bar{B}CI_5 + AB\bar{C}I_6 + ABCI_7$$

Equivalent to Four 2-to-1 MUX feeding into a 4-to-1 MUX




Note labelling with value of $S$, to indicate which input is selected



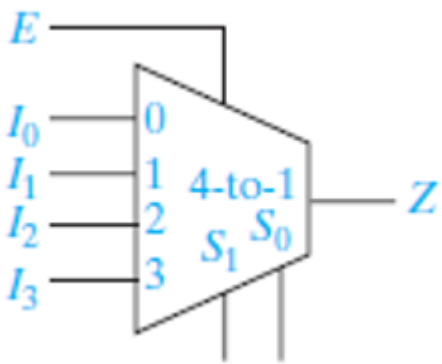

Decimal value of binary control bits

# Multiplexers

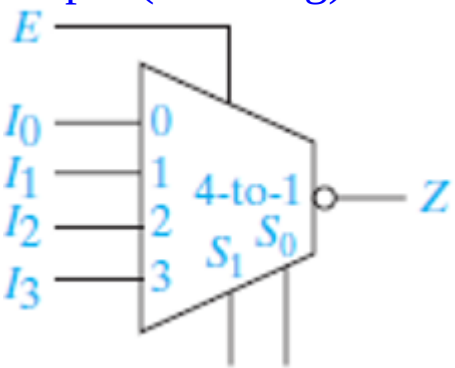
## Enable

- Another type of multiplexer has an additional input called an “ ”
- For example, 8-to-1 MUX can be modified to include an enable by changing the 4-input AND gates to five-input gates
- Enable signal $E$ connected to fifth input of each of the AND gates
  - If $E = 0$, $Z = 0$ independent of the data and control inputs
  - If $E = 1$ the MUX functions as an ordinary 8-to-1 multiplexer
- Variations:

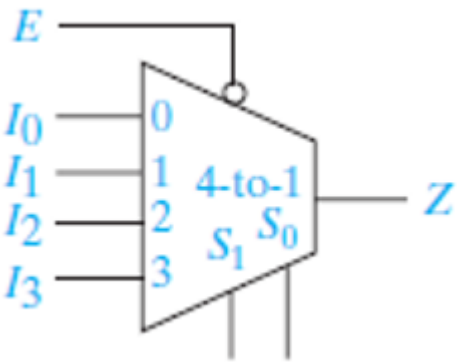
4-to-1 MUX with



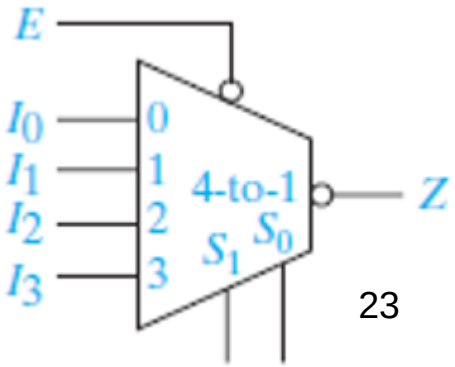

4-to-1 MUX with $E$ & output (inverting)




4-to-1 MUX with

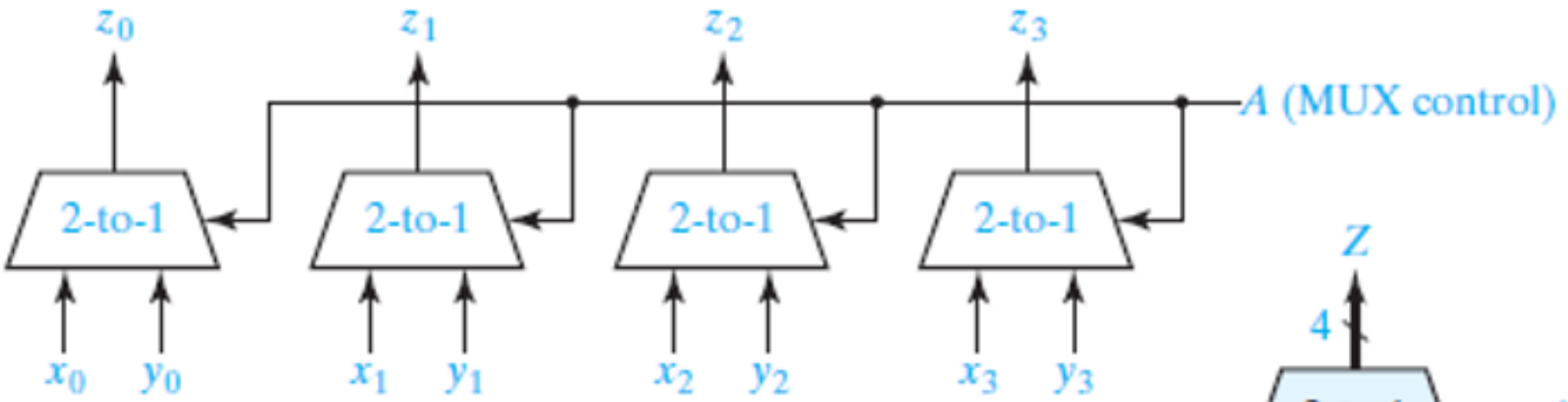



Active-low Enable & active low output



# Multiplexers

- Multiplexers often used to select data which is to be processed or stored in digital system design
- *N* identical MUX connected in parallel, with shared control, can be used to select between *N*-bit data words
- A 4-bit example:
  - $(x_3, x_2, x_1, x_0)$ and $(y_3, y_2, y_1, y_0)$ are the two 4-bit data words to select between
  - $(z_3, z_2, z_1, z_0)$ is the output data word




- Compact notation showing 4-bit buses at inputs and output ⇨
  - Thick line to represent bus
  - Diagonal mark with number designates number of bits on bus