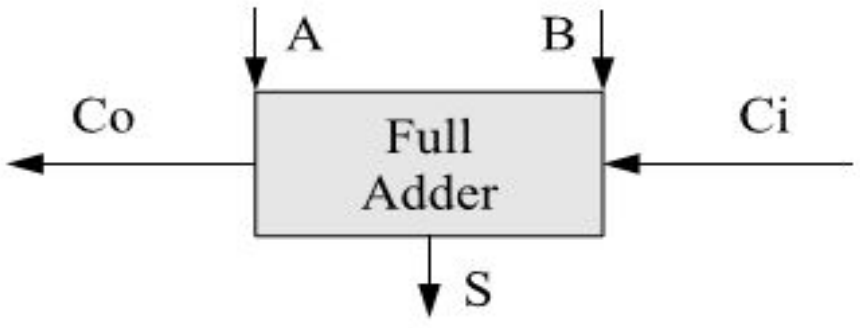
Adders

Adders

- A circuit that can add just two digits is called a half adder. A full adder is one that can add together two bits and a carry from the previous addition. A binary addition will require two , registers to hold the two data and a circuit to perform the addition.



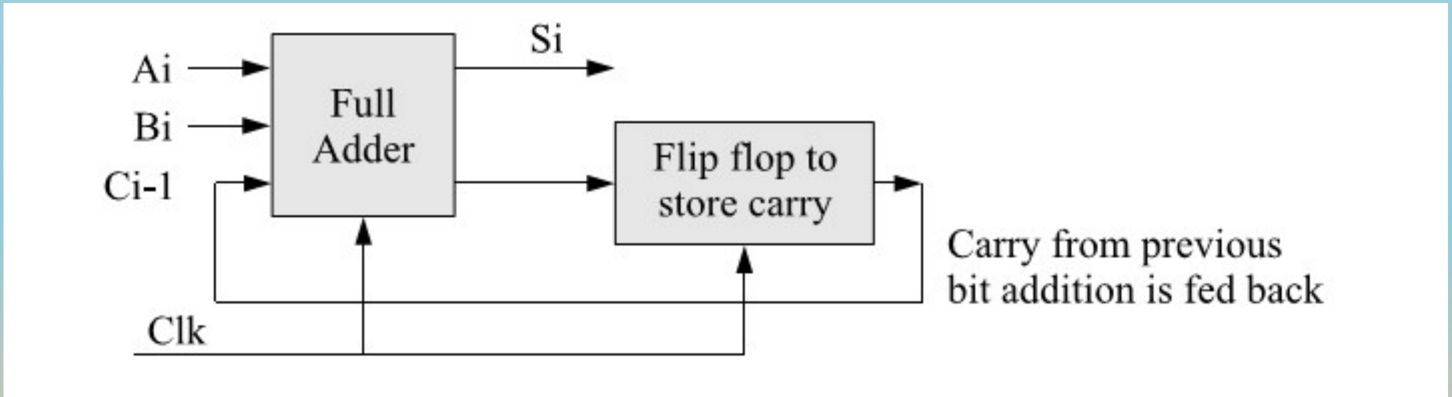
Full Adder: Block

Types of Adders

- **Serial Adder**: Performs binary addition bit by bit
- **Parallel Adder**: Adds all the bits of two numbers in a single clock cycle

Serial Adders

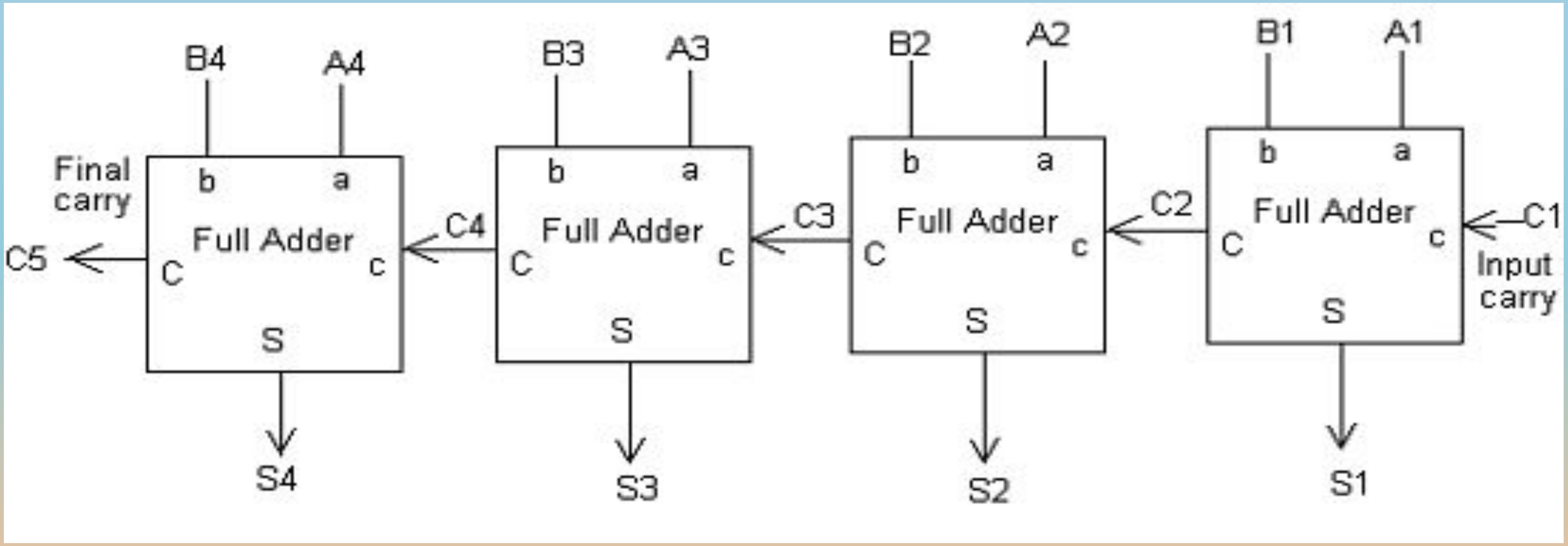
- Serial binary adder is a combinational logic circuit that performs the addition of two binary numbers in **serial** form. **Serial** binary **adder** performs bit by bit addition.



A Serial Adder

Parallel Adders

- A Parallel Adder is a combinational digital circuit that adds two binary numbers in parallel form. It consists of full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder.



Parallel Adders

- Parallel adders use many full adders as the number of bits being added.
- If two registers of n bits each are added, we need to have n full adder units in the circuit so that all the bits are added in parallel at the same time clock cycle.
- In order to add all the bits at the same clock cycle, the main challenge is to decide the carry from the previous bits addition without actually adding them. Hence, there is a need of carry generation circuits.

Carry Propagate Adder (CPA)/ Ripple Carry Adder (RCA)

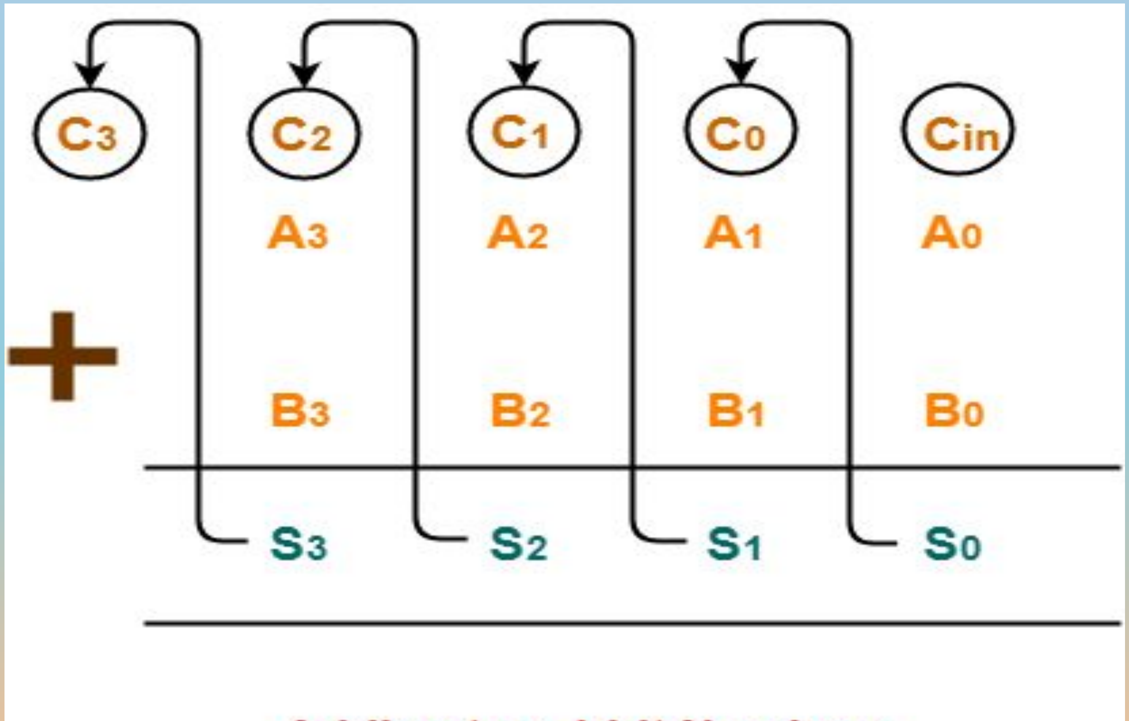
Carry Look Ahead Adder (CLA)

Carry Propagate Adder (CPA)/ Ripple Carry Adder (RCA)

- Each full adder has to wait for its carry-in from its previous stage full adder.
- Thus, nth full adder has to wait until all (n-1) full adders have completed their operations.
- This causes a delay and makes ripple carry adder extremely slow.
- The situation becomes worst when the value of n becomes very large.
- To overcome this disadvantage, Carry Look Ahead Adder comes into play.

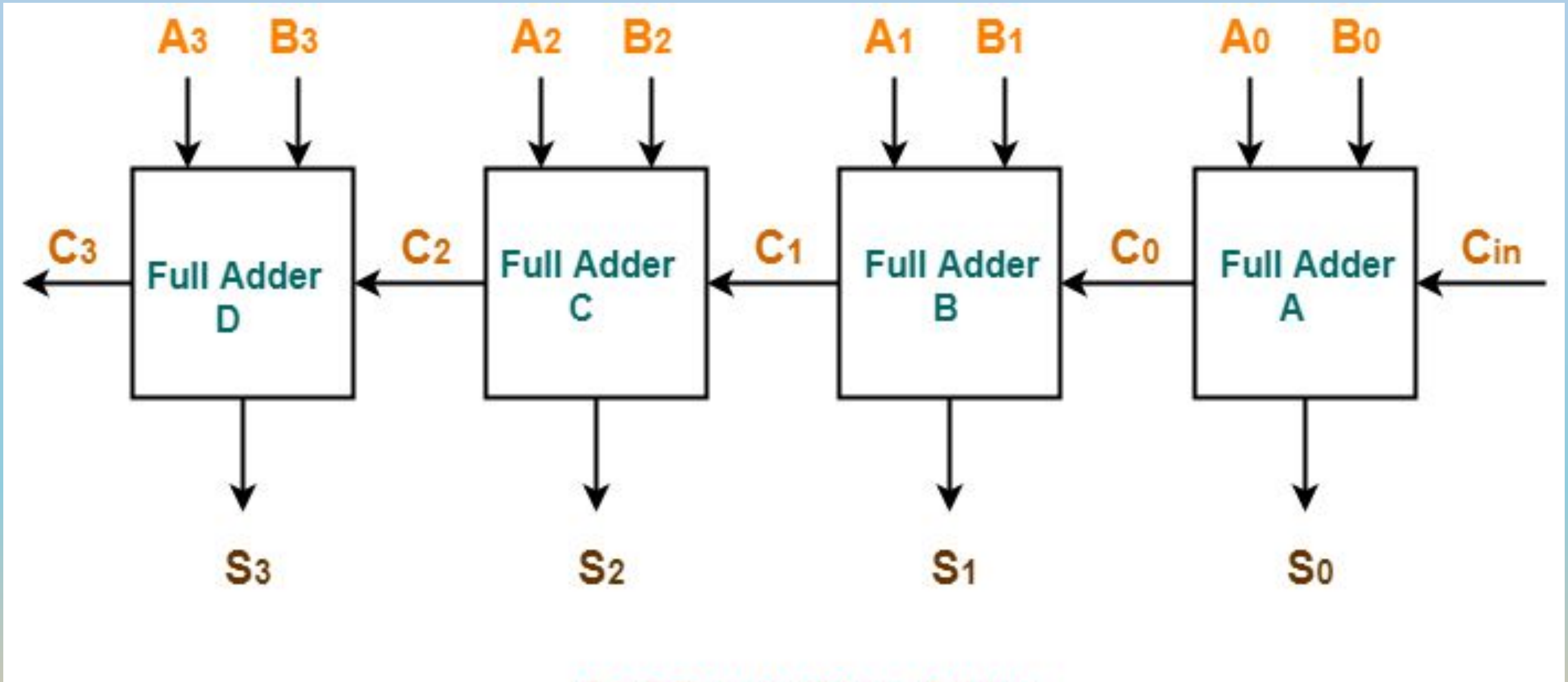
4-bit Ripple Carry Adder-

- In Mathematics, any two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are added as shown below-



Adding two 4-bit Numbers

- Using ripple carry adder, this addition is carried out as shown by the following logic diagram-



4-bit Ripple Carry Adder

- Ripple Carry Adder works in different stages.
- Each full adder takes the carry-in as input and produces carry-out and sum bit as output.
- The carry-out produced by a full adder serves as carry-in for its adjacent most significant full adder.
- When carry-in becomes available to the full adder, it activates the full adder.
- After full adder becomes activated, it comes into operation.

- **<u>Working Of 4-bit Ripple Carry Adder-</u>**
- The two 4-bit numbers are 0101 ($A_3A_2A_1A_0$) and 1010 ($B_3B_2B_1B_0$).
- These numbers are to be added using a 4-bit ripple carry adder.
- 4-bit Ripple Carry Adder carries out the addition as explained in the following stages-
- **<u>Stage-01:</u>**
- When C_{in} is fed as input to the full Adder A, it activates the full adder A.
- Then at full adder A, $A_0 = 1$, $B_0 = 0$, $C_{in} = 0$.
- Full adder A computes the sum bit and carry bit as-
- **<u>Calculation of S_0–</u>**
- $S_0 = A_0 \oplus B_0 \oplus C_{in}$
- $S_0 = 1 \oplus 0 \oplus 0$
- $S_0 = 1$
- **<u>Calculation of C_0–</u>**
-
- $C_0 = A_0B_0 \oplus B_0C_{in} \oplus C_{in}A_0$
- $C_0 = 1.0 \oplus 0.0 \oplus 0.1$
- $C_0 = 0 \oplus 0 \oplus 0$
- $C_0 = 0$

- **Stage-02:**
- When C_0 is fed as input to the full adder B, it activates the full adder B.
- Then at full adder B, $A_1 = 0$, $B_1 = 1$, $C_0 = 0$.
- Full adder B computes the sum bit and carry bit as-
- **Calculation of S_1–**
- $S_1 = A_1 \oplus B_1 \oplus C_0$
- $S_1 = 0 \oplus 1 \oplus 0$
- $S_1 = 1$
- **Calculation of C_1–**
- $C_1 = A_1B_1 \oplus B_1C_0 \oplus C_0A_1$
- $C_1 = 0.1 \oplus 1.0 \oplus 0.0$
- $C_1 = 0 \oplus 0 \oplus 0$
- $C_1 = 0$

- **Stage-03:**
- When C_1 is fed as input to the full adder C, it activates the full adder C.
- Then at full adder C, $A_2 = 1$, $B_2 = 0$, $C_1 = 0$.
- Full adder C computes the sum bit and carry bit as-
- **Calculation of S_2–**
-
- $S_2 = A_2 \oplus B_2 \oplus C_1$
- $S_2 = 1 \oplus 0 \oplus 0$
- $S_2 = 1$
-
- **Calculation of C_2–**
-
- $C_2 = A_2B_2 \oplus B_2C_1 \oplus C_1A_2$
- $C_2 = 1.0 \oplus 0.0 \oplus 0.1$
- $C_2 = 0 \oplus 0 \oplus 0$
- $C_2 = 0$

- **Stage-04:**
- When C_2 is fed as input to the full adder D, it activates the full adder D.
- Then at full adder D, $A_3 = 0$, $B_3 = 1$, $C_2 = 0$.
- Full adder D computes the sum bit and carry bit as-
- **Calculation of S_3–**
- $S_3 = A_3 \oplus B_3 \oplus C_2$
- $S_3 = 0 \oplus 1 \oplus 0$
- $S_3 = 1$
- **Calculation of C_3–**
- $C_3 = A_3B_3 \oplus B_3C_2 \oplus C_2A_3$
- $C_3 = 0.1 \oplus 1.0 \oplus 0.0$
- $C_3 = 0 \oplus 0 \oplus 0$
- $C_3 = 0$

- Thus finally,
- Output Sum = $S_3S_2S_1S_0$ = 1111
- Output Carry = $C_3 = 0$

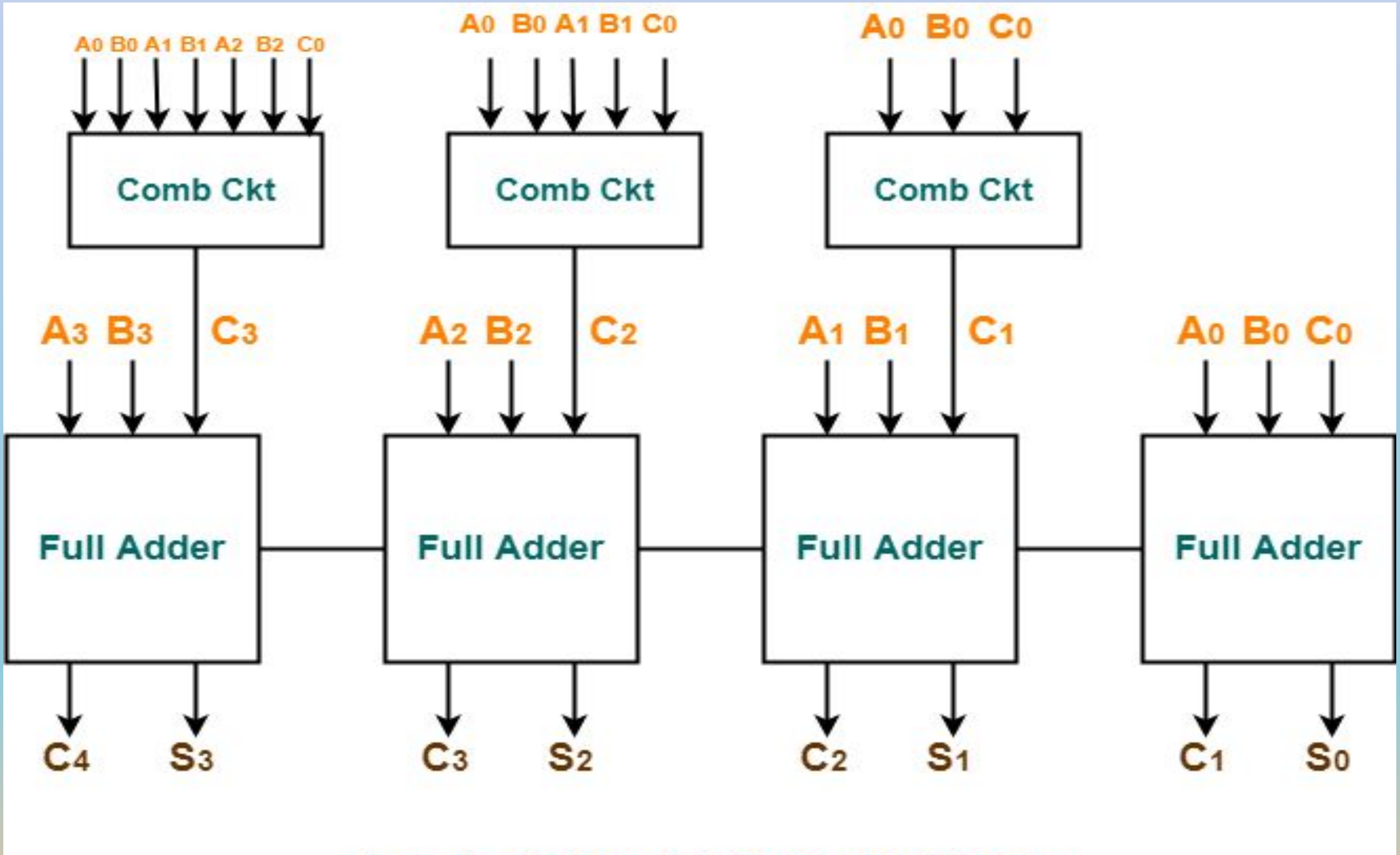
Disadvantages of Ripple Carry Adder-

- Ripple Carry Adder does not allow to use all the full adders simultaneously.
- Each full adder has to necessarily wait until the carry bit becomes available from its adjacent full adder.
- This increases the propagation time.
- Due to this reason, ripple carry adder becomes extremely slow.
- This is considered to be the biggest disadvantage of using ripple carry adder.
- To overcome this disadvantage, **Carry Look Ahead Adder** comes

Carry Look Ahead Adder-

- Carry Look Ahead Adder is an improved version of the ripple carry adder.
- It generates the carry-in of each full adder simultaneously without causing any delay.

Logic Diagram



Carry Look Ahead Adder Logic Diagram

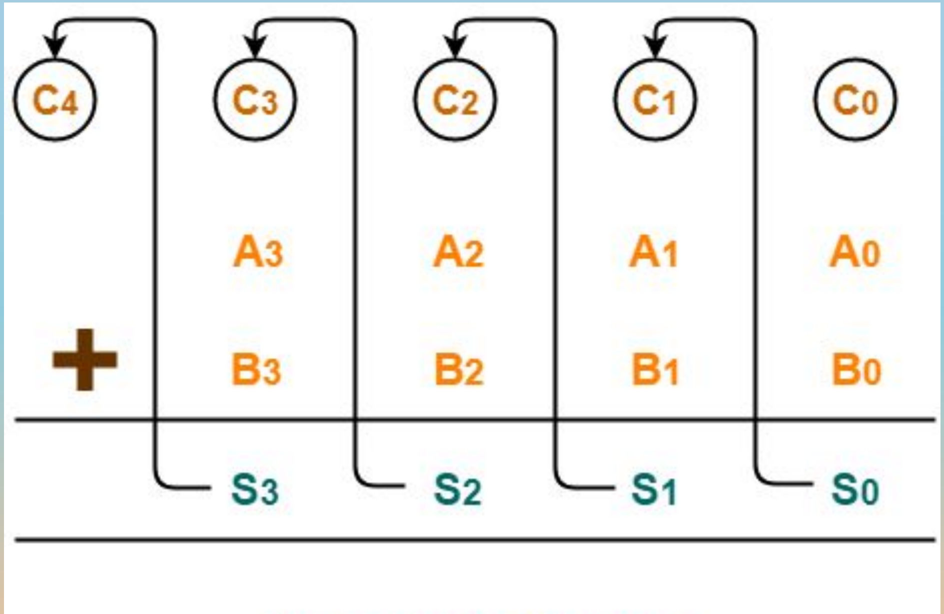
- The carry-in of any stage full adder depends only on the following two parameters-
- Bits being added in the previous stages
- Carry-in provided in the beginning

Now,

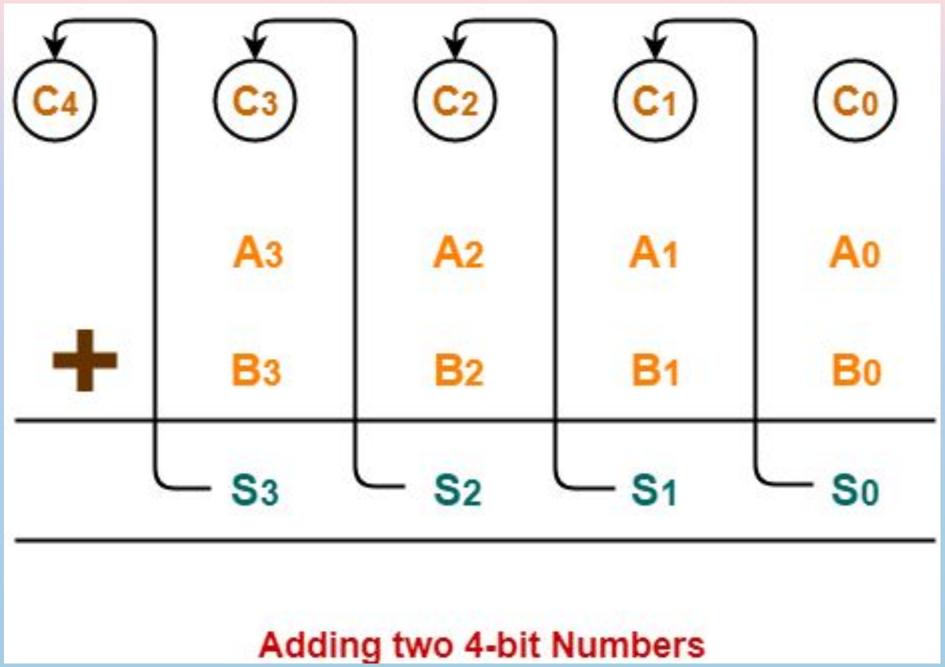
- The above two parameters are always known from the beginning.
- So, the carry-in of any stage full adder can be evaluated at any instant of time.
- Thus, any full adder need not wait until its carry-in is generated by its previous stage full adder.

4-Bit Carry Look Ahead Adder-

- Consider two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are to be added.
- Mathematically, the two numbers will be added as-



Adding two 4-bit Numbers



From here, we have-

- $C_1 = C_0\,(A_0 \oplus B_0) + A_0B_0$
- $C_2 = C_1\,(A_1 \oplus B_1) + A_1B_1$
- $C_3 = C_2\,(A_2 \oplus B_2) + A_2B_2$
- $C_4 = C_3\,(A_3 \oplus B_3) + A_3B_3$

For simplicity, Let-

$G_i = A_iB_i$ where G is called carry generator

$P_i = A_i \oplus B_i$ where P is called carry propagator

Then, re-writing the above equations, we have-

$C_1 = C_0P_0 + G_0$ ………….. (1)

$C_2 = C_1P_1 + G_1$ ………….. (2)

$C_3 = C_2P_2 + G_2$ ………….. (3)

$C_4 = C_3P_3 + G_3$ ………….. (4)

Now,

Clearly, C1, C2 and C3 are intermediate carry bits.

So, let's remove C_1, C_2 and C_3 from RHS of every equation.

Substituting (1) in (2), we get C_2 in terms of C_0.

Then, substituting (2) in (3), we get C_3 in terms of C_0 and so on.

Finally, we have the following equations-

- $C_1 = C_0P_0 + G_0$
- $C_2 = C_0P_0P_1 + G_0P_1 + G_1$
- $C_3 = C_0P_0P_1P_2 + G_0P_1P_2 + G_1P_2 + G_2$
- $C_4 = C_0P_0P_1P_2P3 + G_0P_1P_2P_3 + G_1P_2P_3 + G_2P_3 + G_3$

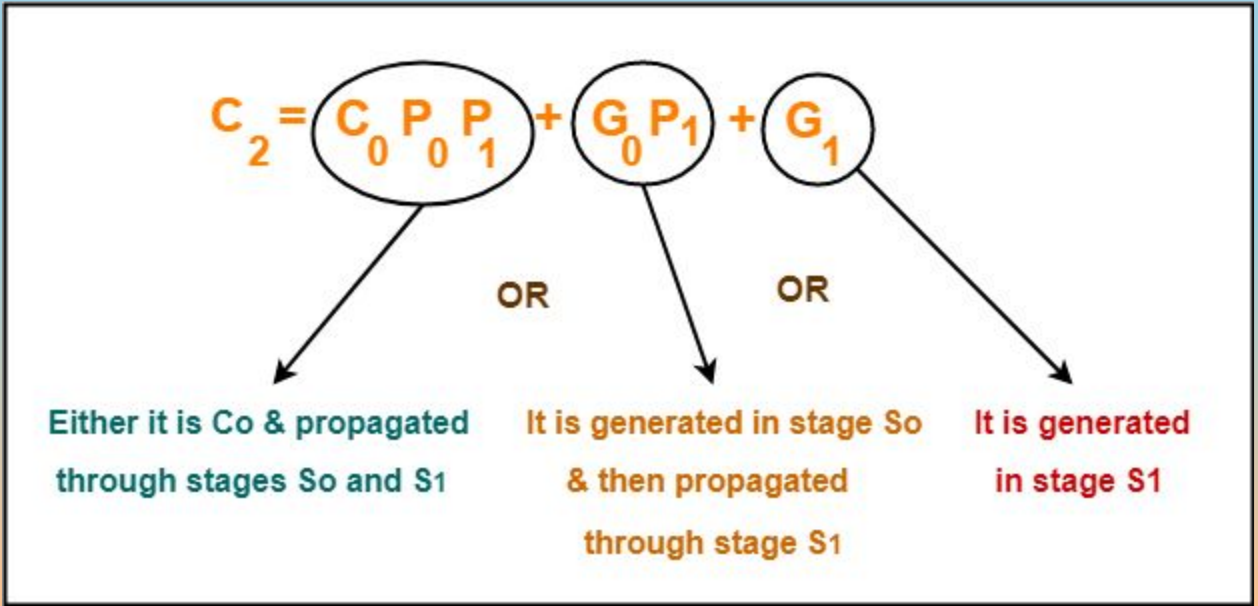
These equations are important to remember.

These equations show that the carry-in of any stage full adder depends only on-

Bits being added in the previous stages

Carry bit which was provided in the beginning

- As an example, let us consider the equation for generating carry bit C_2.
- There are three possible reasons for generation of C_2 as depicted in the following picture-


$C_2 = C_0 P_0 P_1 + G_0 P_1 + G_1$
OR
OR
Either it is Co & propagated through stages So and S1
It is generated in stage So & then propagated through stage S1
It is generated in stage S1

Implementation Of Carry Generator Circuits-

The above carry generator circuits are usually implemented as-

Two level combinational circuits.

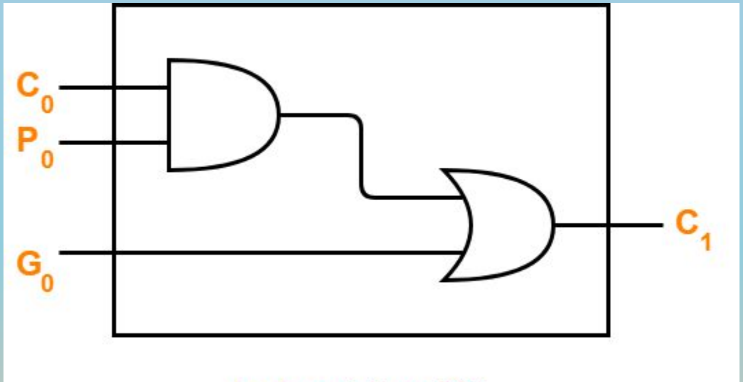
Using AND and OR gates where gates are assumed to have any number of inputs.

Implementation Of C_1–

The carry generator circuit for C1 is implemented as shown below.

It requires 1 AND gate and 1 OR gate.

$C_1 = C_0P_0 + G_0$



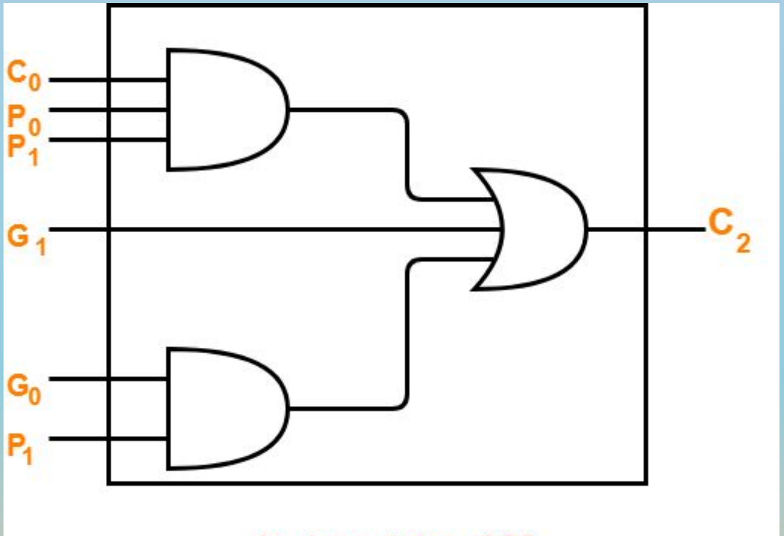
Implementation of C1

• **Implementation Of C_2 –**

The carry generator circuit for C2 is implemented as shown below.

It requires 2 AND gates and 1 OR gate.

$C_2 = C_0P_0P_1 + G_0P_1 + G_1$



Implementation of C2

- **Implementation Of C_3 & C_4–**
- Similarly, we implement C_3 and C_4.
- Implementation of C_3 uses 3 AND gates and 1 OR gate.
- Implementation of C_4 uses 4 AND gates and 1 OR gate.
-
- Total number of gates required to implement carry generators (provided carry propagators P_i and carry generators G_i) are-
- Total number of AND gates required for addition of 4-bit numbers = 1 + 2 + 3 + 4 = 10.
- Total number of OR gates required for addition of 4-bit numbers = 1 + 1 + 1 + 1 = 4.

- For a n-bit carry look ahead adder to evaluate all the carry bits, it requires-
- Number of AND gates = n(n+1) / 2
- Number of OR gates = n

Advantages of Carry Look Ahead Adder

- The advantages of carry look ahead adder are-
- It generates the carry-in for each full adder simultaneously.
- It reduces the propagation delay.

Carry Look Ahead Adder (CLA)

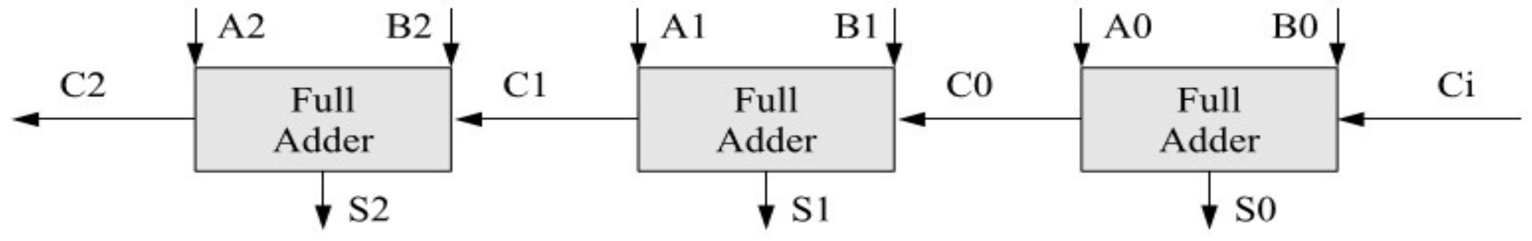
Carry Look Ahead Adder (CLA)

- Most other arithmetic operations, e.g. multiplication and division are implemented using several add/subtract steps. Thus, improving the speed of addition will improve the speed of all other arithmetic. operations.
- Accordingly, reducing the carry propagation delay of adders is of great importance. Different logic design approaches have been employed to overcome the carry propagation problem.
- One widely used approach employs the principle of carry look-ahead solves this problem by calculating the carry signals in advance, based on the input signals.

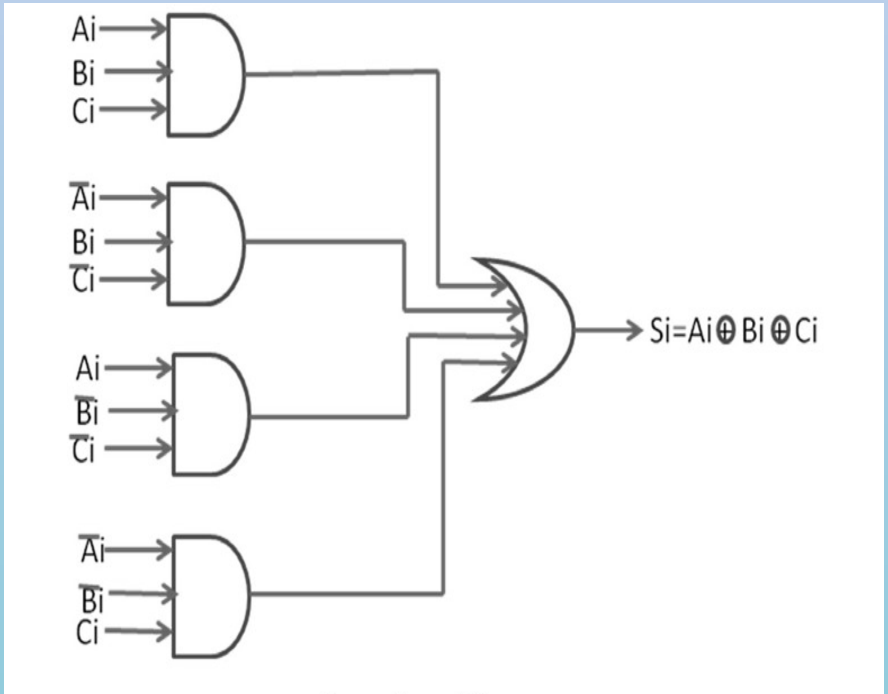
CLA logic uses the concepts of **generating** and **propagating** carries.

For each bit in a binary sequence to be added, the CLA Logic determines whether that bit pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry, ahead of time than when the actual addition is performed.

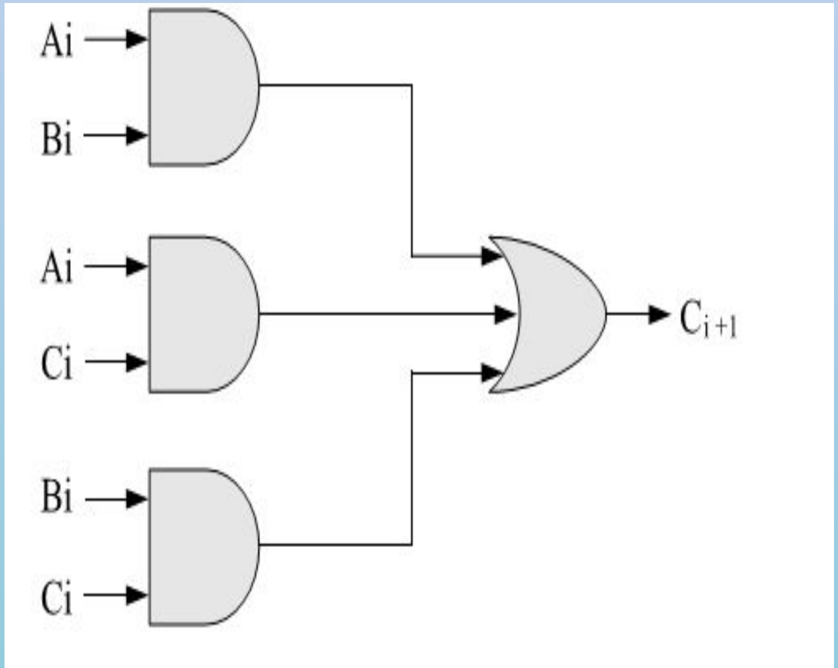
A Carry Look Ahead adder (CLA) improves the speed of calculation by reducing the amount of time required for determining carry bits. So, often it is called a Fast Adder used in digital logic.



Cascaded full adders



Generation of Sum



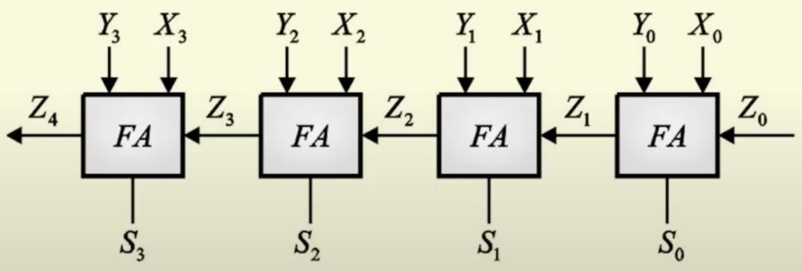
Generation of Carry

Delay Encountered……….

- From the diagram it is clear that for every Ci, we have 2 gate delays (2 layers of gates in carry circuit).
- For a 16-bit register, 15 bits carry propagation takes place. Hence, time delay for the carry propagation is 2 × gate delay × 15 time units, to get C15 for last bit addition.
- From the circuit we can also observe that the final sum can be obtained only by adding last group of bits (A15+ B15+C15), after the 15 time units delay.
- To add the last group, adder delay will be 3 gate delays (three stages in sum circuit). So, total delay in this will be 3 × gate delay.
- Assuming 10 nano seconds as gate delay (TTL Logic) we require total delay = 2 × 15 × 10 + 3 × 10 = 300 + 30 = 330 ns. Unacceptable in high speed adder.

Question

Figure-I shows a 4-bit ripple carry adder realized using full adders and figure-II shows the circuit of a full adder (FA). The propagation delay of the XOR, AND and OR gates in figure II are 20 ns, 15 ns and 10 ns, respectively. Assume all the inputs to the 4-bit adder are initially reset to 0.



Y_3 X_3 Y_2 X_2 Y_1 X_1 Y_0 X_0
Z_4 Z_3 Z_2 Z_1 Z_0
FA FA FA FA
S_3 S_2 S_1 S_0

Fig. I

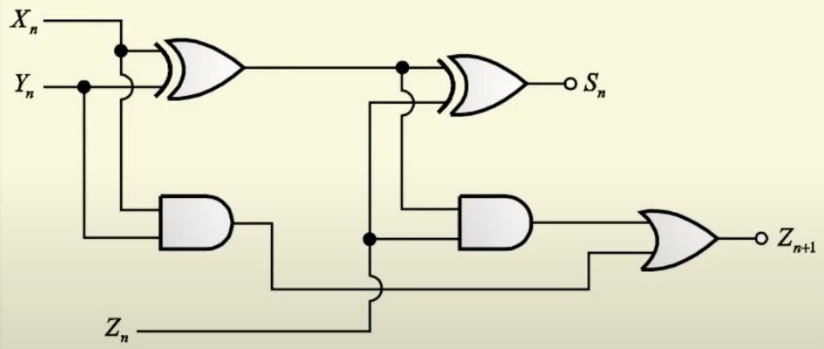


Fig. II

At $t = 0$, the input to the 4-bit adder are changed to $X_3X_2X_1X_0 = 1100, Y_3Y_2Y_1Y_0 = 0100$ and $Z_0 = 1$. The output of the ripple carry adder will be stable at t (in ns) = ________. **[Set - 02]**

Remedy….

- To recover from this unwanted delay, CLA circuits came into being as one of the possible alternatives. In this method, special hardware is used to generate carry Ci (where i > 0).

The principle of generation of carry:

Ci + 1 = Ai Bi + Ai Ci +Bi Ci

= Ai Bi + (Ai + Bi) Ci

= Gi + Pi Ci

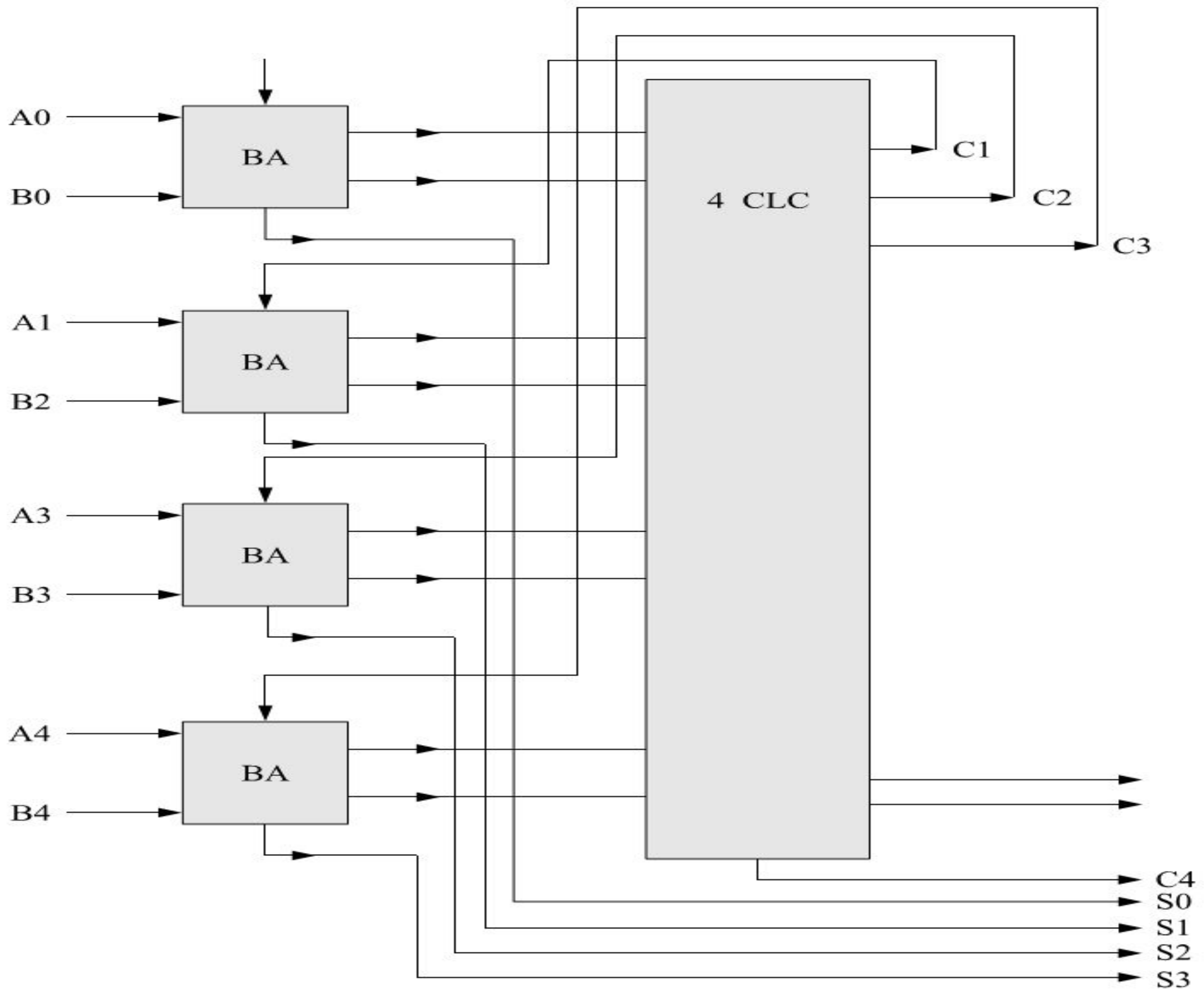
(substituting Gi for Ai Bi and Pi for (Ai + Bi)

If both bits of the current addition are 1's, a carry is generated which is represented by Gi. If Ai or Bi = 1 then the current addition does not generate a carry. However, the previous position's carry Ci bit is propagated to the next bit addition. So the function Gi may be interpreted as Carry Generate Function and the function Pi may be interpreted as Carry Propagate Function.

The carry bits of individual bit position may be expressed as, (considering 4 bits being added)

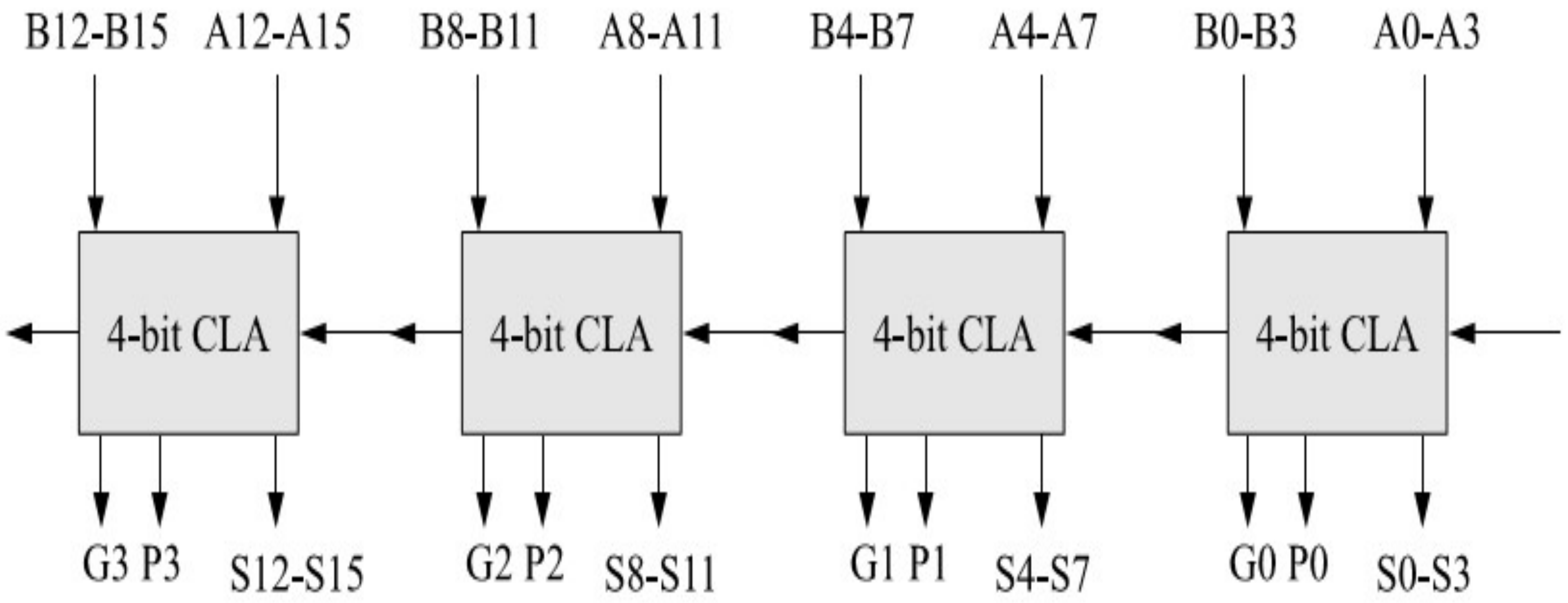
- $C1 = G0 + P0C0$
- $C2 = G1 + P1C1 = G1 + P1(G0 + P0C0) = G1 + P1G0 + P1P0C0$
- $C3 = G2 + P2C2 = G2 + P2(G1 + P1G0 + P1P0C0)$
- $= G2 + P2G1 + P1P2Go + P2P1P0C0$
- $C4 = G3 + P3\ C3 = G3 + P3(G2 + P2G1 + P1P2G0 + P2P1P0C0)$
- $= G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C0$

This clearly indicates that C1, C2, C3, C4, etc., can be generated directly from C0.



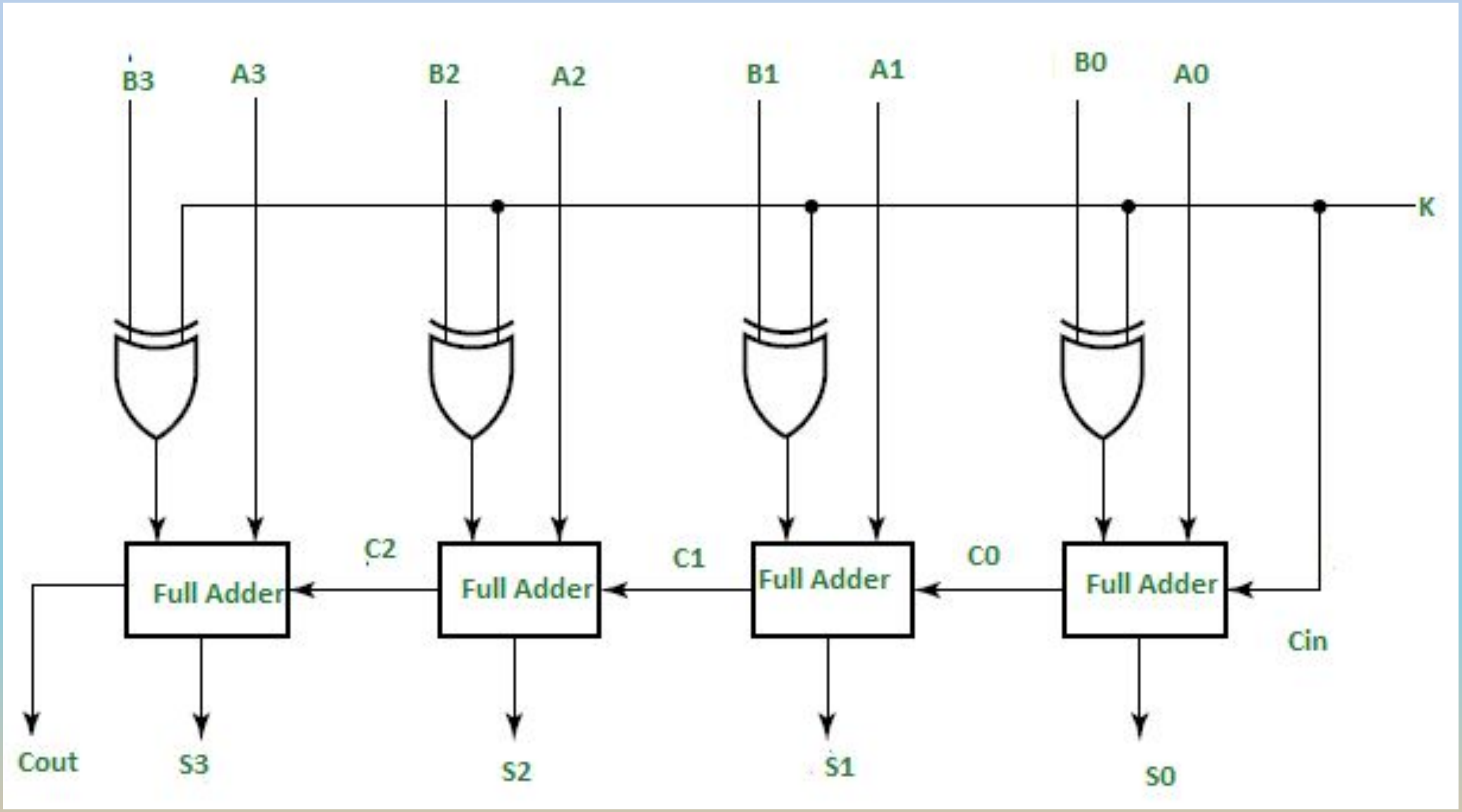
4 bit CLA

One can make use of 4-bit carry look ahead adders in cascade for making bigger sized adder
For example, a 16-bit adder can be built using 4-bit CLA as follows:



16 bit adder from 4 bit CLA

4-bit binary Adder-Subtractor



- A **Binary Adder-Subtractor** is capable of both the addition and subtraction of binary numbers in one circuit itself. The operation is performed depending on the binary value the control signal holds. It is one of the components of the ALU (Arithmetic Logic Unit).
- If the value of K (Control line) is 1, the output of B0(exor)K=B0’(Complement B0). Thus the operation would be A+(B0’). Now 2’s complement subtraction for two numbers A and B is given by A+B’+Cin. This suggests that when K=1, the operation being performed on the four-bit numbers is subtraction.