Combinational logic circuit

Unit-3 (Part-1)

→ A digital system consists of two types of circuits, namely (i) combinational logic circuit
(ii) Sequential logic circuit

→ In a combinational circuit, the output at any time depends only on the input values at that time. In a sequential circuit, the output at any time depends on the present input values as well as the past output values.

→ Examples of the combinational circuits are: adders, subtractors, multiplexer, De-multiplexer, encoder, decoder, parity bit generator, converter etc. Examples of sequential circuits are Latches, flip-flops registers, counters etc.

□ Procedure for the design of combinational circuits:-

Any combinational circuit can be designed by the following procedure:

1. From the word description of the problem, identify the inputs and outputs and draw the block diagram.
2. Draw the truth table such that it completely describes the operation of the circuit for different

combinations of inputs. Truth tables offer an easy means of representing any switching function of n variables.

3. Write down the switching expression(s) for the output(s)
4. Simplify the switching expression(s) using either algebric or K-map method.
5. Implement the simplified expression using logic gates.

Arithmetic circuits :~

The basic building blocks of the arithmetic unit in a digital computer are adders.

Half-adder :~

The ~~simpliest~~ simplest combinational circuit which performs the arithmetic addition of two binary digits is called a half-adder.

→ Two inputs are represented by A and B, and the two outputs are the sum (S) of A and B and the carry bit denoted by C.

(a) Logic symbol

→ From the truth table of the half-adder, one can understand that the sum output is 1 when either

of the inputs (A or B) is 1, and the carry output is 1 when both of the inputs (A and B) are 1.

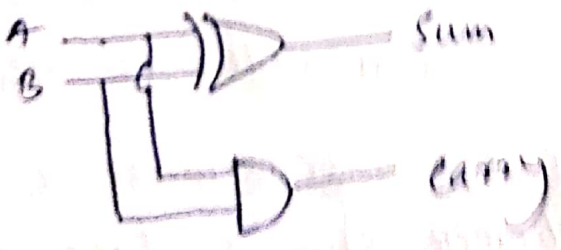
Inputs		outputs	
Augend A	Addend B	Sum S	carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

→ From the truth table, the logic expression for the sum output can be written as a sum of product expression by summing up the input combinations for which the sum is equal to 1.

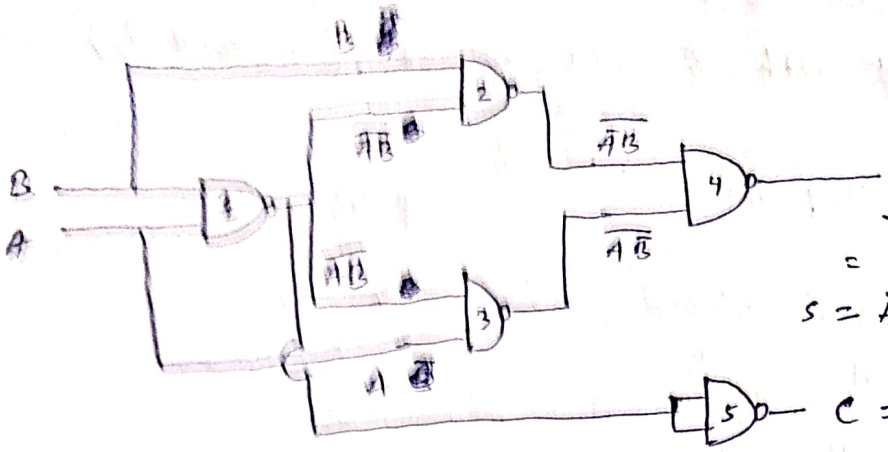
$$S = \bar{A}B + A\bar{B} = A \oplus B$$

and $$C = AB$$

→ The sum output corresponds to a logic Ex-OR function which the carry output corresponds to an AND function.



(b) Logic diagram



2 → $\overline{\overline{AB} \cdot B} = AB + \overline{B}$

$= A + \overline{B} = \overline{\overline{A}B}$

3 → $\overline{\overline{AB} \cdot A} = AB + \overline{A} = \overline{A} + B = \overline{A\overline{B}}$

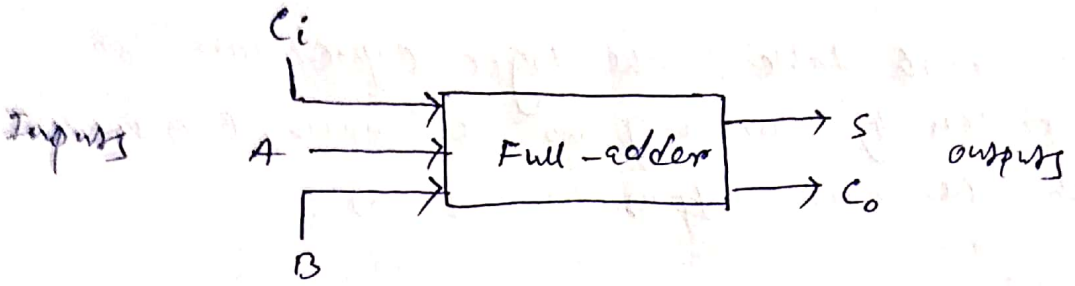
Fig c Realization using NAND gates.

→ Fig (b) and (c) gives the realization of the half-adder using minimum number of NAND gates.

□ Full-adder:

→ A half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multibit addition is performed. For this purpose, a full-adder is designed.

→ A full adder is a combinational circuit that performs the arithmetic sum of three input bits and produces a sum output and a carry.



2(a). logic symbol

→ The logic symbol of the full adder shown in fig 2(a). It consists of three inputs and two outputs. Two input variables denoted by A and B represent the two significant bits to be added.
The third input, C_i, represents the carry from the previous lower significant position.

→ The truth table for the full adder circuit is shown in table below.

Inputs			outputs	
Augend A	Addend B	Carry input C_i	Sum S	carry output C_o
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

→ From the truth table, the logic expression for S can be written by summing up the input combinations for which the sum output is 1 as:

$$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$$

Simplifying the above expression, we get

$$S = \bar{A}(\bar{B}C_i + B\bar{C}_i) + A(\bar{B}\bar{C}_i + BC_i)$$
$$= \bar{A}(B \oplus C_i) + A(\overline{B \oplus C_{in}})$$

Let $B \oplus C_{in} = X$

Now, $S = \bar{A}X + A\bar{X} = A \oplus X$

Replacing X by $B \oplus C_i$ in the above expression, we have

$$S = A \oplus B \oplus C_i$$

→ Similarly, the logic expression for C_o can be written by summing up the input combinations for which C_o is 1, as given below:

$$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$
$$= BC_i(A + \bar{A}) + A\bar{B}C_i + AB\bar{C}_i$$
$$= C_i(B + \bar{B}A) + AB\bar{C}_i$$
$$= C_i(B + A) + AB\bar{C}_i$$
$$= BC_i + AC_i + AB\bar{C}_i$$
$$= BC_i + A(C_i + \bar{C}_iB)$$
$$= BC_i + AC_i + AB$$

From the simplified expressions of S and C_o, the full-adder circuit can be implemented using one three-input XOR gate, three two-input AND gates and one 3-input OR gate as shown in fig 26

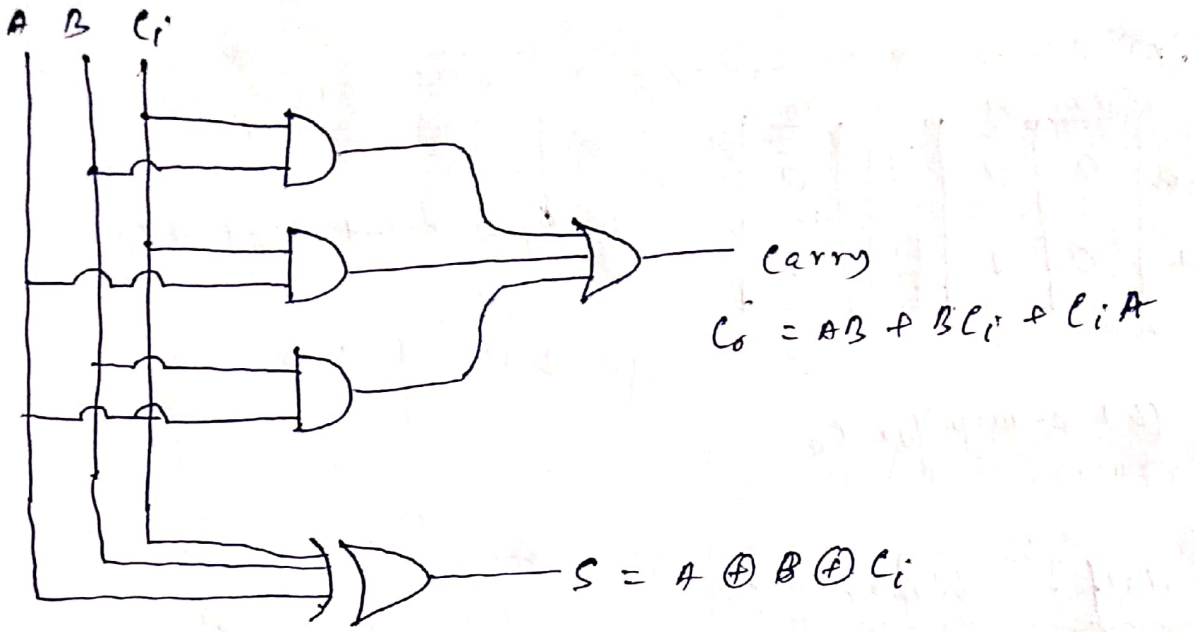


Fig 2b : logic diagram of full adder

→ A full-adder can be formed using two half adder circuits and an OR gate as shown in fig. 2c.

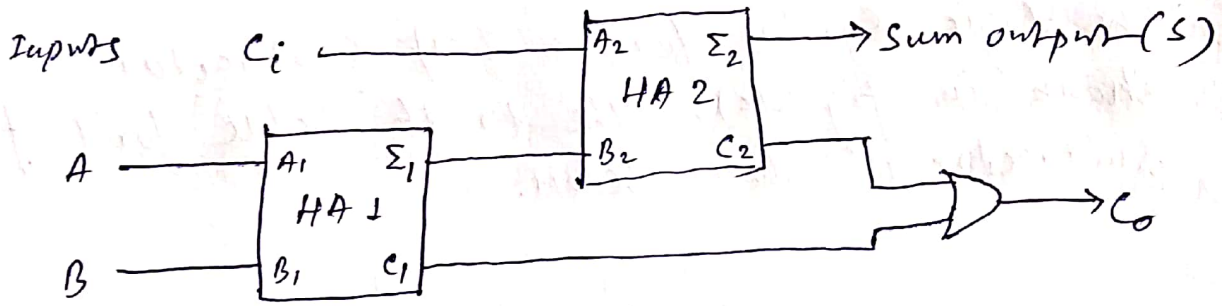
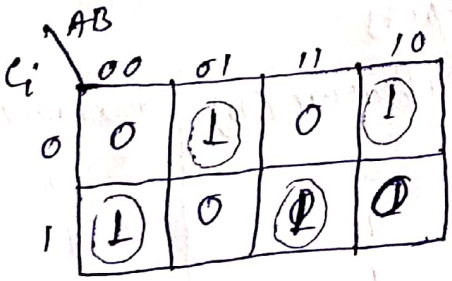


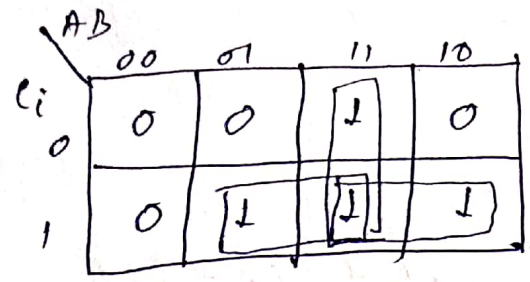
Fig 2 (c) Symbol using 2 half-adders.

→ K-map simplification : ✓



$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$

(a) K-map for sum



$$C_o = AB + BC_i + AC_i$$

(b) K-map for C_o

Half-Subtractor :-

→ The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, x (minuend) and y (subtrahend) and two outputs D (difference) and B_o (Borrow).

→ The logic symbol for a half-subtractor is shown in fig 3(a). The truth table for half-subtractor is shown below.

3(a): logic symbol

Inputs		outputs	
Minuend (x)	subtrahend (y)	Difference (D)	Borrow (B_o)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

→ From the truth table it is clear that the difference output is 0 if $x=y$ and 1 if $x \neq y$; the borrow output B_0 is 1 whenever $x=0$ and $y=1$. If x is less than y, then subtraction is done by borrowing 1 from the next higher order bit.

→ So, the boolean expression for difference and Borrow out can be written as follows:

$$D = \bar{x}y + x\bar{y} = x \oplus y$$

$$B_{out} = \bar{x}y$$

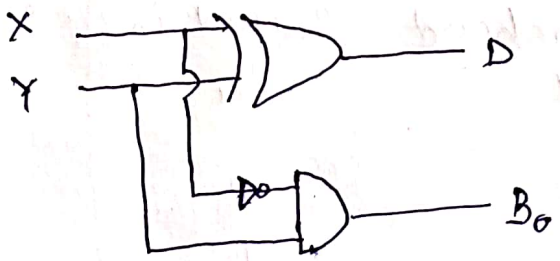


Fig 3(b) Logic diagram.

□ Full-Subtractor :-

→ A full-subtractor is a combinational circuit that performs subtraction involving three bits, namely minuend bit, subtrahend bit and the borrow from the previous stage. The logic symbol for full subtractor is shown in fig 4(a).

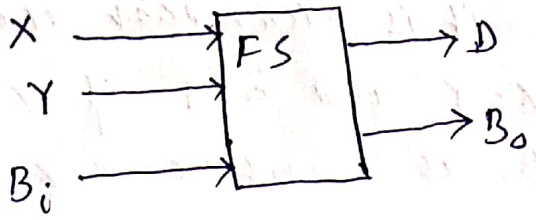


Fig 4(b) logic symbol

→ It has three inputs, X (minuend), Y (subtrahend) and B_i (borrow from the previous stage) and two outputs D (difference) and B_o (borrow out).

Table: Truth-table of full subtractor.

Inputs			Outputs	
Minuend X	Subtrahend Y	Borrow in B_i	Difference D	Borrow out (B_o)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From the table, the sum of product expression for the difference (D) output can be written as:

$$D = \bar{X}\bar{Y}B_i + \bar{X}Y\bar{B}_i + X\bar{Y}\bar{B}_i + XYB_i$$

Simplifying the above expression ~~for the~~

$$D = B_i(XY + \bar{X}\bar{Y}) + \bar{B}_i(X\bar{Y} + \bar{X}Y)$$

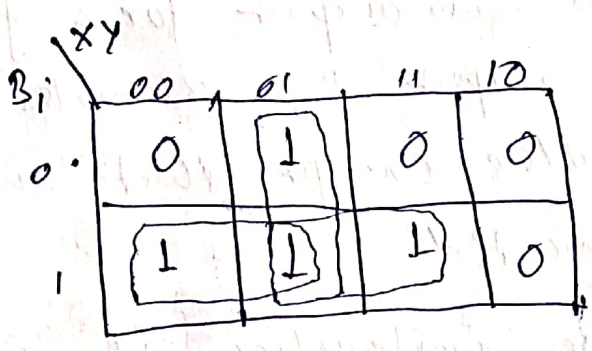
$$= B_i\overline{(X \oplus Y)} + \bar{B}_i(X \oplus Y)$$

$$= X \oplus Y \oplus B_i$$

→ Similarly, the sum of product expression for B_o can be written from the truth table as:

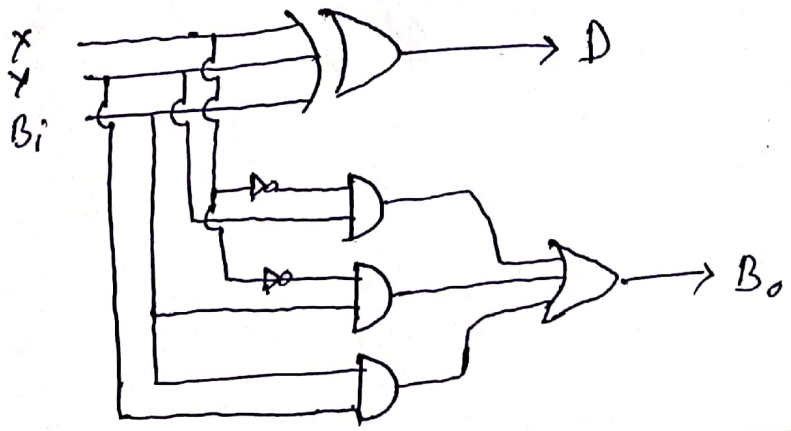
$$B_o = \bar{X}\bar{Y}B_i + \bar{X}Y\bar{B}_i + \bar{X}YB_i + XYB_i$$

The above equation can be simplify using K-map



$$B_o = \bar{X}Y + \bar{X}B_i + YB_i$$

Using the above simplified expressions, the full-subtracto can be realised as shown in below:



The full-subtractor can be implemented using two half-subtractors and an OR gate as shown in fig below:

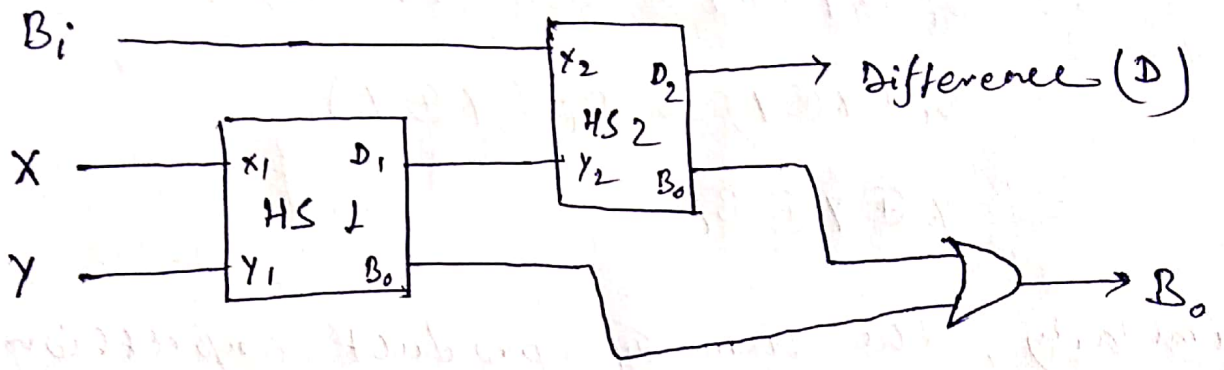


Fig 4(c): Full-Subtractor

→ One can notice that the equation for D is the same as the sum output for a full-adder, and the borrow output B_0 resembles the carry output for full-adder except that one of the inputs is complemented.

From these similarities, it is possible to convert a full-adder into a full subtractor.

Parallel binary adder (or Ripple-carry adder) :-

→ In most logic circuits, addition of more than 1-bit is carried out. For 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 fig 5(a).

→ Since all the bits of the augend and addend 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: Let the 4-bit words to be added be represented by $A_3A_2A_1A_0 = 1111$ and $B_3B_2B_1B_0 = 0011$

Input carry	:	1	1	1	0
Augend word A	:	1	1	1	1
Addend word B	:	0	0	1	1
		1	0	0	1	0 ← Sum
output carry ↑					

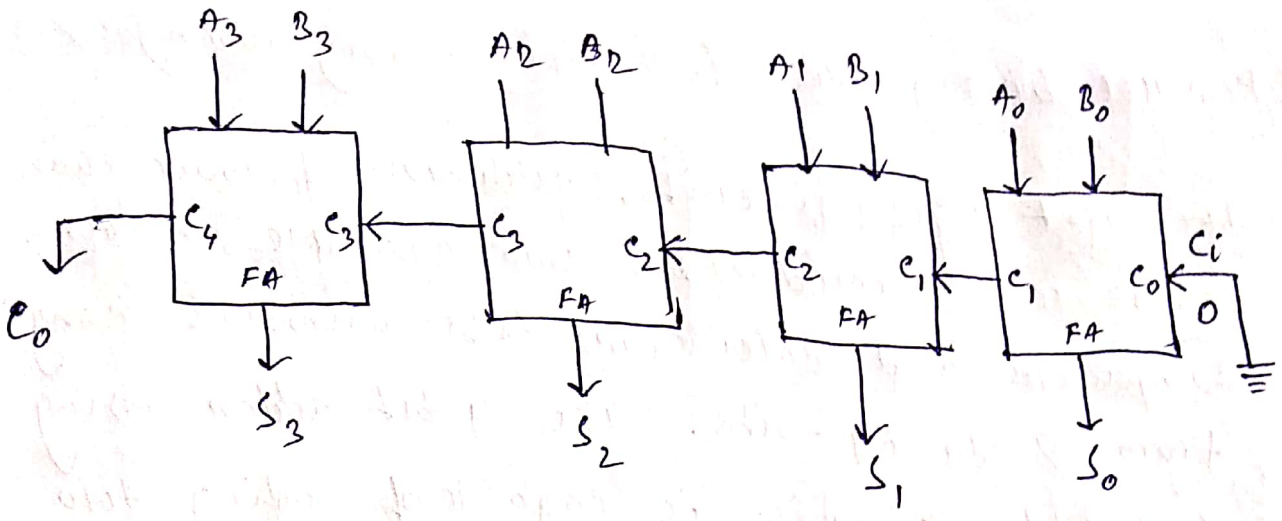


Fig 5(a): 4-binary parallel adder.

→ 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_0 are added resulting in sum S_0 and carry C_1. This carry C_1 becomes the carry input to the second stage.
In the second stage, A_1, B_1 and C_1 are added resulting in S_1 and C_2; similarly after fourth stage, the circuit results in a sum ($S_3 S_2 S_1 S_0$) and a carry out (C_0).

→ 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 t_p of a full-adder (which has an inherent propagation delay) is the time difference between the instant at which the inputs (A_i, B_i and C_i) are applied and the instant at which its outputs (S_i and C_{i+1}) are generated.

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

→ one of the methods of speeding up this process is look-ahead carry addition which eliminates the ripple-carry delay.

□ 4-bit parallel Adder / subtractor :-

→ The 4-bit parallel binary adder / subtractor circuit shown in fig 6, performs the operations of both addition and subtraction.

→ It has two 4-inputs $X_3X_2X_1X_0$ and $Y_3Y_2Y_1Y_0$. The ADD/SUB control line connected with C_i of the least significant bit of the full-adder, is used to perform the operation of addition and subtraction.

The 5 XOR gates are used as controlled inverter

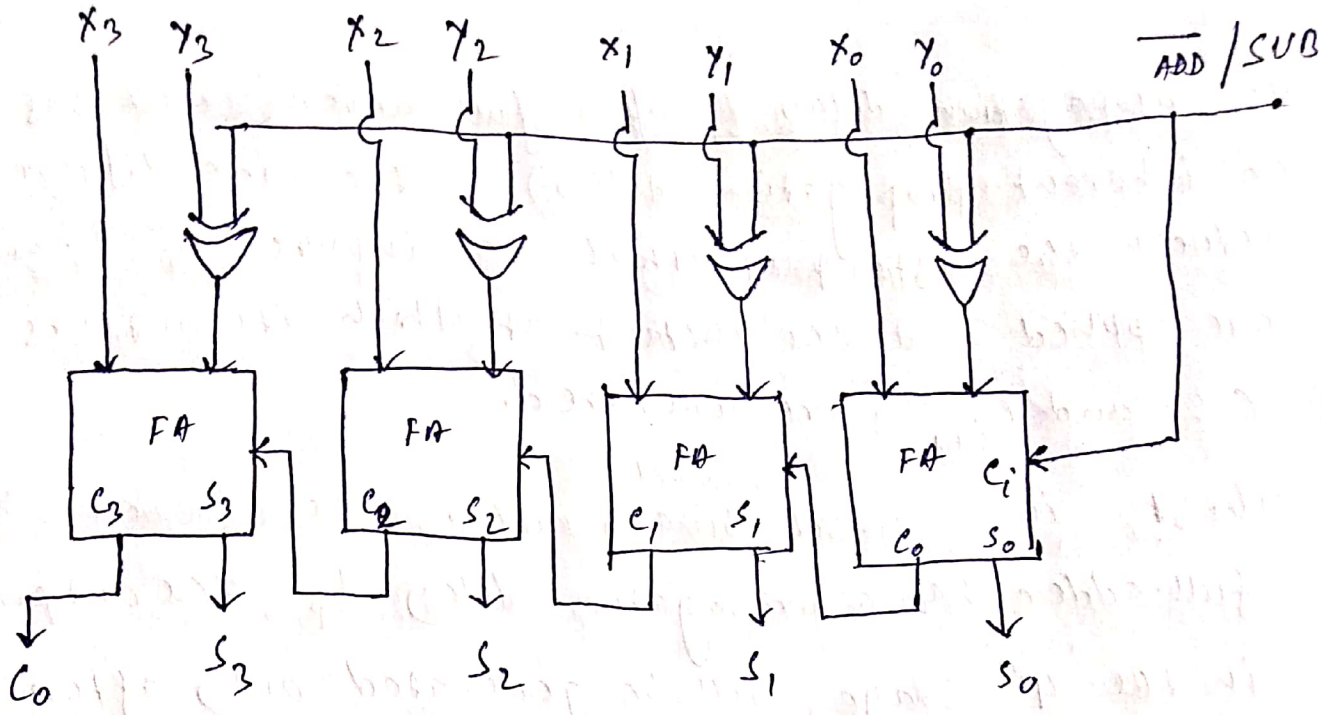


Fig 6: 4-bit parallel adder/subtractor

→ To perform subtraction, the $\overline{ADD}$/SUB control input is kept high. Now the controlled inverter produces the 1's complement of the addend ($\bar{Y}_3 \bar{Y}_2 \bar{Y}_1 \bar{Y}_0$). Since 1 is given to C_i of the least significant bit of the adder, it is added to the complemented addend producing 2's complement of the addend before addition.

Now, the data $X_3 X_2 X_1 X_0$ will be added to the 2's complement of $Y_3 Y_2 Y_1 Y_0$ to produce the sum, i.e. the difference between the addend and the augend, and C_0 i.e. the borrow output of 4-bit subtractor.

Also, it has $S_3 S_2 S_1 S_0$ as sum output and C_0 as carry output.

→ When $\overline{ADD}/SUB$ input is LOW, the controlled inverter allows the addend $(Y_3Y_2Y_1Y_0)$ without any change to the input of the full-adder, and the carry input c_i of the least significant bit of full-adder, becomes zero. Now, the augend $(X_3X_2X_1X_0)$ and addend $(Y_3Y_2Y_1Y_0)$ are added with $c_i = 0$. Hence, the circuit functions as a 4-bit adder resulting in sum $S_3S_2S_1S_0$ and carry c_o.

☐ Fast adder : Carry look ahead adder :-

→ In the parallel binary adder discussed earlier the addition process can be considered to be complete only after the ~~above~~ carry propagation delay through adders, which is proportional to number of stages in it.
One of the methods of making this process faster is look ahead carry addition, 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 high order carry is generated. This adder reduces the carry delay by reducing the number of gates through which a carry signal must propagate.

→ Consider the truth table of full-adder,

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 i.e. $c_o = c_i$
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 i.e. $c_o = 1$
7	1	1	1	1	1	

In rows 0 and 1, the carry output (c_o) is always zero and independent of carry input (c_i), while in rows 6 and 7, the c_o is always 1 and independent of c_i. These are known as carry generate combinations.

In 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 combinations (i.e. $c_o = 1$ only when $c_i = 1$).

→ Let G_i represents the unity carry (i.e. $c_o = 1$) generate condition and P_i represents the carry propagate condition of the i^{th} stage of a parallel adder.

→ From the truth table, G_i is obtained by summing up the combinations corresponding to 6th and 7th rows as follows:

$$G_i = A_i B_i C_{i-1} + A_i B_i \overline{C_{i-1}} = A_i B_i$$

Similarly the carry propagation condition (P_i) 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 4-bit 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 i^{th} 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 as follows:

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

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

~~The Sum of A and B is given by~~

→ In a 4-bit 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 $G_0 = A_0 B_0$; $P_0 = A_0 \oplus B_0$ and $C_{in} = 0$

$C_1 = G_1 + P_1 C_0$

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

$= G_1 + P_1 G_0 + P_1 P_0 C_{in}$ ..... where $G_1 = A_1 B_1$

and $P_1 = A_1 \oplus B_1$

$C_2 = G_2 + P_2 C_1$

$=$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$ .... where $G_2 = A_2 B_2$

and $P_2 = A_2 \oplus B_2$

$C_3 = G_3 + P_3 C_2$

$= \ldots$

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

where $G_3 = A_3 B_3$ and

$P_3 = A_3 \oplus B_3$

The sum of A and B is given by

$S = C_2 S_3 S_2 S_1 S_0$ where

$S_i = A_i \oplus B_i C_{i-1}$ for $i = 0, 1, 2, 3$

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

$S_1 = A_1 \oplus B_1 \oplus C_0$

$S_2 = A_2 \oplus B_2 \oplus C_1$

$S_3 = A_3 \oplus B_3 \oplus C_2$

□ Serial adder :-

→ Though the parallel adder performs the addition of two binary numbers at a relatively faster rate, the disadvantage of the parallel addition is that it requires a large amount of logic circuitry. This increases the direct proportion with the number of bits in the numbers being added.

→ On the other hand, in serial addition, the addition operation is carried out bit-by-bit. Therefore, the serial adder requires simpler circuitry than a parallel adder but results in a low speed of operation.

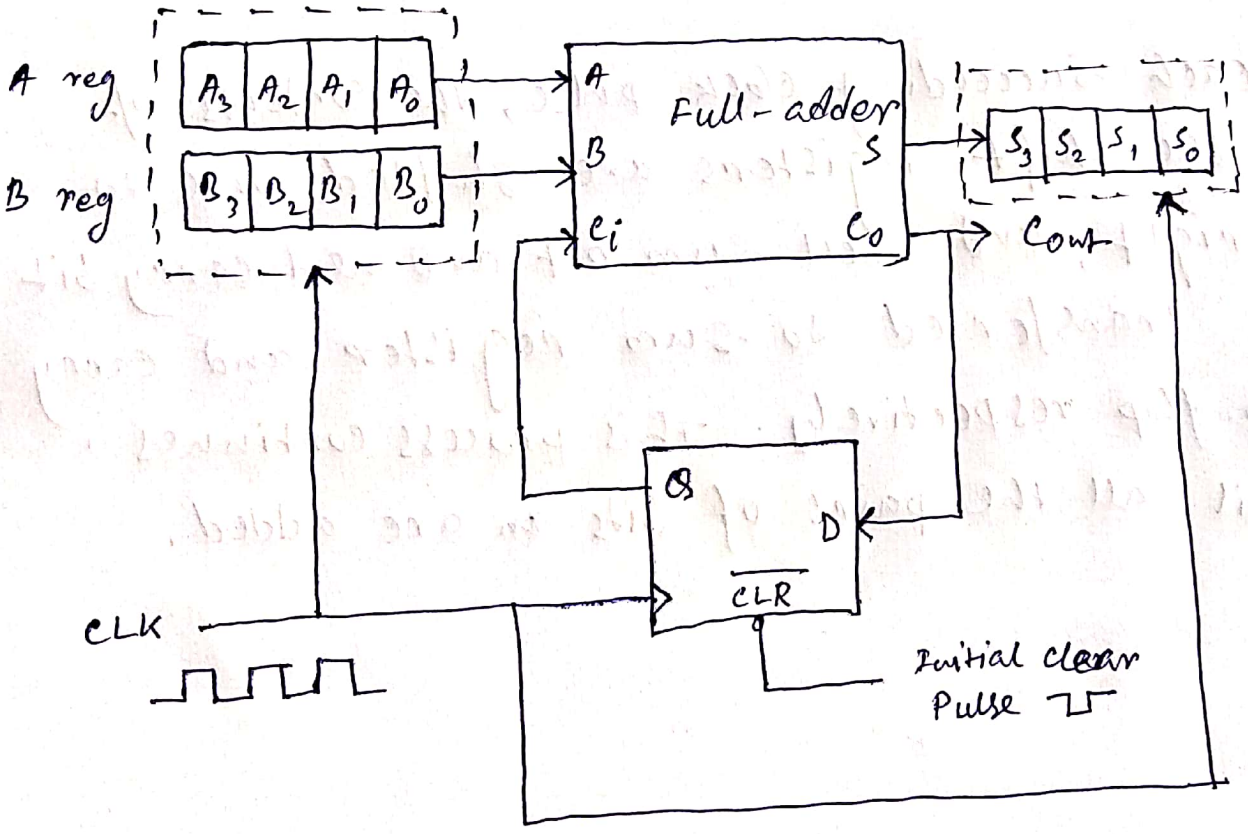


Fig: 4-bit serial adder.

→ The two shift registers A and 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 new sum bit and new carry bit are transferred to sum register and carry flip-flop respectively. This process continues until all the pairs of bits an are added.

—