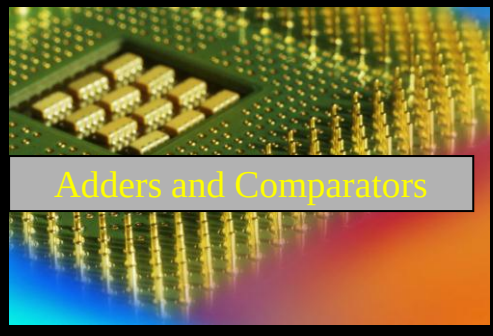
EEE251

Logic Design and Circuits

Assist.Prof. Yavuz BAYAM



# Adders and Comparators

December 9, 2013

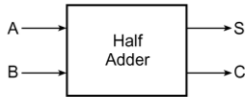
## Half-Adder

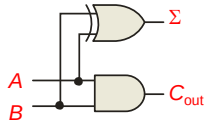
Basic rules of binary addition are performed by a **half adder**, which has two binary inputs (*A* and *B*) and two binary outputs (Carry out and Sum).

The inputs and outputs can be summarized on a truth table.

| Inputs | | Outputs | |
|---|---|---|---|
| *A* | *B* | $C_{out}$ | $\Sigma$ |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

The logic symbol and equivalent circuit are:



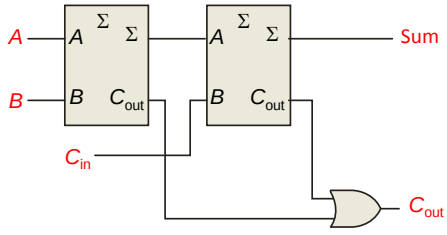



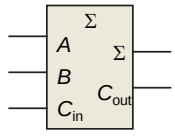
## Full-Adder

By contrast, a **full adder** has three binary inputs (*A*, *B,* and Carry in) and two binary outputs (Carry out and Sum). The truth table summarizes the operation.

A full-adder can be constructed from two half adders as shown:

| Inputs | | | Outputs | |
|---|---|---|---|---|
| $A$ | $B$ | $C_{in}$ | $C_{out}$ | $\Sigma$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



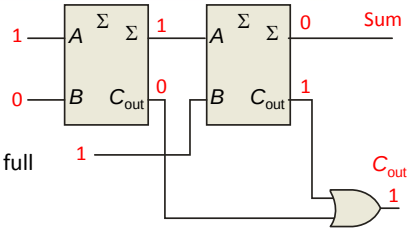




Symbol

## Full-Adder

**Example**

For the given inputs, determine the intermediate and final outputs of the full adder.




**Solution**

The first half-adder has inputs of 1 and 0; therefore the Sum =1 and the Carry out = 0.

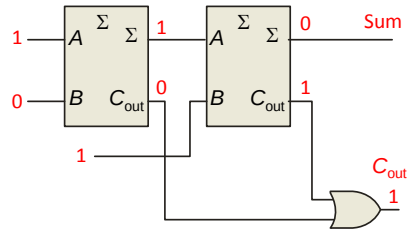
The second half-adder has inputs of 1 and 1; therefore the Sum = 0 and the Carry out = 1.

The OR gate has inputs of 1 and 0, therefore the final carry out = 1.

## Full-Adder

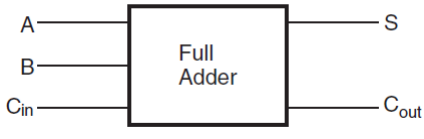
Notice that the result from the previous example can be read directly on the truth table for a full adder.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| $A$ | $B$ | $C_{in}$ | $C_{out}$ | $\Sigma$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

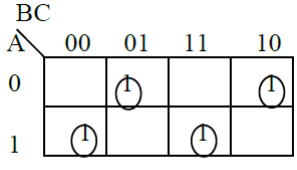



## Full-Adder

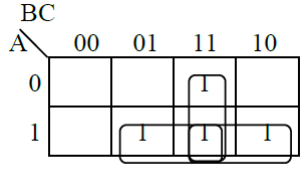
- A full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a SUM and a CARRY output




| A | B | C | S | Co |
|---|---|---|---|---|
| 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 |




$S=\bar{A}\bar{B}C+\bar{A}\bar{B}C+ABC+A\bar{B}\bar{C}$

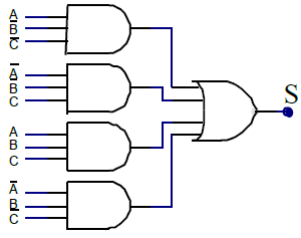


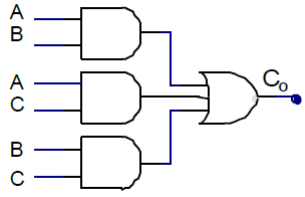

$Co=AC+BC+AB$

## Full-Adder

$S=\bar{A}\bar{B}C+\bar{A}\bar{B}C+ABC+\bar{A}B\bar{C}$

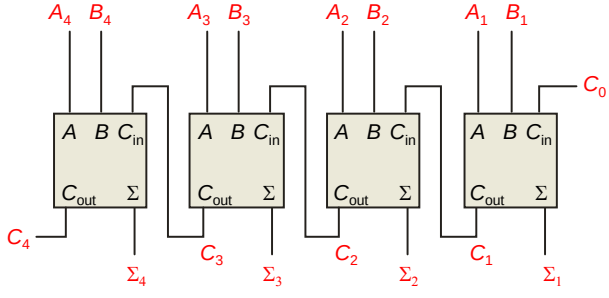
$Co=AC+BC+AB$




## Parallel Adders

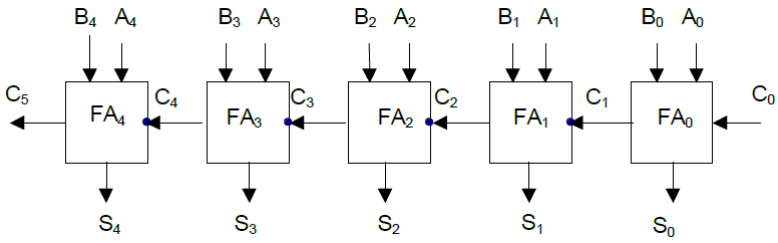
Full adders are combined into parallel adders that can add binary numbers with multiple bits. A 4-bit adder is shown.




The output carry ($C_4$) is not ready until it propagates through all of the full adders. This is called *ripple carry*, delaying the addition process.

## Parallel Adders

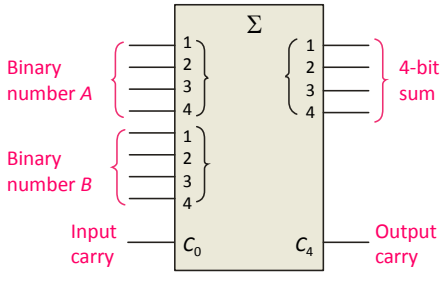
- For the addition of two *n* bits of data, *n* numbers of full adders can be cascaded as demonstrated in the figure




- The addition technique adopted here is a parallel type as all the bit addition operations are performed in parallel
- Therefore, this type of adder is called a parallel adder

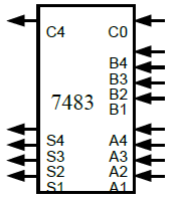
## Parallel Adders

The logic symbol for a 4-bit parallel adder is shown. This 4-bit adder includes a carry in (labeled ($C_0$) and a Carry out (labeled $C_4$).

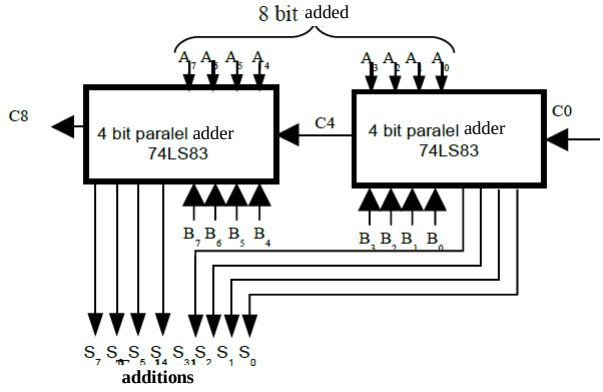



The 74LS283 is an example. It features *look-ahead carry*, which adds logic to minimize the output carry delay. For the 74LS283, the maximum delay to the output carry is 17 ns.

## Parallel Adders

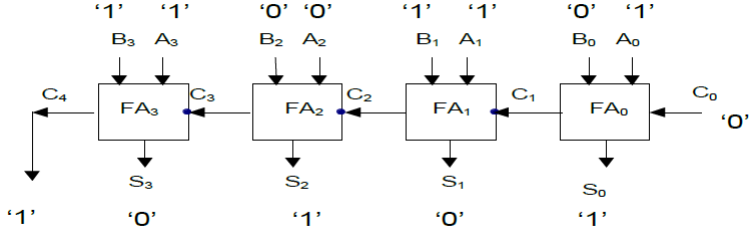



74LS83 **Symbol**



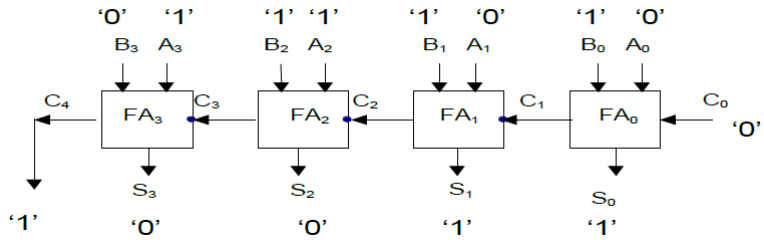

## Parallel Adders

Example: Add the numbers 1011 and 1010 by using 4 bit parallel adders



## Parallel Adders

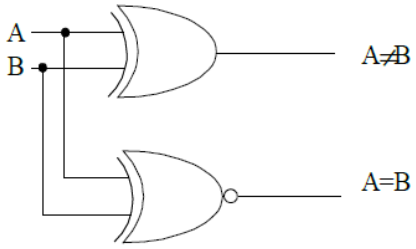
**Example:** Add the numbers 0111 and 1100 by using 4 bit parallel adders




## Comparators

- A magnitude comparator is a combinational circuit that compares two given numbers and determines whether one is equal to, less than or greater than the other
- Commonly used in arithmetic logic units
- If two binary numbers are considered as A and B, the magnitude comparator gives three outputs for A > B, A < B, and A = B.
- A magnitude comparator is one of the useful combinational logic networks and has wide applications

## Comparators




| Inputs | | Outputs | |
|---|---|---|---|
| A | B | A≠B | A=B |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

## Comparators

Design a comparator to compare 2 bit data and produce outputs for A=B, A>B and A<B

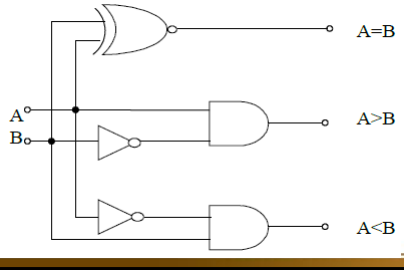
| A | B | A>B | A=B | A<B |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |

$A>B=AB'$

$A=B=A'B'+AB$

$=A\odot B$

$A<B=A'B$



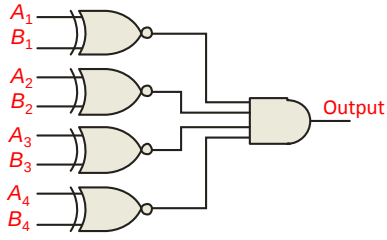
## Comparators

The function of a comparator is to compare the magnitudes of two binary numbers to determine the relationship between them. In the simplest form, a comparator can test for equality using XNOR gates.

**Example Solution**

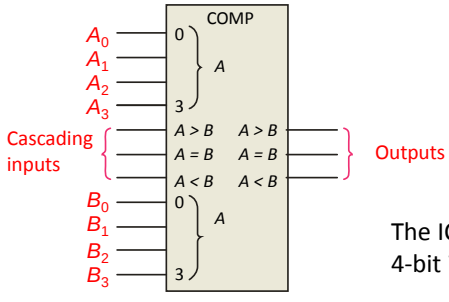
How could you test two 4-bit numbers for equality?

AND the outputs of four XNOR gates.




## Comparators

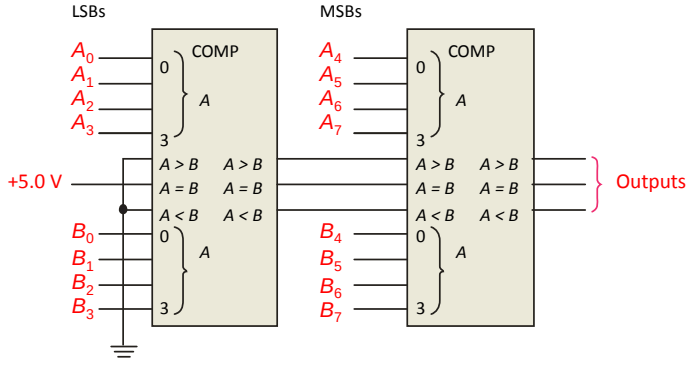
IC comparators provide outputs to indicate which of the numbers is larger or if they are equal. The bits are numbered starting at 0, rather than 1 as in the case of adders. Cascading inputs are provided to expand the comparator to larger numbers.




The IC shown is the 4-bit 74LS85.

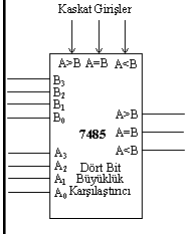
## Comparators

IC comparators can be expanded using the cascading inputs as shown. The lowest order comparator has a HIGH on the *A* = *B* input.




## Comparators

- For example, 7485 is a four-bit magnitude comparator of the TTL logic family
- These devices can be cascaded together to perform operations on larger bit numbers without the help of any external gates

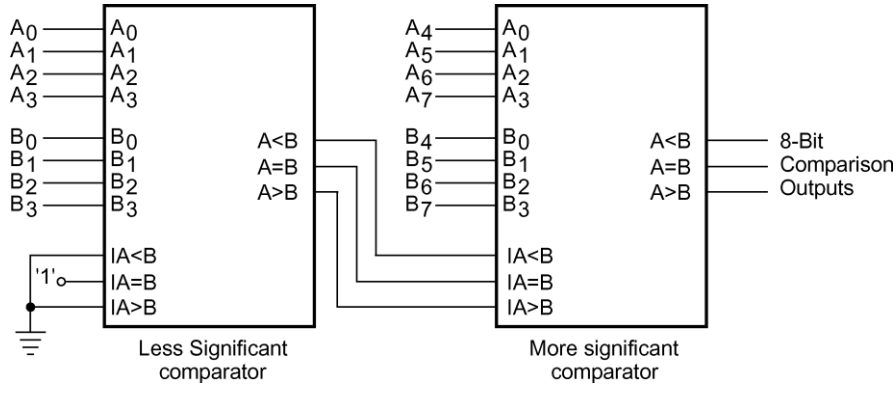



| Comparison inputs | | | | Cascading inputs | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|
| $A_3,B_3$ | $A_2,B_2$ | $A_1,B_1$ | $A_0,B_0$ | A > B | A < B | A = B | A > B | A < B | A = B |
| $A_3 > B_3$ | $X$ | $X$ | $X$ | $X$ | $X$ | $X$ | HIGH | LOW | LOW |
| $A_3 < B_3$ | $X$ | $X$ | $X$ | $X$ | $X$ | $X$ | LOW | HIGH | LOW |
| $A_3 = B_3$ | $A_2 > B_2$ | $X$ | $X$ | $X$ | $X$ | $X$ | HIGH | LOW | LOW |
| $A_3 = B_3$ | $A_2 < B_2$ | $X$ | $X$ | $X$ | $X$ | $X$ | LOW | HIGH | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 > B_1$ | $X$ | $X$ | $X$ | $X$ | HIGH | LOW | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 < B_1$ | $X$ | $X$ | $X$ | $X$ | LOW | HIGH | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 > B_0$ | $X$ | $X$ | $X$ | HIGH | LOW | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 < B_0$ | $X$ | $X$ | $X$ | LOW | HIGH | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | HIGH | LOW | LOW | HIGH | LOW | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | LOW | HIGH | LOW | LOW | HIGH | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | LOW | LOW | HIGH | LOW | LOW | HIGH |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | $X$ | $X$ | HIGH | LOW | LOW | HIGH |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | HIGH | HIGH | LOW | LOW | LOW | LOW |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | LOW | LOW | LOW | HIGH | HIGH | LOW |

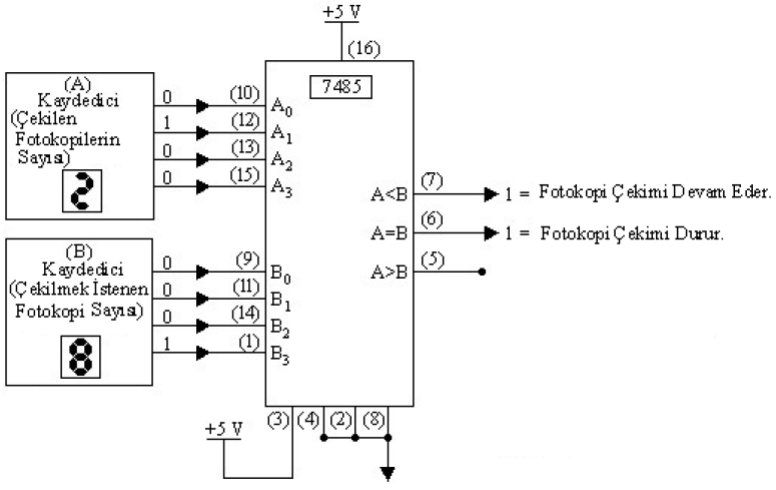
## Comparators

- The two numbers being compared here are ($A_7$ ….$A_0$) and (B7 …..B0).
- The less significant comparator handles (A3, A2, A1, A0) and (B3, B2, B1, B0), and the more significant comparator handles (A7, A6, A5, A4) and (B7, B6, B5, B4).
- Let us take the example of the two numbers being such that A7 > B7.
- From the first-row entry of the function table it is clear that, irrespective of the status of other bits of the more significant comparator, and also regardless of the status of its cascading inputs, the final output produces a HIGH at the A>B output and a LOW at the A<B and A = B outputs.
- Since the status of cascading inputs of the more significant comparator depends upon the status of comparison bits of the less significant comparator, the cascade arrangement produces the correct output for A7 > B7 regardless of the status of all other comparison bits.
- On similar lines, the circuit produces a valid output for any given status of comparison bits.

## Comparators



## Comparators




). Fotokopi makinası kontrol devresi.