

# Lecture 8: Medium Scale Integrated (MSI) Devices

Digital Electronics I

Digital Electronics I, Slide 8.1

## Points Addressed in this Lecture

- Adders
- 7 Segment Displays
- Comparators

Digital Electronics I, Slide 8.2

## Binary Addition

- ◆ Recall the binary addition process

|    |   |   |   |   |
|----|---|---|---|---|
| A  | 1 | 0 | 0 | 1 |
| +B | 0 | 0 | 1 | 1 |
| S  | 1 | 1 | 0 | 0 |

- LS Column has 2 inputs 2 outputs
  - Inputs:  $A_0, B_0$
  - Outputs:  $S_0, C_1$
- Other Columns have 3 inputs, 2 outputs
  - Inputs:  $A_n, B_n, C_n$
  - Outputs:  $S_n, C_{n+1}$
  - We use a "half adder" to implement the LS column
  - We use a "full adder" to implement the other columns
  - Each column feeds the next-most-significant column.

Digital Electronics I, Slide 8.3

## Half Adder

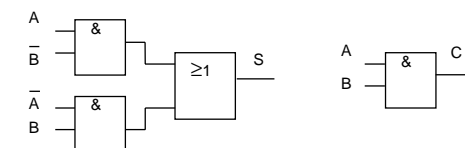
- ◆ Truth Table

| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

- Boolean Equations
 
$$S = \bar{A}B + A\bar{B}$$

$$C = AB$$

- ◆ Implementation



- Note also XOR implementation possible for S

Digital Electronics I, Slide 8.4

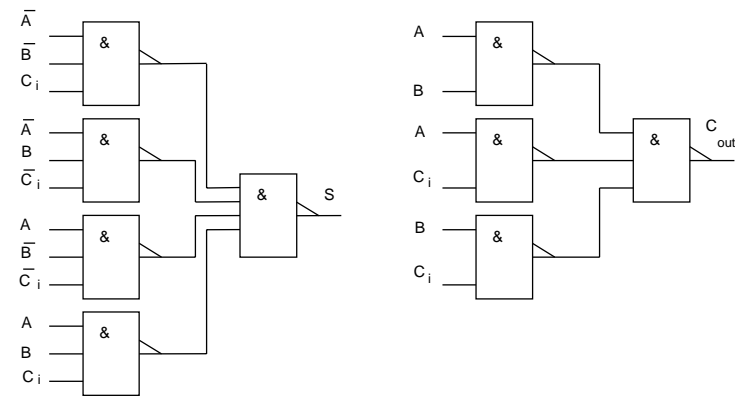
# Full Adder

◆ Truth Table

| A | B | C <sub>i</sub> | S | C <sub>o</sub> |
|---|---|----------------|---|----------------|
| 0 | 0 | 0              | 0 | 0              |
| 0 | 0 | 1              | 1 | 0              |
| 0 | 1 | 0              | 1 | 0              |
| 0 | 1 | 1              | 0 | 1              |
| 1 | 0 | 0              | 1 | 0              |
| 1 | 0 | 1              | 0 | 1              |
| 1 | 1 | 0              | 0 | 1              |
| 1 | 1 | 1              | 1 | 1              |

- ◆ Boolean Equations
- $$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$$
- $$= A \oplus B \oplus C_i$$
- $$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$
- $$= AB + AC_i + BC_i$$
- $$= AB + C_i(A + B)$$

◆ Implementation (using NAND gates only)



- Note: Full adder can also be obtained from 2 half adders.
- (See Question Sheet)

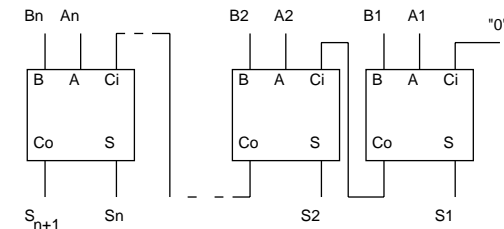
# Full Adder from Half Adders

◆ Truth Table

| A | B | HA <sub>s</sub> | HA <sub>c</sub> | C <sub>i</sub> | S | C <sub>o</sub> |
|---|---|-----------------|-----------------|----------------|---|----------------|
| 0 | 0 | 0               | 0               | 0              | 0 | 0              |
| 0 | 0 | 0               | 0               | 1              | 1 | 0              |
| 0 | 1 | 1               | 0               | 0              | 1 | 0              |
| 0 | 1 | 1               | 0               | 1              | 0 | 1              |
| 1 | 0 | 1               | 0               | 0              | 1 | 0              |
| 1 | 0 | 1               | 0               | 1              | 0 | 1              |
| 1 | 1 | 0               | 1               | 0              | 0 | 1              |
| 1 | 1 | 0               | 1               | 1              | 1 | 1              |

# Parallel Adder

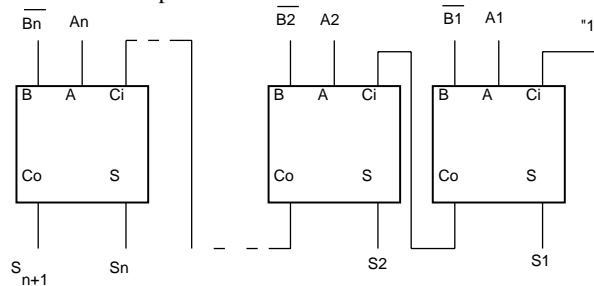
◆ Uses 1 full adder per bit of the numbers



- ◆ The carry is propagated from one stage to the next most significant stage
  - takes some time to work because of the carry propagation delay which is n times the propagation delay of one stage.
- ◆ Available as a single part (74LS83 - 4-bit full adder)

## Parallel Subtraction using Parallel Adder

- ◆ Subtraction can be achieved by adding the complement
  - E.g.:  $6 - 3 = 6 + (-3) = 3$
- ◆ 2's complement :- invert all bits and then add 1
  - Use Carry-in of first stage for the "add 1"
  - Invert all the inputs bits of B



Digital Electronics I, Slide 8.9

## Delays in Adder Circuits

- ◆ In the parallel adder, it takes time for the carry outputs to propagate through all the bits
  - 2 gate delays per bit of adder
- ◆ Worst case (4-bit example)
  - $(1111)_2 + (0001)_2$
  - carry out generated at every stage
- ◆ Sum output of  $i^{\text{th}}$  stage is valid after  $(2*i + 2)$  gate delays
- ◆ Carry output of  $i^{\text{th}}$  stage is valid after  $(2*i + 2)$  gate delays

Digital Electronics I, Slide 8.10

## Carry Look-Ahead Adder

- ◆ Carry look-ahead adder aims to overcome this limitation
  - The carries are generated in advance for all the stages of the adder
  - Extra logic required but it operates faster
- ◆ For the  $i^{\text{th}}$  stage
- ◆ if  $A = B = 1$  then  $C_{out}$  always equals 1
  - 2 gate delays
- ◆ if  $A = B = 0$  then  $C_{out}$  always equals 0
  - 2 gate delays
- ◆ if  $A \neq B$  then  $C_{out}$  depends on  $C_{in}$ 
  - 2 or 3 gate delays depending on implementation (see slide 8.5)

Digital Electronics I, Slide 8.11

- ◆ Idea of carry lookahead is to express the carry outputs in terms of the inputs
  - the carry out of stage  $i$  should be 1 if
    - (a carry is generated in stage  $i$  because both A and B are 1) or
    - (the carry in to stage  $i$  is 1 and one of A or B is 1)
- ◆ Let the carry generation function at stage  $i$  be
 
$$G = A \cdot B$$
- ◆ Let the carry propagation function at stage  $i$  be
 
$$P = A \oplus B$$
- ◆ Then
 
$$S = P \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

$$= AB + C_{in}(A \oplus B)$$

$$= G + C_{in}P$$

Digital Electronics I, Slide 8.12

- ◆ Each carry-out can be expressed in terms of P's and G's from previous stages and the carry-in of the zero'th stage

$$C_1 = G_0 + P_0C_0$$

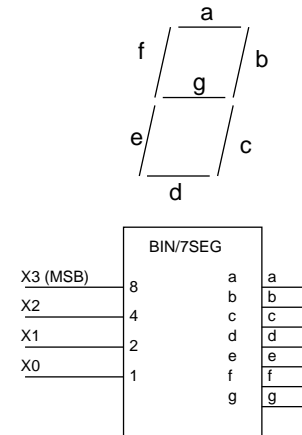
$$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

etc

## 7 Segment Displays

- ◆ LCD or LED displays can display digits made of up to 7 segments or lines
- ➔ ◆ Hex character set needs 4 bits
  - Includes decimal character set
- ◆ Decode 4 bits into 7 control signals using a BIN/7SEG decoder

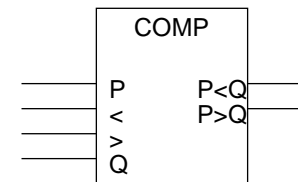


- ◆ The 7 segment decoder is the largest circuit that would be economic to implement using gates
  - We will see other implementation methods later (ROM, PLD)

| X3   | X2 | X1 | X0 | a | b | c | d | etc. |
|------|----|----|----|---|---|---|---|------|
| 0    | 0  | 0  | 0  |   |   |   |   |      |
| 0    | 0  | 0  | 1  |   |   |   |   |      |
| 0    | 0  | 1  | 0  |   |   |   |   |      |
| 0    | 0  | 1  | 1  |   |   |   |   |      |
| 0    | 1  | 0  | 0  |   |   |   |   |      |
| 0    | 1  | 0  | 1  |   |   |   |   |      |
| 0    | 1  | 1  | 0  |   |   |   |   |      |
| 0    | 1  | 1  | 1  |   |   |   |   |      |
| 1    | 0  | 0  | 0  |   |   |   |   |      |
| 1    | 0  | 0  | 1  |   |   |   |   |      |
| etc. |    |    |    |   |   |   |   |      |

## Comparators

- ➔ ◆ A circuit which compares 2 unsigned numbers
- ◆ Each bit is compared, MSB first
  - The first bit which differs determines which number is bigger
- ◆ 1-bit comparator:



- ➔ ◆ The 1-bit inputs are presented at P and Q
- ◆ Outputs P>Q and P<Q set according to P and Q
- ◆ Over-ride inputs, > and <, are used to force one of the outputs to be true.

# Multi-bit Comparators

- ◆ n 1-bit comparators can be combined
  - E.g. 3-bit comparator

