## Parallel Binary Adder (or Ripple Carry Adder)

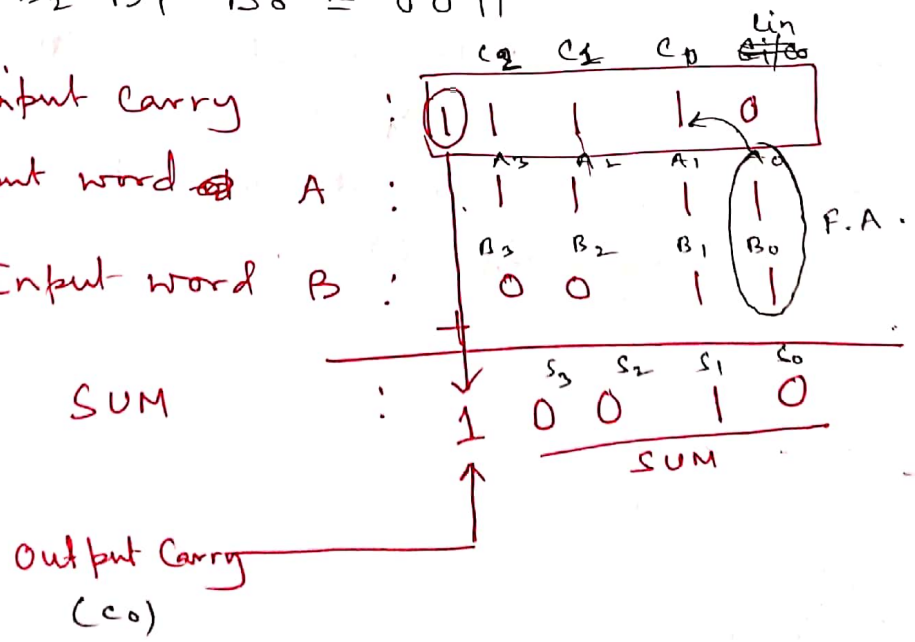
⇒ In most of the logic circuits, addition of more than one bit is carried out. As an example, modern computers and calculators use numbers ranging from 8 to 64 bits.

⇒ The 4-bit adder using full-adder circuits is capable of adding two - 4 bit numbers resulting in a 4-bit sum and a carry output as shown in figure below.

⇒ Since, all the bits of the inputs are fed into the adder circuits simultaneously and the additions in each position are taking place at the same time, the circuit is known as "Parallel Adder".

⇒ The Addition operation is illustrated in the following example: Lets the 4-bit words to be added be represented by – $A_3 A_2 A_1 A_0 = 1111$

and $B_3 B_2 B_1 B_0 = 0011$



So, 4-full adder circuit one can use for the

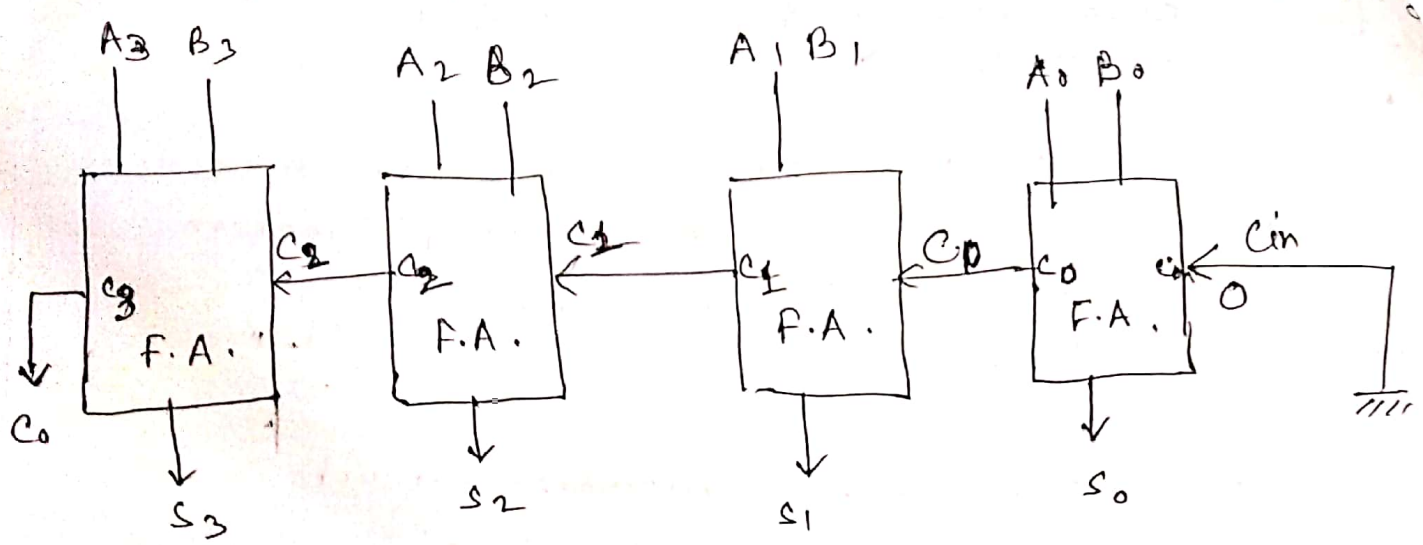



Fig:- 4-bit binary parallel Adder

=> Here Carry output lower order-stage is connected to the Carry input of the next higher order stage. Hence this type of adder is called "Ripple-Carry adder".

=> In the least significant stage, $A_0$, $B_0$ and $C_{in}$ are added resulting in sum $S_0$ and Carry $C_0$. This Carry $C_0$ becomes the Carry input to the second stage.

=> In the second stage $A_1$, $B_1$ and $C_0$ are added resulting in $S_1$ and $C_1$; simultaneously, after fourth stage, the circuit results in a sum ($S_3 S_2 S_1 S_0$) and a Carry output ($C_0$).

# 4 - bit Parallel Adder or Subtractor :-

=> The 4-bit parallel binary adder/subtractor Circuit is shown in figure below. It performs the operation of both addition and sub-traction.

=> It has two 4-bit inputs '$A_3 A_2 A_1 A_0$' and '$B_3 B_2 B_1 B_0$'. The Addition (ADD)/subtraction (SUB) Control line Connected with '$C_i$' of the least significant bit, is used to perform the operation of addition/subtraction.

=> The X-OR Gates are used as Controlled Inverters.

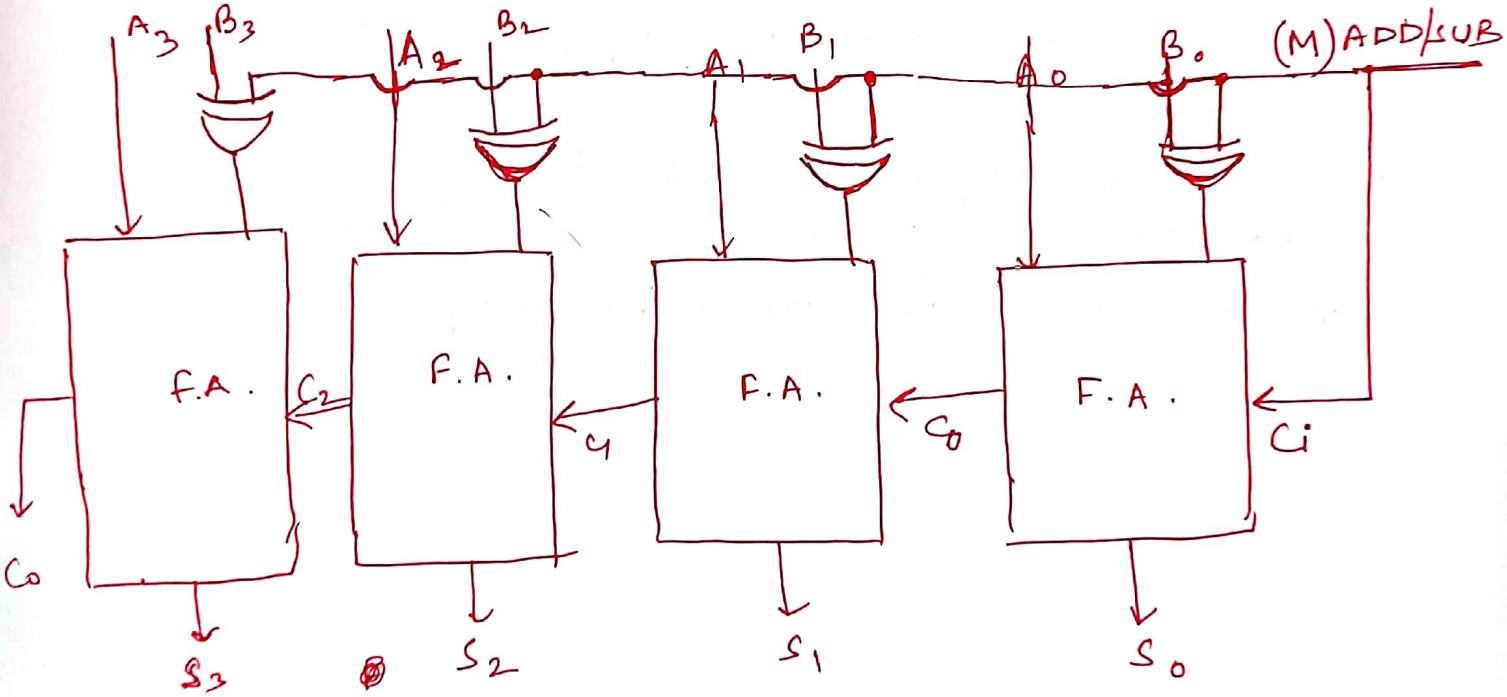



Fig :- 4-bit Parallel Adder/Subtractor

=> Though, the parallel binary adder is said to generate its output immediately after the inputs are applied, its speed of operation is limited by the carry propagation delay through all stages.

=> The propagation delay 'tp' of a full-adder (which has an inherent propagation delay) is the time difference between the instance at which the inputs ($A_i$, $B_i$ and $C_i$) are applied and the instance at which its outputs ($S_i$ and $C_{i+1}$) are generated.

Thus in a 4-bit parallel adder, where each full-adder has a propagation delay of 'tp', the output in the 4th-stage will be generated only after '4tp'.

=> One of the methods of speeding-up this process is "look-ahead carry" addition, which eliminates the ripple-carry delay.

=> It is very important to note that the EX-OR Gates are used as Controlled Inverters.

Say for Most Significant stage, the EX-OR Gate contains two input '$B_3$' & ADD/SUB Control (M).

So, $M \oplus B_3$

$= \overline{M} B_3 + M \overline{B_3}$

$\therefore$ If $M = 0$; $M \oplus B_3 = B_3$ —— (i)

$\therefore$ else if $M = 1$; $M \oplus B_3 = \overline{B_3}$ —— (ii)

=> To perform Addition, the ADD/SUB Control Input (M) is kept low.

=> To perform Subtraction, the ADD/SUB Control line is kept high.

- As, shown in eqn (ii), for the $M = 1$; the inverter produces the 1's Complement of the addend ($\overline{B_3}, \overline{B_2}, \overline{B_1}, \overline{B_0}$).

- Since '1' value also given to $C_i$ of the least significant bit of the adder, it is added to the Complemented addend producing 2's Complement (1's Complement + 1) of the addend before addition.

- Now the data $A_3 A_2 A_1 A_0$ is going to add with the 2's Complement of $B_3 B_2 B_2 B_1$; i.e; the Subtraction is going to take place. $C_o$ = the borrow output of 4-bit Subtractor.

# -: Carry Look Ahead Adder :-

## (Fast Adder)

=> In the parallel binary adder discussed earlier, the addition process can be considered to ~~be~~ completed only after the carry propagation delay through adders, which is proportional to number of stages in it.

=> One of the method of making the process faster is look ahead carry adder, which eliminates the ripple carry delay.

=> The carry look ahead adder is based on the principle of looking at the lower order bits of the augend and addend if a higher order carry is generated. This adder reduces the carry delay by reducing the number of gates through which a carry signal must propagate.

Table :- Truth Table of full-adder emphasizing the conditions at which carry generation occurs.

| Row | A | B | $C_i$ | S | $C_o$ | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | No carry generation. i.e; $C_o = 0$ |
| 1 | 0 | 0 | 1 | 1 | 0 | |
| 2 | 0 | 1 | 0 | 1 | 0 | Carry propagation $C_o = C_i$ |
| 3 | 0 | 1 | 1 | 0 | 1 | |
| 4 | 1 | 0 | 0 | 1 | 0 | |
| 5 | 1 | 0 | 1 | 0 | 1 | |
| 6 | 1 | 1 | 0 | 0 | 1 | Carry generation $C_o = 1$ |
| 7 | 1 | 1 | 1 | 1 | 1 | |

=> In rows - 0 and 1, the Carry output ($C_o$) is always zero and independant of Carry input ($C_i$), while in rows-6 & 7, the '$C_o$' is always '1' and independant of '$C_i$'. These are known as "Carry generations Combinations".

~~=> Let, 'Gi' represents the unity Carry (i.e; Co=1).~~

=> For rows - 2, 3, 4 and 5, the Carry output is equal to the Carry input. i.e; $C_o = C_i$.

These are Carry Propagate Conditions, (i.e; $C_o = 1$, only when $C_i = 1$)

=> Let '$G_i$' represents the unity Carry generate Condition and '$P_i$' represents the Carry Propagate Condition of the ith stage of a parallel adder.

=> From truth table, '$G_i$' is obtained by summing up the Combinations Corresponding to 6th & 7th rows as follows:-

$$G_i = A_i B_i C_i + A_i B_i \overline{C_i} = A_i B_i$$

Similarly, the Carry propagate Condition ($P_i$), ~~over~~ occurs when either $A_i = 1$ and $B_i = 0$ or vice-versa as shown in truth-table.

$$P_i = A_i \overline{B_i} + \overline{A_i} B_i = A_i \oplus B_i$$

** Consider, the addition of two binary numbers 'A' ($A_3 A_2 A_1 A_0$) and B ($B_3 B_2 B_1 B_0$).

The unity Carry output of the ith stage can be expressed in terms of '$G_i$', $P_i$ and $C_{i-1}$, which is the unity Carry output of the (i-1)th stage. This is written as follows :-

$$C_i \, (C_0) = G_i + P_i C_{i-1}$$

where, $C_{i-1}$ for the LSB stage is '$C_{in}$', which is assume to be zero.

=> In a 4bit binary adder, four stages of addition are required to add $A_0 B_0$, $A_1 B_1$, $A_2 B_2$ and $A_3 B_3$.

Therefore, for $i = 0, 1, 2, 3$; the '$C_i$' are given by- $C_0 = G_0 + P_0 C_{in}$ (where $G_0 = A_0 B_0$; $P_0 = A_0 \oplus B_0$) $C_{in} = 0$

$$\therefore \; C_1 = G_1 + P_1 C_0$$
$$= G_1 + P_1 (G_0 + P_0 C_{in})$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

[Here; $G_1 = A_1 B_1$, $P_1 = A_1 \oplus B_1$]

$\therefore C_2 = G_2 + P_2 C_1$ [where, $G_2 = A_2 B_2$]

$= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_{in})$

$= G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_{in}$

$\therefore C_3 = G_3 + P_3 C_2$ [where; $G_3 = A_3 B_3$, $P_3 = A_3 \oplus B_3$]

$= G_3 + P_3 (G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_{in})$

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_1 P_2 P_3 G_0 + P_0 P_1 P_2 P_3 C_{in}$

$\therefore$ The sum of 'A' and 'B' is given by -

$S = C_3 S_3 S_2 S_1 S_0$

where; $S_i = A_i \oplus B_i \oplus C_{i-1}$ for $i = 0, 1, 2 \ldots$

i.e; $S_0 = A_0 \oplus B_0 \oplus C_{in}$

$S_1 = A_1 \oplus B_1 \oplus C_0$

$S_2 = A_2 \oplus B_2 \oplus C_1$

$S_3 = A_3 \oplus B_3 \oplus C_2$

---

* Hence, from the eqn it is observed that to get the value of 'C3' or output of MSB side, one can directly calculate it from the value of 'Cin' or initial carry.

So, the time require for the operation is much lesser. That is why it is one of the fastest ~~method~~ adder used in practice.

## Serial Adder:-

=> Though the parallel adder performs the addition of two binary numbers at a relatively faster rate, the dis-advantage of the parallel addition is that it requires a large amount of Logic Circuitry. ~~This increases~~

=> The Logic Circuits increases in proportion with the number of bits being added.

=> In Serial adder, the addition operation is Carried out bit-by-bit. Therefore, Serial adder requires simpler Circuitry than a parallel adder but results in a low speed operation.

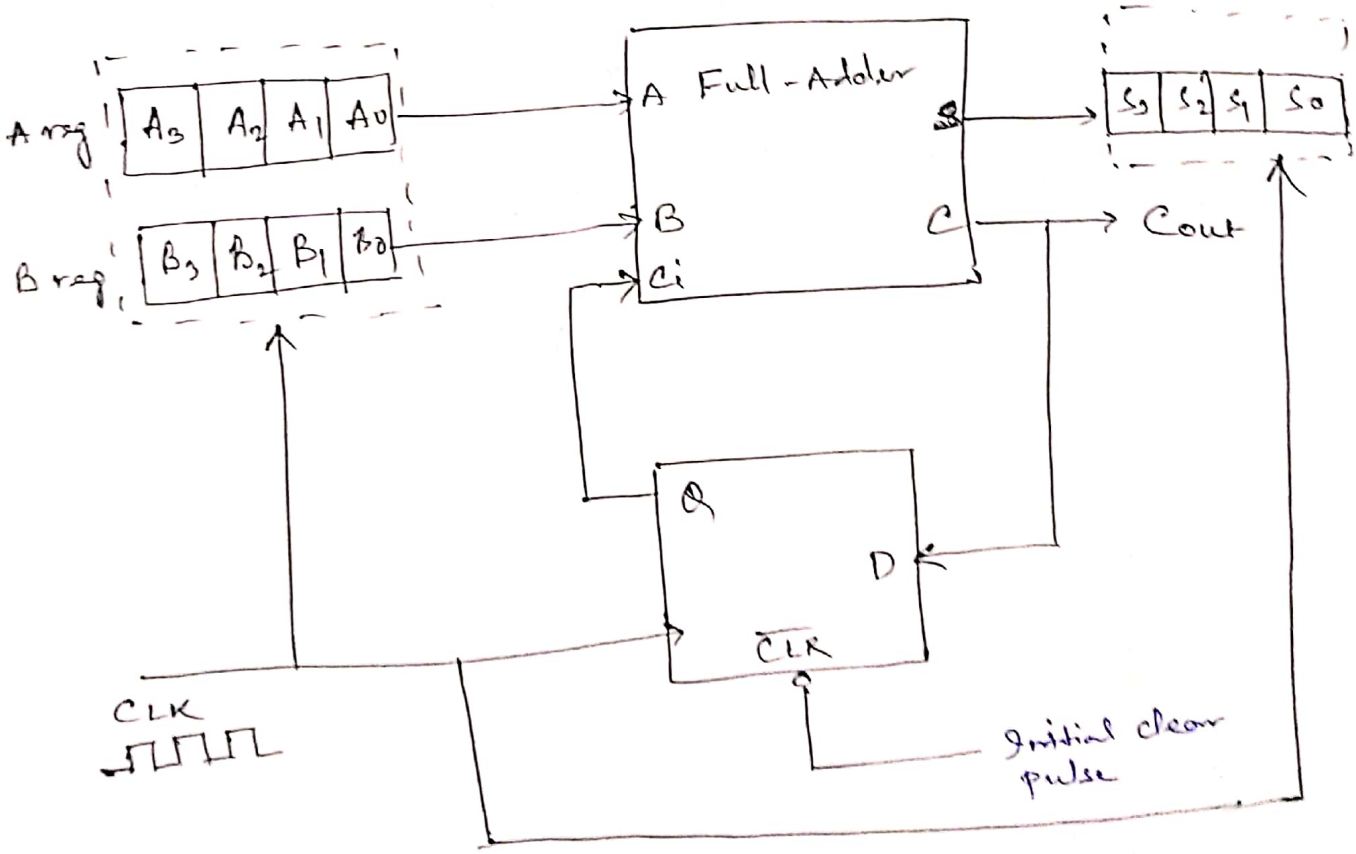



Fig:- 4-bit Serial Adder

=> The two shift registers 'A' & 'B' are used to store the numbers to be added serially. A single full-adder is used to add one-pair of bits at a time along with the carry.

The 'D' flip-flop, i.e; carry flip-flop is used to store the carry output of the full-adder so that it can be added to the next significant position of the numbers in the registers.

=> The contents of the shift registers shift from left to right and their outputs starting from $A_0$ and $B_0$ are fed into a single full-adder along with the output of the carry flip-flop upon application of each clock pulse

=> For each succeeding clock pulse, the contents of both the shift registers are shifted once to the right, ~~and~~ The new sum bit and new carry bit are transferred to sum-register and carry flip-flop respectively. This process continues un-till all the pairs of bits are added.