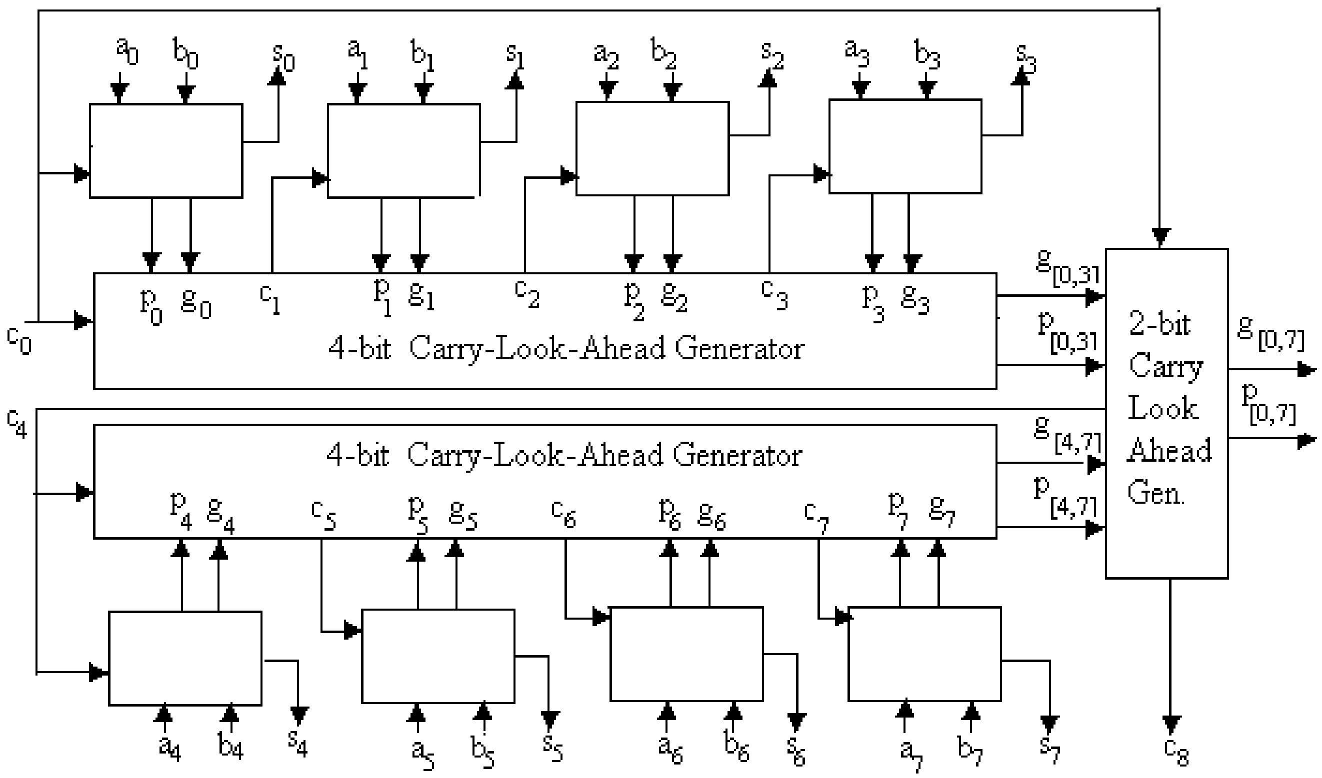
8-bit Carry Look Ahead Adder



It is an improved version of the Ripple Carry Adder, and its main purpose is to remove the time delay caused by the Ripple Carry Adder.

It calculates the carry-in of each full adder **simultaneously** without waiting for the carry-in of previous adder.

<u>Principle:</u>

It works on the principle that the carry-in at any stage does not depend on the carry-in of previous stage but only on the inputs provided by the user i.e., input bits and the first carry-in.

<u>Working:</u>

Consider two 8-bit numbers $A=A_7A_6A_5A_4A_3A_2A_1A_0$ and $B= B_7B_6B_5B_4B_3B_2B_1B_0$

And a carry-in C_0.

Mathematically, the sum is calculated as follows:

$$C_8\ C_7\ C_6\ C_5\ C_4\ C_3\ C_2\ C_1\ C_0$$

$$
\begin{array}{cccccccccc}
 & A_7 & A_6 & A_5 & A_4 & A_3 & A_2 & A_1 & A_0 \\
+ & \underline{B_7} & \underline{B_6} & \underline{B_5} & \underline{B_4} & \underline{B_3} & \underline{B_2} & \underline{B_1} & \underline{B_0} \\
 & \underline{S_7} & \underline{S_6} & \underline{S_5} & \underline{S_4} & \underline{S_3} & \underline{S_2} & \underline{S_1} & \underline{S_0}
\end{array}
$$

Sum S= S7S6S5S4S3S2S1S0 and final carry-out C_8.

By properties of bit-wise addition, we get:

$$C_1 = C_0 (A_0 \oplus B_0) + A_0B_0$$

$$C_2 = C_1 (A_1 \oplus B_1) + A_1B_1$$

$$C_3 = C_2 (A_2 \oplus B_2) + A_2B_2$$

$$C_4 = C_3 (A_3 \oplus B_3) + A_3B_3$$

$$\text{Till } C_8 = C_7(A_7 \oplus B_7) + A_7B_7$$

Where $\oplus$ indicates OR operator and product mean AND.

Now, we assign:

$G_i = A_iB_i$ called carry-generator.

$P_i = A_i \oplus B_i$ called carry-propagator.

So our equations become:

- $C_1 = C_0P_0 + G_0$
- $C_2 = C_0P_0P_1 + G_0P_1 + G_1$ and so on.

One can clearly see that while calculating each carry-in we only require the input bits and C_0.

Implementation

The carry-in circuits are implemented using a circuit of AND, OR gates. You can see that C_1 requires 1 AND, 1 OR gate. Similarly C_2 requires 2 AND, 1 OR gate. Therefore C_n requires n AND gates, 1 OR gate.

Therefore, total number of AND gates required= 1+2+….n=n(n+1)/2.

OR gates required=n.

For 8-bit you need 36 AND gates and 8 OR gates.

NOTE- Although we have done the mathematics considering 8- full adders, we generally divide the process in 2 4-bit adders and then combine the result, the same is shown in the logic diagram.