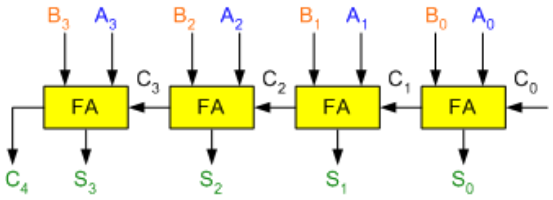
Carry Look Ahead Adders

Carry Look Ahead Adder:

In ***ripple carry adders***, the carry propagation time is the major speed limiting factor as seen in the previous lesson.



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.

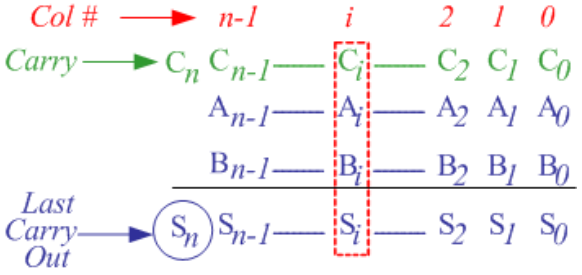
Accordingly, reducing the carry propagation delay of adders is of great importance. Different logic design approaches have been employed to overcome the carry propagation problem.

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.

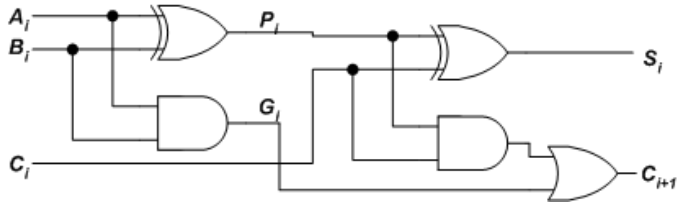
This type of adder circuit is called as ***carry look-ahead adder (CLA adder).*** It is based on the fact that a carry signal will be generated in two cases:

(1) when both bits A_i and B_i are 1, or
(2) when one of the two bits is 1 and the carry-in (carry of the previous stage) is 1.

To understand the carry propagation problem, let's consider the case of adding two *n-bit* numbers ***A*** and ***B***.



The Figure shows the full adder circuit used to add the operand bits in the i^{th} column; *namely A_i & B_i and the carry bit coming from the previous column (C_i).*



In this circuit, the 2 internal signals P_i and G_i are given by:

$$P_i = A_i \oplus B_i \text{ .....(1)}$$

$$G_i = A_i B_i \text{ .....(2)}$$

The output sum and carry can be defined as :

$$S_i = P_i \oplus C_i \text{ .....(3)}$$

$$C_{i+1} = G_i + P_i C_i \text{ .....(4)}$$

G_i is known as the ***carry Generate*** signal since a carry (C_{i+1}) is generated whenever G_i =1, regardless of the input carry (C_i).

P_i is known as the ***carry propagate*** signal since whenever P_i =1, the input carry is propagated to the output carry, i.e., C_{i+1}. = C_i (note that whenever P_i =1, G_i =0).

Computing the values of P_i and G_i only depend on the input operand bits (A_i & B_i) as clear from the Figure and equations.

Thus, these signals settle to their ***steady-state value*** after the propagation through their respective gates.

Computed values of ***all*** the P_i's are valid one XOR-gate delay after the operands A and B are made valid.

Computed values of ***all*** the G_i's are valid one AND-gate delay after the operands A and B are made valid.

The Boolean expression of the carry outputs of various stages can be written as follows:

$C_1 = G_0 + P_0C_0$
$C_2 = G_1 + P_1C_1 = G_1 + P_1 (G_0 + P_0C_0)$
$\quad = G_1 + P_1G_0 + P_1P_0C_0$
$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
$C_4 = G_3 + P_3C_3$
$\quad = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

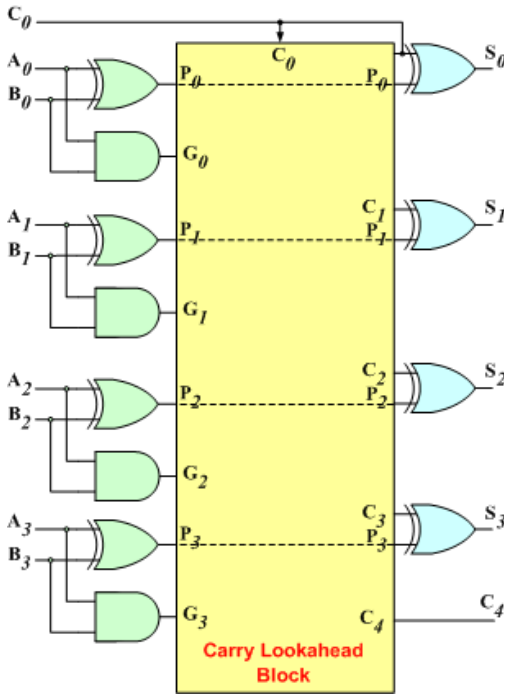
In general, the $i^{th.}$ carry output is expressed in the form $C_i = F_i$ (P's, G's , C_0).

In other words, each carry signal is expressed as a direct SOP function of C_0 rather than its preceding carry signal.

Since the Boolean expression for each output carry is expressed in SOP form, it can be implemented in two-level circuits.

The 2-level implementation of the carry signals has a propagation delay of 2 gates, i.e., 2τ.

The 4-bit carry look-ahead (CLA) adder consists of 3 levels of logic:



First level: Generates all the P & G signals. Four sets of P & G logic (each consists of an XOR gate and an AND gate). Output signals of this level (P's & G's) will be valid after 1τ.

Second level: The Carry Look-Ahead (CLA) logic block which consists of four 2-level implementation logic circuits. It generates the carry signals (C_1, C_2, C_3, and C_4) as defined by the above expressions. Output signals of this level (C_1, C_2, C_3, and C_4) will be valid after 3τ.

Third level: Four XOR gates which generate the sum signals (S_i) ($S_i = P_i \oplus C_i$). Output signals of this level (S_0, S_1, S_2, and S_3) will be valid after 4τ.

Thus, the 4 Sum signals (S_0, S_1, S_2 & S_3) will all be valid after a total delay of 4τ compared to a delay of $(2n+1)\tau$ for Ripple Carry adders.

For a 4-bit adder ($n = 4$), the Ripple Carry adder delay is 9τ.

The disadvantage of the CLA adders is that the carry expressions (and hence logic) become quite complex for more than 4 bits.

Thus, CLA adders are usually implemented as 4-bit modules that are used to build larger size adders.