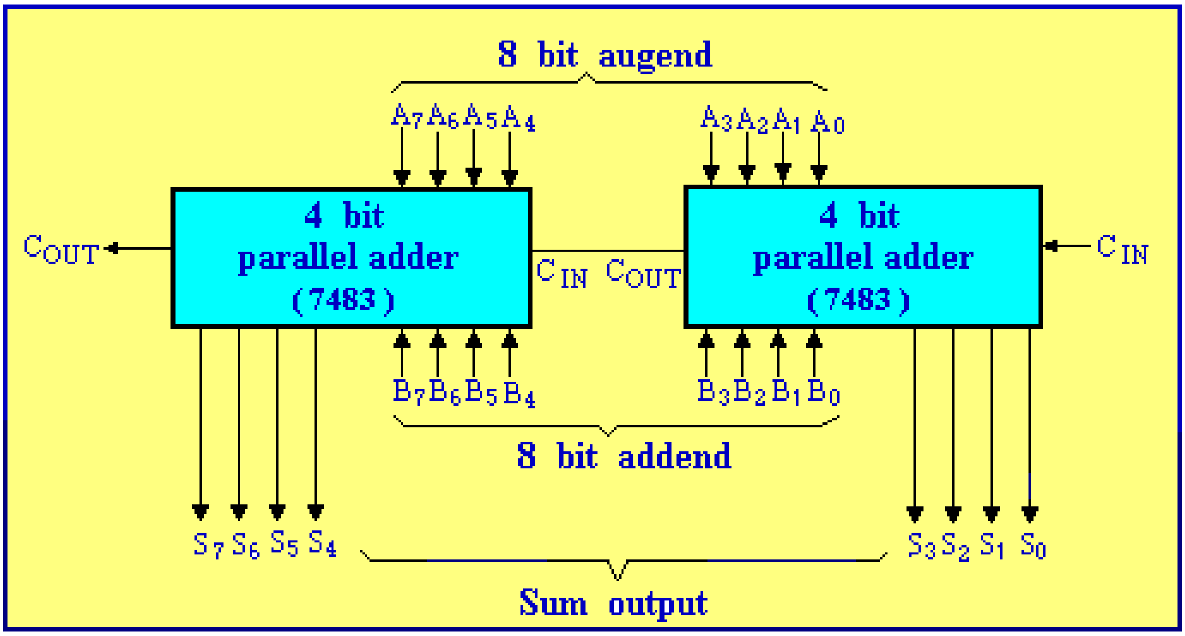
6. Digital Arithmetic and Circuits



Objectives

1. Perform binary addition on two binary numbers.
2. Know difference between binary addition and OR addition.
3. Manipulate signed binary numbers using the 2's complement numbering system.
4. Understand the BCD addition process.
5. Design the Full Adder unit using combinational design methodology.
6. Employ full adders in the design of parallel binary adders.

Binary Addition

Addition of Binary numbers is performed the same way as with decimal numbers, i.e. is done starting with the LSD, on a digit-by-digit basis with the carry-out added to the next digit. E.g,

$$\begin{array}{ccccc} & 7 & 4 & {}^{1}2 & 8 \\ + & 2 & 1 & 5 & 3 \\ \hline & 9 & 5 & 8 & 1 \end{array}$$

The same general steps are used in binary addition.

0 + 0 = 0	
1 + 0 = 1	
1 + 1 = 10	0 + carry of 1 into next position
1 + 1 + 1 = 11	1 + carry of 1 into next position

For Example,

1)

$$\begin{array}{ccccc} & 0 & 1 & 1 & (3) \\ + & 1 & 1 & 0 & (6) \\ \hline 1 & 0 & 0 & 1 & (9) \end{array}$$

2)

$$\begin{array}{ccccccc} & 1 & 1 & . & 0 & 1 & 1 & (3.375) \\ + & 1 & 0 & . & 1 & 1 & 0 & (2.750) \\ \hline 1 & 1 & 0 & . & 0 & 0 & 0 & (6.125) \end{array}$$

Representing Signed Numbers

Signed Binary numbers can be represented in two ways:

(a) Sign-magnitude system

(b) 2's complement system

(a) Sign-magnitude.

In this system, the MSB is designated the sign bit and the magnitude is expressed in the true binary form for both positive and negative numbers:

- a positive (+ve) number is indicated by a sign bit of 0
- a negative (-ve) number is indicated by a sign bit of 1

Eg. Representation of decimal +52 and -52 in sign-magnitude form.

A7	A6	A5	A4	A3	A2	A1	A0	
0	0	1	1	0	1	0	0	= + 52
sign bit								+ve number

A7	A6	A5	A4	A3	A2	A1	A0	
1	0	1	1	0	1	0	0	= - 52
sign bit								-ve number

In order to understand the 2nd system,

1's Complement form

The One’s complement of a binary number is obtained by complementing or inverting each bit in the binary number.

1	0	1	1	0	1	Original binary number
⇓	⇓	⇓	⇓	⇓	⇓	
0	1	0	0	1	0	Invert each bit to form 1's complement

2's Complement form

The Two's complement of a binary number is formed by taking the 1's complement of the binary number and adding 1 to the LSB.

Eg:

1	0	1	1	0	1	True binary equivalent of 45_{10}
⇓	⇓	⇓	⇓	⇓	⇓	Invert each bit to form 1's complement
0	1	0	0	1	0	
				+	1	Add 1 to form 2's complement
0	1	0	0	1	1	**2's complement** of original binary number

(b) Signed Numbers Using 2's complement.

If the number is positive - magnitude is represented in true binary form and sign bit of 0 is placed in front of MSB.

0	1	0	1	1	0	1	= + 45_{10}
Sign bit	True binary form						

If the number is negative - magnitude is represented in 2's complement form and sign bit of 1 is placed in front of MSB.

0	1	0	1	1	0	1	= + 45_{10}
1	0	1	0	0	1	0	1's complement
					+	1	Add 1
1	0	1	0	0	1	1	= - 45_{10}
Sign bit							2's complement

Numbers in 2's complement form can be added and subtracted using same circuit, which is basically an adder.

Examples on 2's Complement system

1. Express -125_{10} in 8-bits 2's Complement numbering system.

Start with +125	→	0	1	1	1	1	1	0	1
1's complement	→	1	0	0	0	0	0	1	0
+ 1 to 1's complement								+	1
to obtain 2's com		1	0	0	0	0	0	1	1

This value, i.e. the 2's complement represents the negative value of +125

2. Other Examples:

$+95_{10}$	=	0	1	0	1	1	1	1	1

$+69_{10}$	=	0	0	0	0	0	1	0	1
1's com		1	0	1	1	1	0	1	0
2’s								+	1
-69_{10}		1	0	1	1	1	0	1	1

Special Cases in 2's Complement

Whenever a signed number has a 1 in the sign bit and all 0's for the magnitude bits, its decimal equivalent is -2^N ,
where **N** is the number of bits in the magnitude.

E.g. 1000 = -2^3 = -8
1000 = -2^4 = -16, etc.

Complete range of values in the 2's complement system is:

$$-2^N \quad \text{to} \quad +(2^N - 1)$$

There will be a total of 2^{N+1} different values including zero.

E.g. For a 4 bits using the 2's complement system,

Decimal Value	Signed binary 2's complement		Decimal Value	Signed binary 2's complement
-8	1 0 0 0		0	0 0 0 0
-7	1 0 0 1		1	0 0 0 1
-6	1 0 1 0		2	0 0 1 0
-5	1 0 1 1		3	0 0 1 1
-4	1 1 0 0		4	0 1 0 0
-3	1 1 0 1		5	0 1 0 1
-2	1 1 1 0		6	0 1 1 0
-1	1 1 1 1		7	0 1 1 1

Addition in the 2's Complement System

Assuming an 8 bits, including the sign-bit, 2’s complement system, consider:

Case 1: Addition of Two positive numbers

Dec.		Sign	Magnitude	
+13	→	0	0 0 0 1 1 0 1	(augend)
+11	→	0	0 0 0 1 0 1 1	(addend)
+24		0	0 0 1 1 0 0 0	(sum)

As expected, the sign-bit of the summation indicates a +ve result.

Case 2: Positive no. & smaller negative no.

Dec.		Sign	Magnitude	
+19	→	0	0 0 1 0 0 1 1	(augend)
-11	→	1	0 0 1 0 1 0 1	(addend)
+8		0	0 0 0 1 0 0 0	(sum)

A positive result occurs and this is indicated by the sign-bit of 0.

For 8 bits, including sign-bit, 2’s complement system,

Case 3: Addition of Positive number and larger negative number

Dec		sign	magnitude	
+27	→	0	0 0 1 1 0 1 1	
+14	→	0	0 0 0 1 1 1 0	(augend)
-27	→	1	1 1 0 0 1 0 1	(addend)
-13		1	1 1 1 0 0 1 1	(sum)

With a negative result and the magnitude is expressed in 2's complement form.

Case 4 Two negative numbers

Dec		sign	magnitude	
-14	→	1	1 1 1 0 0 1 0	(augend)
-7	→	1	1 1 1 1 0 0 1	(addend)
-21		1	1 1 0 1 0 1 1	(sum)

Subtraction in the 2's Complement System

Digital computers or processors do not normally employ hardware subtraction circuit.

The subtract operation is performed by the same hardware adder circuit using the addition function in the 2's complement system.

For example:

59_{10} - 69_{10} decimal subtraction is the same as

59_{10} + (-69_{10}) decimal addition

i.e. the operation can be performed using the addition function.

Using the 8-bits, 2's complement system, this yield the results:

	0 1 0 0 0 1 0 1	+69
	1 0 1 1 1 0 1 1	2’s comp of +69
	0 0 1 1 1 0 1 1	+59
+	1 0 1 1 1 0 1 1	-69
	1 1 1 1 0 1 1 0	(-10)

The result is negative and is in the 2’s complement form.

Arithmetic Overflow

When adding binary numbers the sum result *MUST* be in the range:

$$-2^{N} \text{ to } +(2^{N} - 1)$$

where N is the number of bits used for magnitude.

An arithmetic overflow occurs when the sum result is outside the range:

$$-2^{N} \text{ to } +(2^{N} - 1)$$

For Example in the 8-bits 2's complement system,

	Sign	Magnitude
(+87) →	0	1 0 1 0 1 1 1
(+43) →	0	0 1 0 1 0 1 1
(+130)	1	0 0 0 0 0 1 0

The results in this case is obviously erroneous because a +ve answer is expected

For 8-bits 2's complement system, the results of an addition must fall within the range of:

-128 to +127

BCD Addition

Addition of BCD numbers is generally performed similarly as in Binary addition except that the following two conditions apply:

Sum equals to less than 9

5	0 1 0 1	← BCD for 5
+ 4	+ 0 1 0 0	← BCD for 4
9	1 0 0 1	← BCD for 9

If the sum of an addition is less than or equal to 9, the sum result is left as it is with no adjustments.

Example

45	0 1 0 0	0 1 0 1	← BCD for 45
+ 33	+ 0 0 1 1	0 0 1 1	← BCD for 33
78	0 1 1 1	1 0 0 0	← BCD for 78

In this decimal addition no decimal carries were produced.

Sum greater than 9

6		0 1 1 0	← BCD for 6
+ 7		0 1 1 0	← BCD for 7
13		1 1 0 1	← Invalid BCD code. Adjustment required.
		+ 1 1 0	← Add BCD 6 to correct the results
	0 0 0 1	0 0 1 1	

Whenever a sum result is greater than 9, BCD 6 must be added to it so that a valid code group results.

Example

47	0 1 0 0	0 1 1 1	← BCD for 47
+35	0 0 1 1	0 1 0 1	← BCD for 35
82	0 1 1^{1}1	1 1 0 0	← Invalid sum for LSD
		1 1 0	← Add 6 for correction
82	1 0 0 0	0 0 1 0	← Correct BCD sum

Arithmetic Circuits

All Digital computers contain the basic functions to perform the arithmetic tasks of:

a) Addition

b) Subtraction

c) Multiplication

d) Division

The basic hardware circuit to perform *all* the above arithmetic functions is only the ***ADDER***.

With just the adder circuit,

Subtraction can be performed using the 2's complement method,

Multiplication through software algorithm of repeated addition and,

Division through the algorithm of repeated subtraction

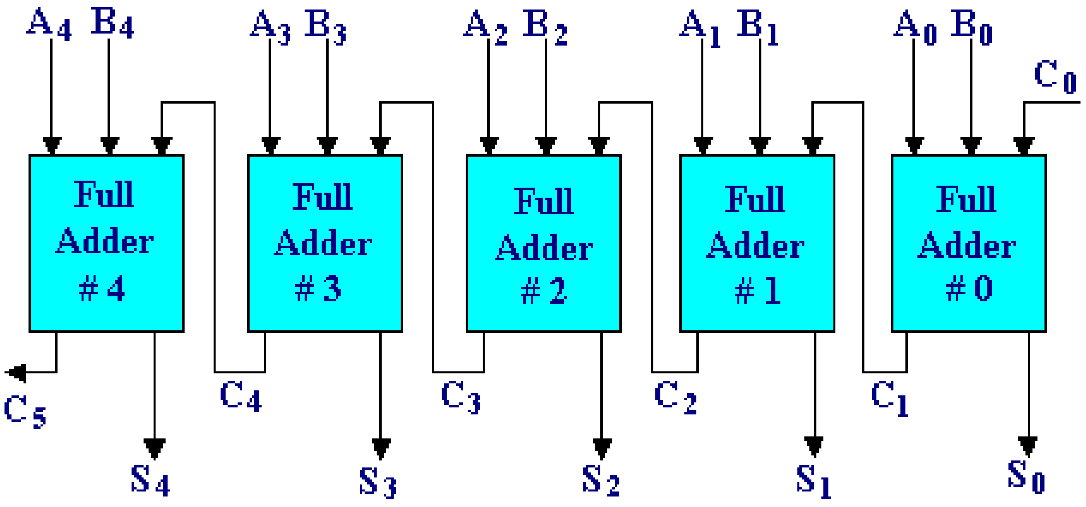
This Adder circuit is a major component found in the computer's Arithmetic Logic Unit or ALU, a main part of the CPU.

Parallel Binary Adder

Computers and calculators need to be able perform the binary addition of two numbers.

The circuit that allows this function to be achieved is called the parallel adder circuit. In the simplest form, this circuit consists of several Full-Adder units connected in cascade as illustrated below.

The arrangement is called a parallel adder because all bits of the augend (A) and addend (B) are fed into the adder simultaneously.

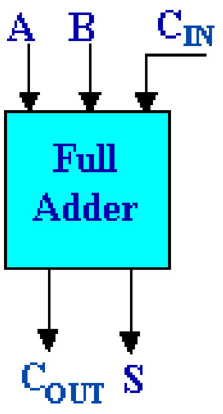


Block diagram of a 5-bit parallel adder circuit using full adders

Design of a Full Adder

To design the parallel Adder, start with the design of its component part called the Full Adder.

This is combinational logic device that adds its 3 inputs A, B and Cin (carry-in) to yield 2 outputs appropriately called Cout (Carry-out) and Sum.



Symbol and Truth-table of Full Adder

A	B	Cin	Cout	Sum
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

NB: A is the Augend, B is the Addend and Cin is Carry-input.

With 3 inputs, the FA truth table has a standard 8 input combinations.

The values of the outputs are obtained by adding all three values of the inputs A, B and Cin for each combination,
i.e. Cout Sum = A + B + Cin

E.g. Cout Sum = 1 1 when A=1, B=1 and Cin=1
Because 1+1+1 = 1 1 (or decimal 3).

Cout Sum

From the truth table, the Boolean expressions for the two outputs are:

$$\mathrm{Cout} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$\mathrm{Sum} = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

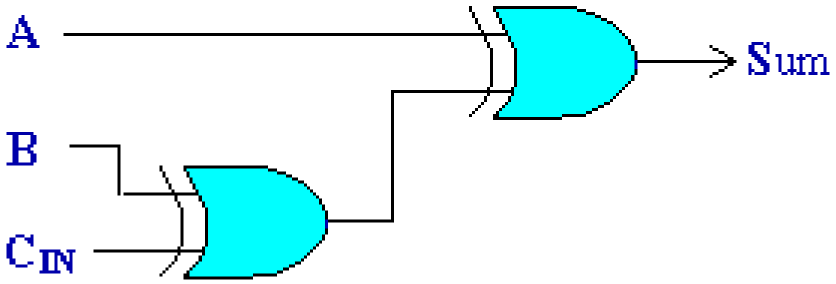
Using Boolean algebra to simplify the Sum equation,

$$\mathrm{Sum} = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

Collecting terms,

$$\begin{aligned}\mathrm{Sum} &= \overline{A}(\overline{B}C + BC) + A(\overline{B}\,\overline{C} + BC) \\ &= \overline{A}(B \oplus C) + A(\overline{B \oplus C}) \\ &= A \oplus (B \oplus C) \\ &= A \oplus B \oplus C\end{aligned}$$

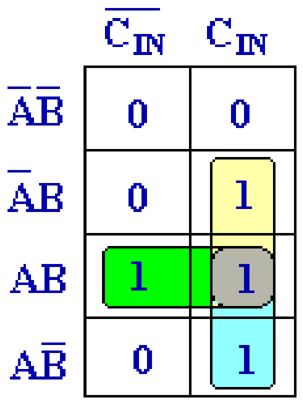
Implementing, the simplified sum equation,



To implement Cout, simplify the equation:

$$\text{Cout} = \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C$$

using either the K-map or Boolean algebra

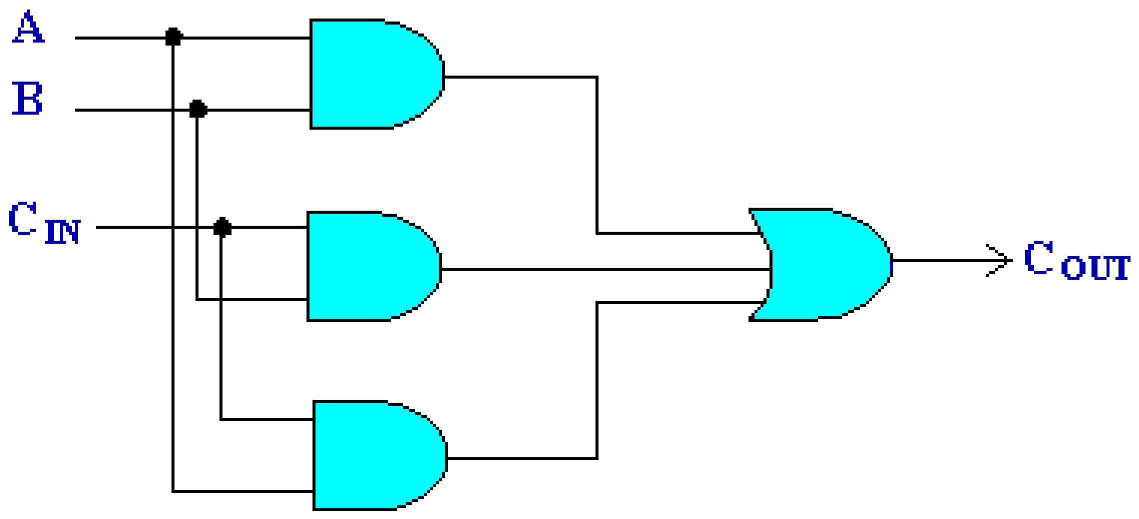


	$\overline{C_{IN}}$	C_{IN}
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	0	1
AB	1	1
$A\bar{B}$	0	1

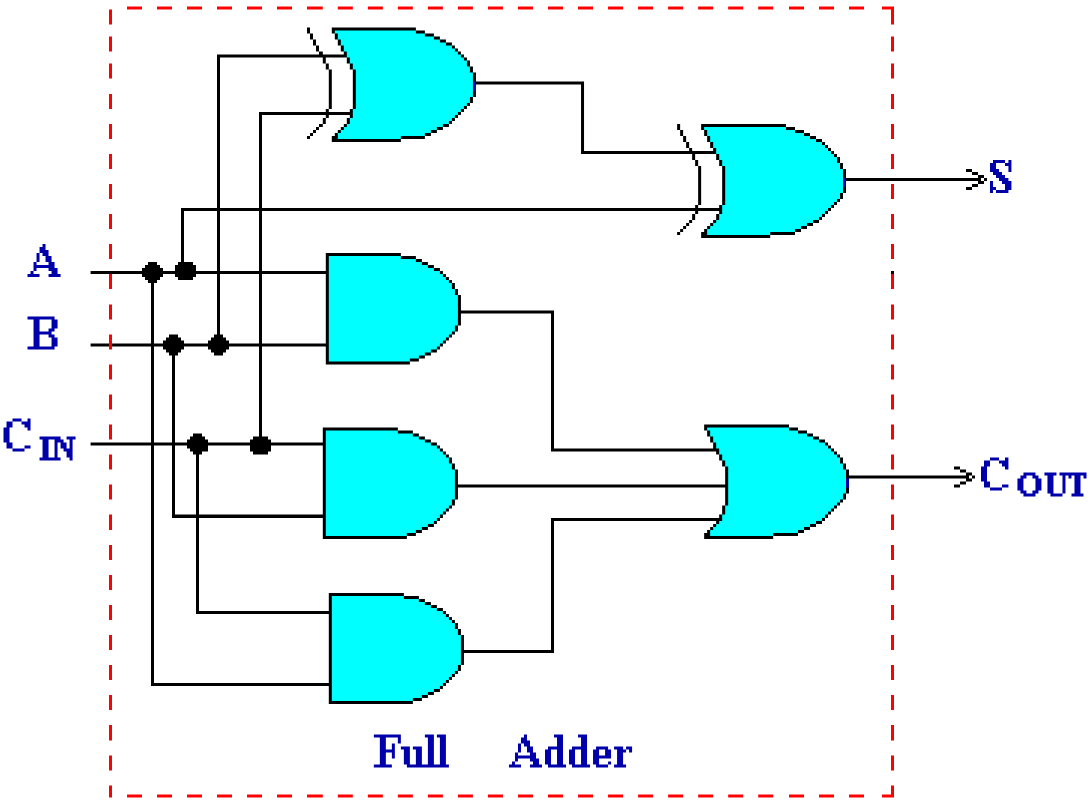
Using the K-map,
the simplified equation is:

Cout = A B + B Cin + A Cin

which when implemented, yields:



Combining the simplified circuits obtained for Sum and Cout, the circuit of the full adder is as shown below:



Replicating and cascading the correct number Full Adder units, a parallel adder (one that adds multiple bits at a time) can be easily built.

Question:
How many full adder units are required to build an 8 bit parallel adder?

Integrated Circuit Parallel Adder

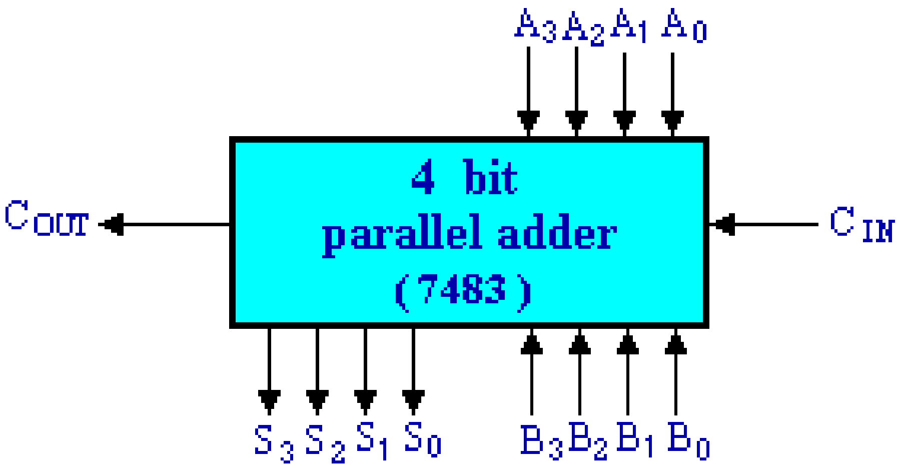
In practice, there is no necessity to construct a parallel adder from individual Full-Adder units as Integrated Circuit Parallel Adders are available.

An example is the 7483 4-bit parallel Adder IC.

It is called a 4-bit Adder as it is able to add two sets of 4-bit numbers at a time.

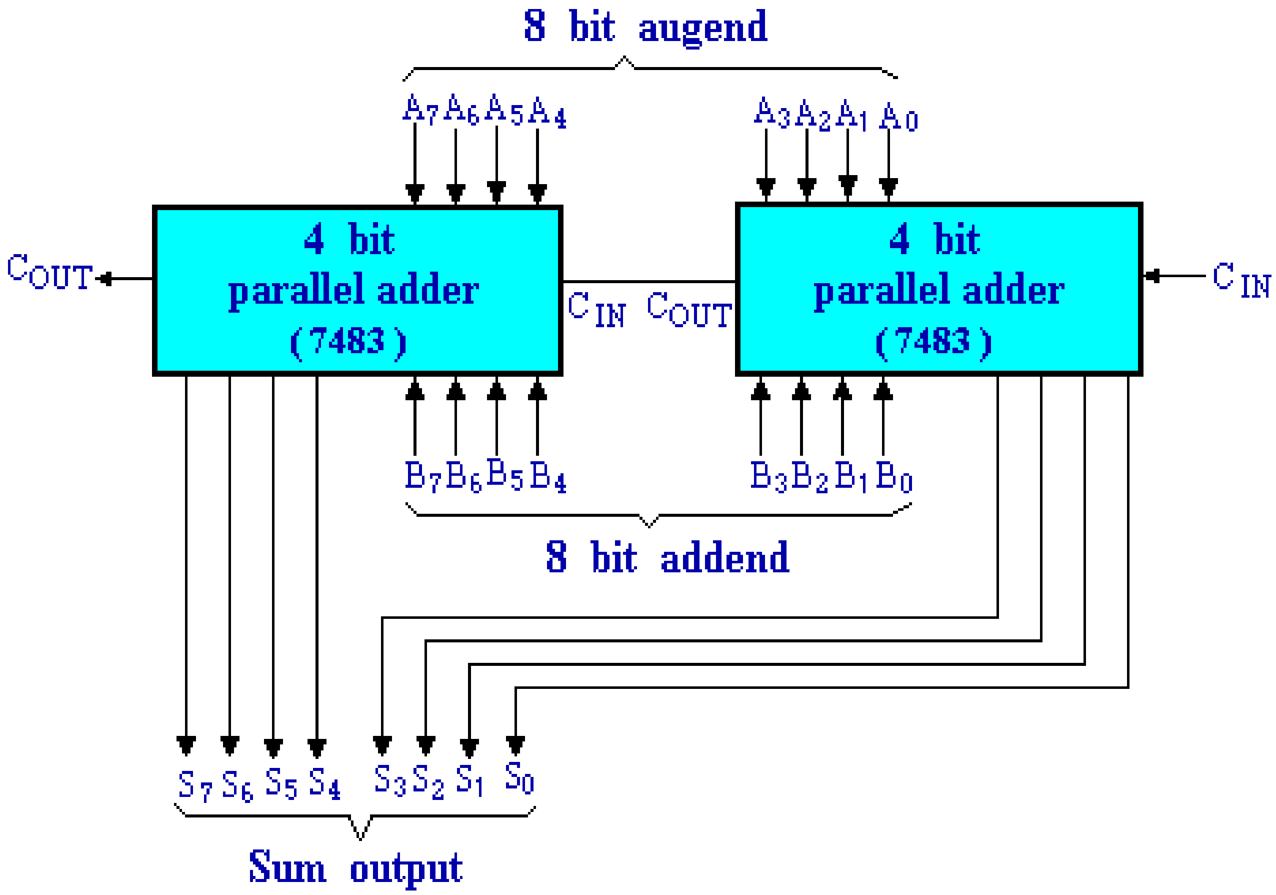
Internally, the 7483 comprises 4 Full-adder units connected in cascade.

Carry-input, Cin and carry-output, Cout are provided in the 7483 IC to allow multiple units to be cascaded to form a bigger parallel adder circuit.



NB: Subscript 0 denotes the LSB of each number

An example of an 8-bit adder circuit employing 2 units of the 7483 ICs is shown below:



Question:

How many 7483 ICs are needed to configure 32 bit Adder circuit?

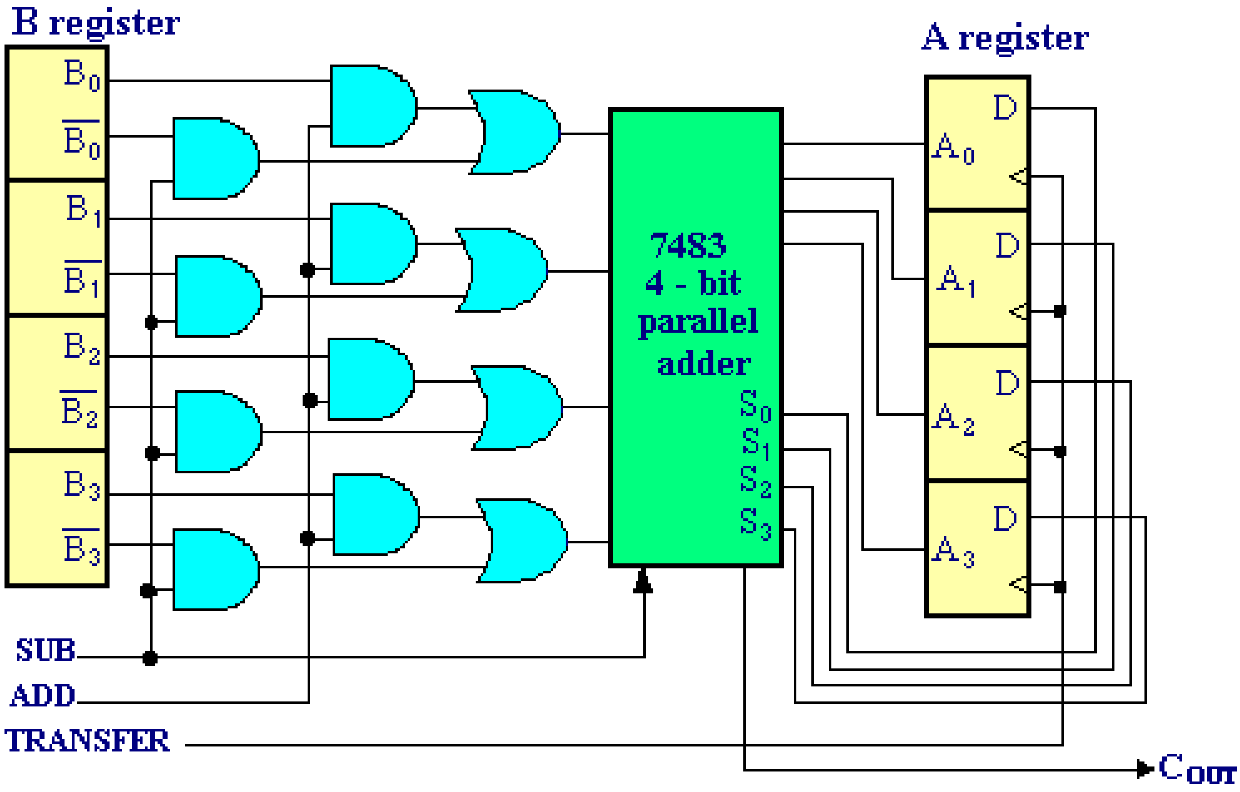
Answer this question by selecting (a) for 4 ICs, (b) for 6 ICs, (c) for 8 ICs or, (d) for 10 ICs

Combined Addition and Subtraction

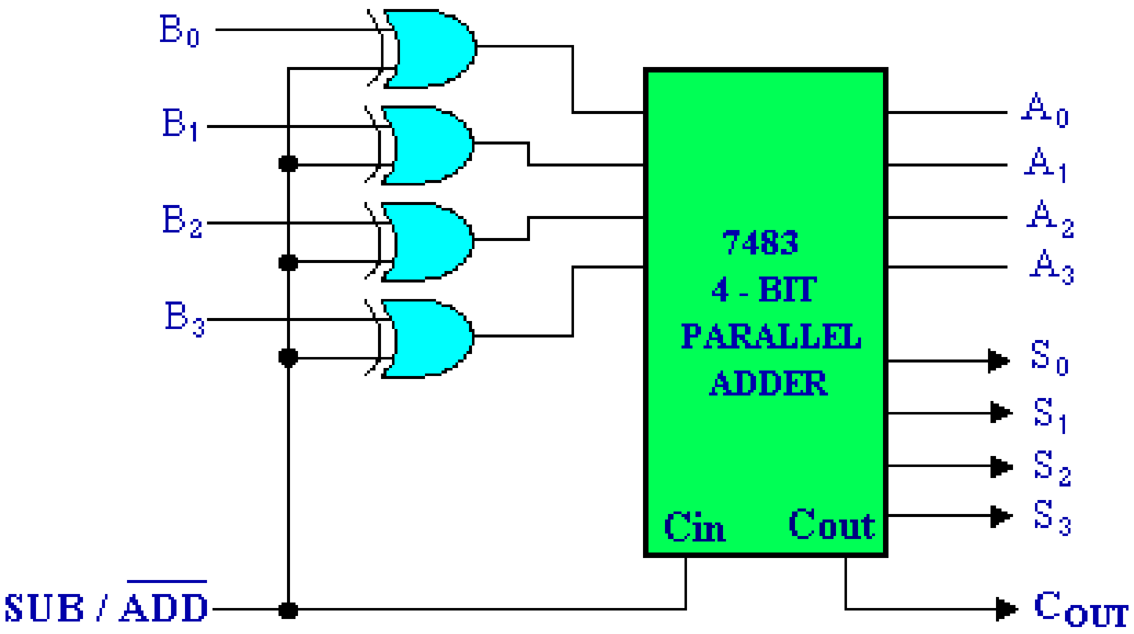
Using the 2's complement system, the 7483 can be configured to add as well as to subtract 2 numbers.
An example is the following circuit which provides the functions (A + B) or (A - B).

Two control inputs Add and Sub allows the selection of the operation.

(i) Sub = 1 Add = 0	→	the circuit functions as a Subtractor; output is A - B
(ii) Sub = 0 Add = 1	→	the circuit functions as an Adder; output is A + B



Another Example is one that uses XOR gates:



The XOR gates connected to the B inputs function as inverters or as buffers, i.e. without inversion, depending on the level at the $\text{SUB}/\overline{\text{ADD}}$ control input.

When $\text{SUB}/\overline{\text{ADD}}$ = 1, the XOR gates behave as inverters & the B inputs is complemented, with Cin = 1, the resultant is 2's complement. Hence the function performed is A + (2's complement of B), Or A + (–B) = A - B .

When $\text{SUB}/\overline{\text{ADD}}$ = 0, the circuit performs A + B.