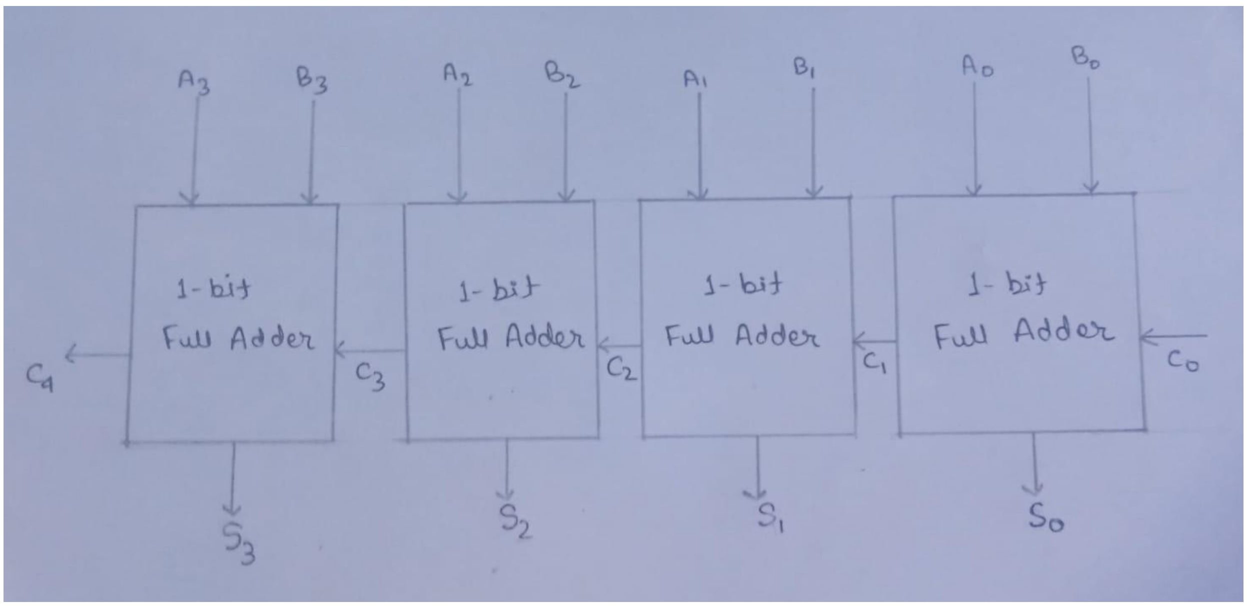
8-bit Carry Look-Ahead Adder

- **Aim + Motivation**

First of all, we realize the need of the CLA. We have already seen the 8-bit ripple carry adder which performs additions on two 8-bit binary numbers A and B, and outputs the sum and carry out depending upon the value of C_{in}. But, there is a problem with the ripple carry adder.

Consider the following 4-bit ripple carry adder:



Through the diagram, it is clear that the value of C_i depends upon values of C_{i-1}. (For ex., C_2 depends on C_1 which depends upon C_0). Also, the values of S_0, S_1, S_2, and S_3 will also depend upon the carry values. Hence, to calculate the subsequent carry values, the circuit will take some time which is called the propagation time. In other words, the carry values would have to propagate through each of the intermediate stages for the final sum and carry value to be calculated.

This propagation time depends upon the number of full-adders and the propagation delay caused by each adder. Hence, the delay will keep increasing if we move to more complex circuit architectures.

Hence, we want to design a circuit which will reduce this propagation delay and the time to calculate the final outputs.

- **Theory + Working**

Consider the following table which contains the values of C_{out} for different values of A, B and C_{in}:

A	**B**	**C_{in}**	**C_{out}**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- Consider the last two rows. Value of C_{out} depends only on A and B. It can be given by A.B
- Consider the middle 4 rows. Value of C_{out} can be given by $(A\oplus B).C_{in}$

In general, the overall table can be formulated as following:

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

We next define two new terms:
G :- Carry Generate = A.B
P :- Carry Propagate = $(A\oplus B)$

Hence, we can write the above formula as $C_{out} = G + P.C_{in}$

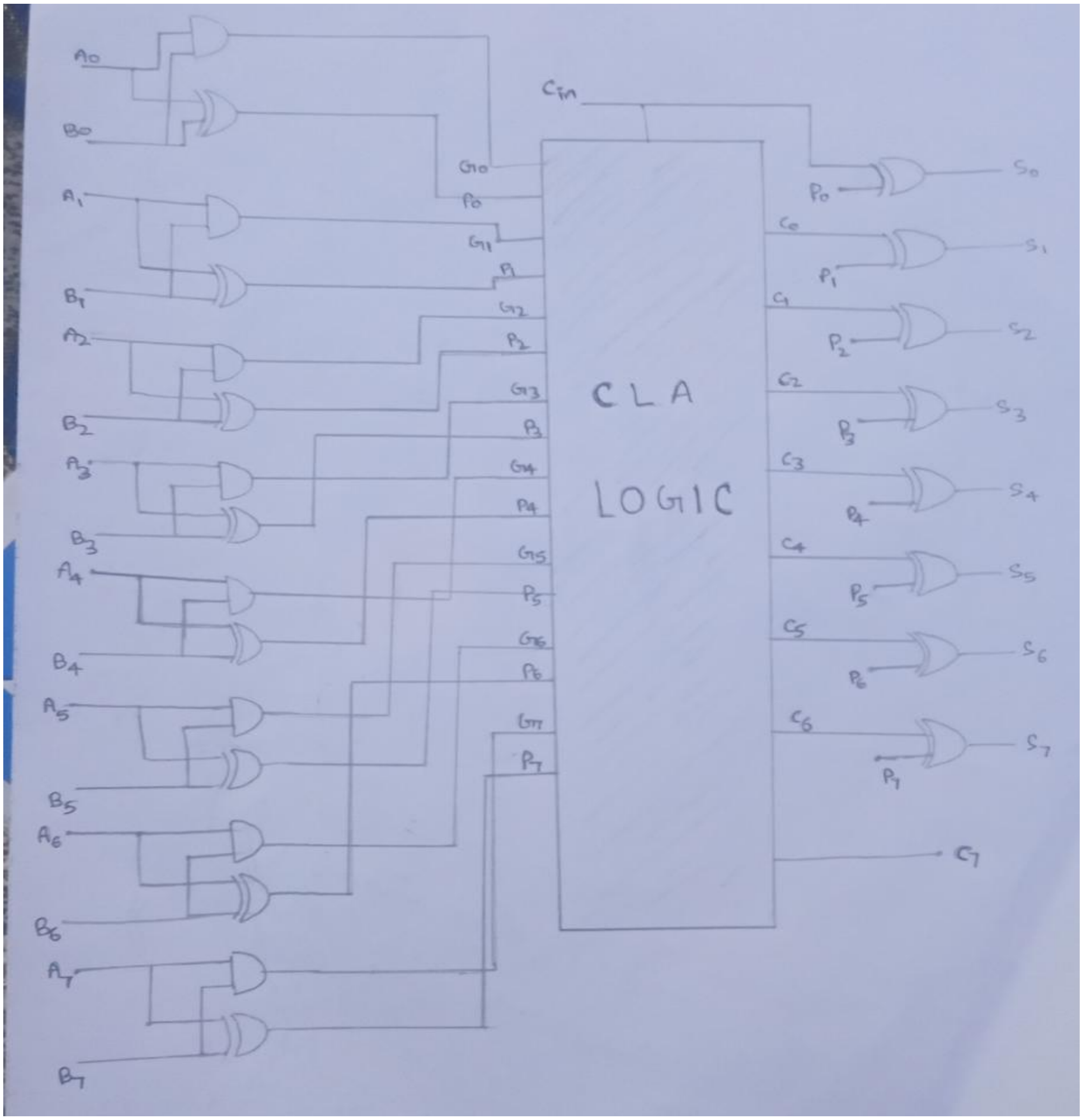
For any adder in general, we can write :-

$$\boxed{C_i = G_i + P_i.C_{i-1}}$$

For example, for a 8-bit adder, we will have
$C_1 = G_1 + P_1.C_{-1}$ (here, $C_{-1} = C_{in}$)
$C_2 = G_2 + P_2.C_1 = G_2 + P_2(G_1 + P_1.C_{-1}) = G_2 + P_2.G_1 + P_2.P_1.C_{-1}$
$C_3 = G_3 + P_3.C_2 = G_3 + P_3.(G_2 + P_2.G_1 + P_2.P_1.C_{-1}) = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.C_{-1}$
And so on…..

Hence, the above equations prove that we need not calculate intermediate carry values to get the final outputs. We just need to calculate G and P for the given inputs and then perform the above operations to calculate the final outputs.

- **Diagram**



- **Advantages**
    - ❖ We are able to avoid the large propagation delay caused due to the sequential propagation of carry values through the intermediate adders.

- **Limitations/ Drawbacks**
    - ❖ More hardware is required and this can grow exponentially once the size of the circuit increases.

8-bit Johnson Counter

- **Aim + Motivation**
  Johnson counter is basically a counter used to count the number of times an event occurs within the circuit. It's a sequential logic circuit used to count several pulses. It is designed using a clock signal and a group of flip flops where the complemented output of the last flip-flop is fed into the input of the first flip flop. Usually, it is implemented using D or JK flip flops.

- **Truth Table:** Truth Table for 8-bit Johnson Counter is as follows:

CLK PULSE	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	1	0	0	0
6	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1
10	0	0	1	1	1	1	1	1
11	0	0	0	1	1	1	1	1
12	0	0	0	0	1	1	1	1
13	0	0	0	0	0	1	1	1
14	0	0	0	0	0	0	1	1
15	0	0	0	0	0	0	0	1

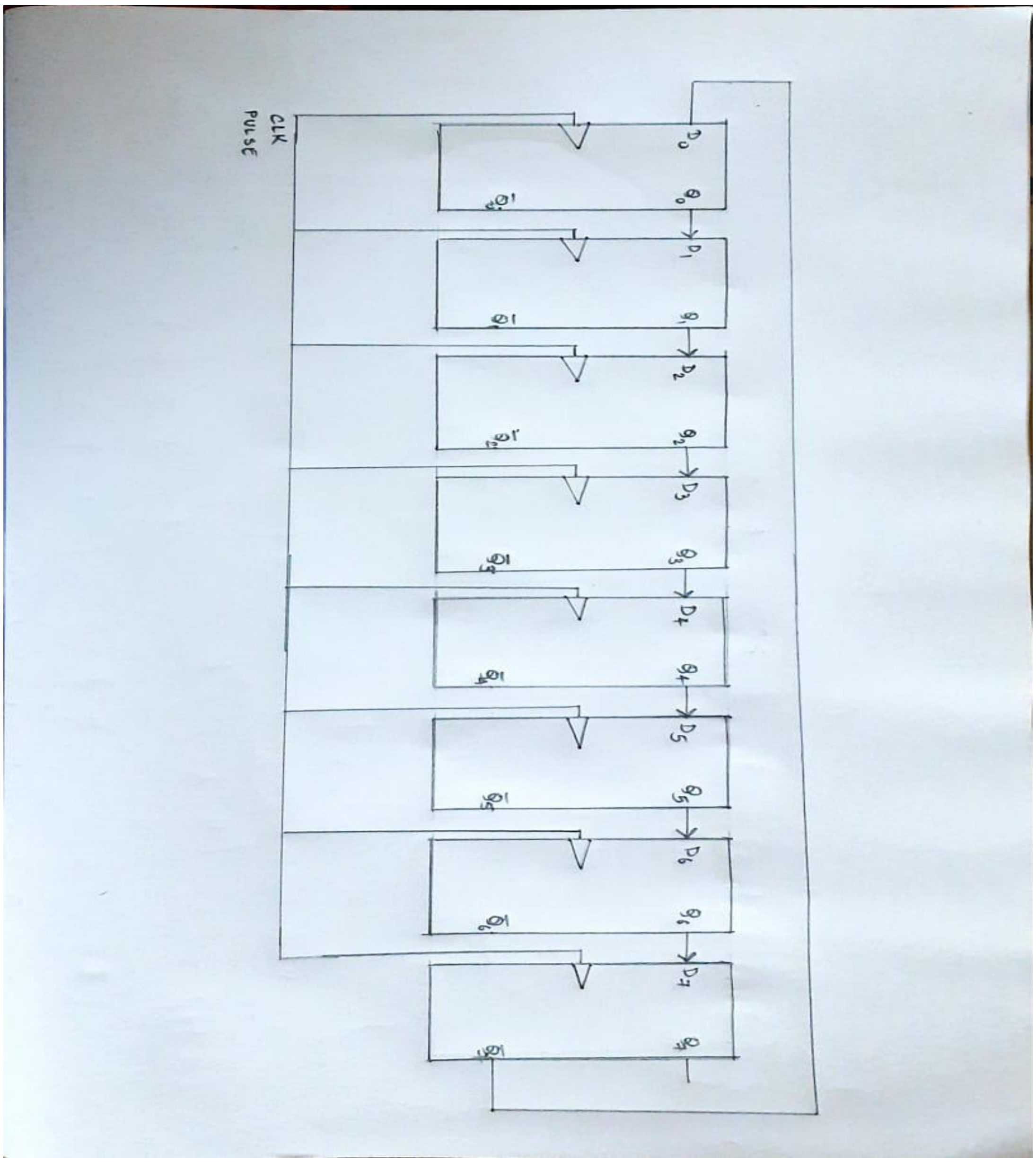
Where, Q_n's are the states.

The total number of used states in 8-bit Johnson counter are = 2n
= 2*8
= 16

And,

The total number of unused states in 8-bit Johnson counter are = 2^n - 2n
= 2^8 - 2*8
= 256 - 16
= 240

- **Circuit Diagram:**
  Implementation of Johnson counter using D-flip flops:



- **Advantages**
    - ❖ Johnson Counter is a self-decoding circuit.
    - ❖ It has an advantage over the ring counter i.e. despite using the same number of flip flops, the johnson counter can count twice the number of states the ring counter can count. Therefore, it’s considered a “mod 2n counter”.
    - ❖ It counts events in a continuous closed loop.
- **Disadvantages**
    - ❖ A lot of states remain unused in the johnson counter.
    - ❖ It doesn’t count in binary sequence.