8-Bit Carry Look Ahead Adder

Carry look ahead adder is an improvised version of ripple carry adder. It generates the carry-in of each full adder simultaneously without causing any delay. A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two level logic.

The time complexity of carry look-ahead adder = O(logn).

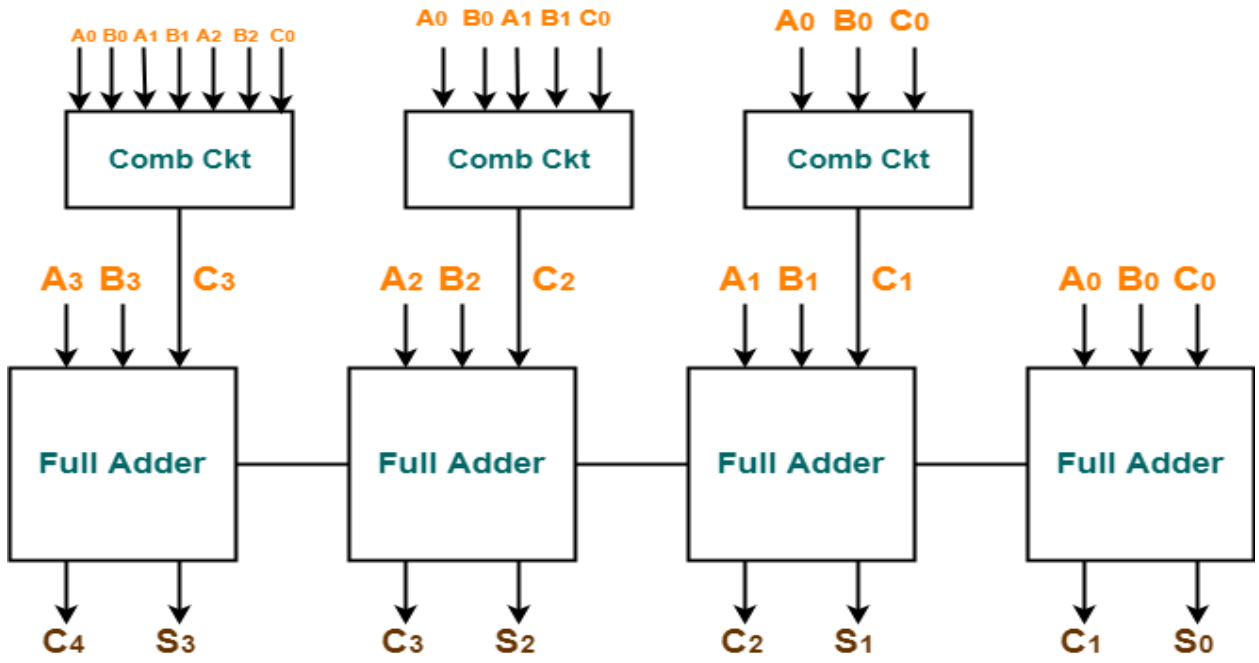
The working of the carry look-ahead adder is based on the principle- The carry-in of any stage full adder is independent of the carry bits generated during intermediate stages. The carry-in of any stage full adder depends only on the following two paramters:

- 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 a 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.

Logic Diagram: (for 4 bit {can be extended to 8 bits })



Carry Look Ahead Adder Logic Diagram

Detailed Working:

Consider the full adder circuit shown above with the corresponding truth table. We define two variables as *carry generate Gi* and *carry propagate Pi.* Then,

$$Pi = Ai \wedge Bi$$

$$Gi = Ai \& Bi$$

The sum output and carry output can be expressed in terms of carry generate Gi and carry propagate Pi as

$$Si = Pi \wedge Ci$$

$$Ci+1 = Gi \mid (Pi \& Ci)$$

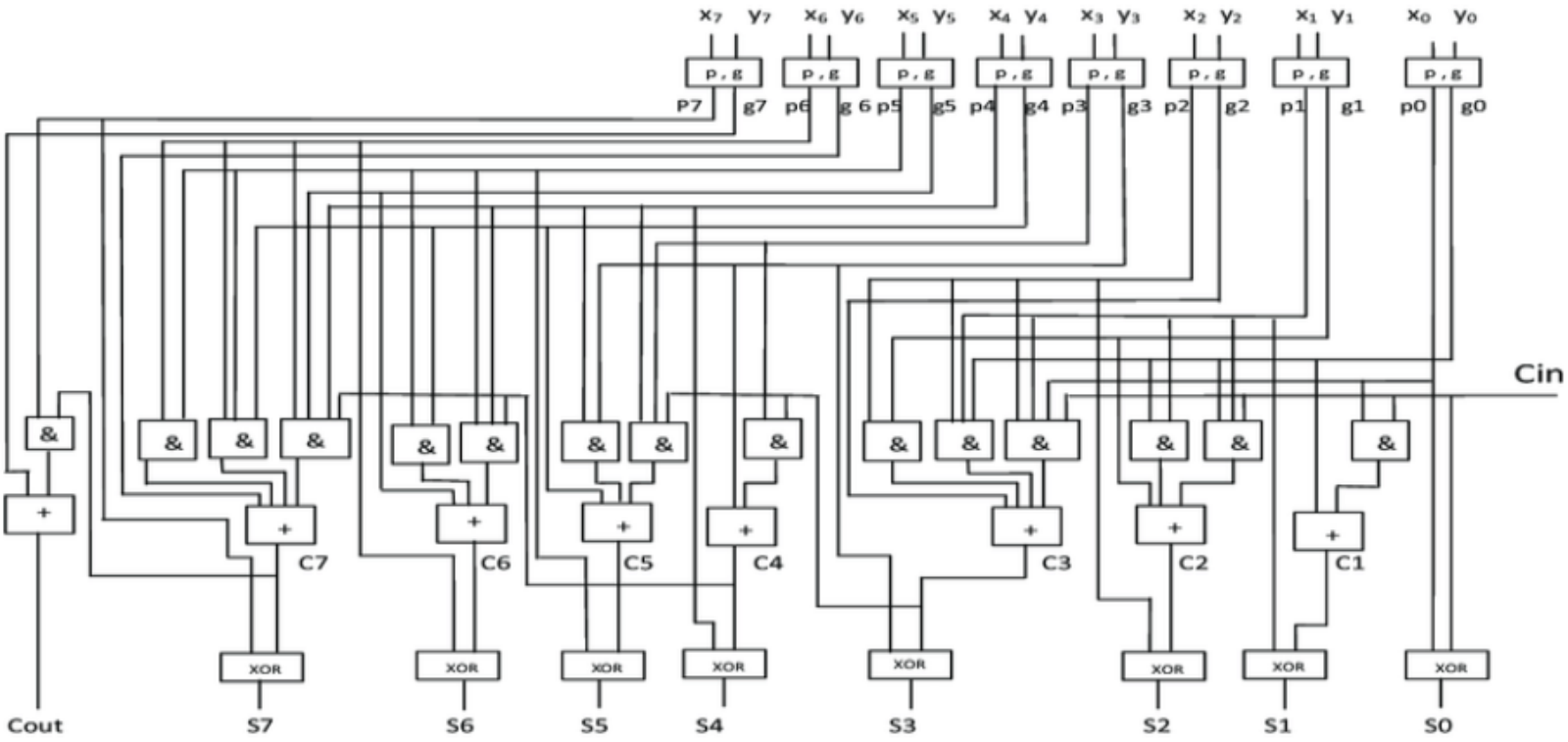
where Gi produces the carry when both Ai and Bi are 1, regardless of the carry input. Pi is associated with the propagation of carry from Ci to Ci+1.

The carry output Boolean function of each stage in a 8 stage carry look-ahead carry adder can be expressed as

$C_1 = G_0 \mid (P_0 \& C_{in})$
$C_2 = G_1 \mid (P_1 \& C_1) = G_1 \mid (P_1 \& G_0) \mid (P_1 \& P_0 \& C_{in})$
$C_3 = G_2 \mid (P_2 \& C_2) = G_2 \mid (P_2 \& G_1) \mid (P_2 \& P_1 \& G_0) \mid (P_2 \& P_1 \& P_0 \& C_{in})$
$C_4 = , G_3 \mid (P_3 \& 3) = G_3 \mid (P_3 \& G_2) \mid (P_3 \& P_2 \& G_1) \mid (P_3 \& P_2 \& P_1 \& G_0) \mid (P_3 \& P_2 \& P_1 \& P_0 \& C_{in})$
.....and so on all other carry are calculated.

From the above Boolean equations, we can observe that C_{i+1} does not have to wait for C_i or C_{i-1} to propagate but C_{i+1} is propagated at the same time as C_i, C_{i-1}....and so. Since the boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by OR gate.

Gate Level Implementation of Look Ahead Carry Adder :



Time Complexity Analysis:

We could think carry look-ahead adder made up of two parts:

1. The part that computes carry for each bit.
2. The part that adds the input bits and carry for each bit position.

The *log(n)* complexity arises from the part that generates the carry, not the part that adds the bits.
Now for the generation of nth carry bit we need to perform a AND between n+1 inputs.

The complexity of the adder comes down to how we perform AND operation. If we have AND gates, each with fan-in of k. Then we can find the AND of n+1 bits in logk(n+1) time.

Advantages of Carry Look-Ahead Adder:

- It generates the carry-in for each full adder simultaneously.
- It reduces the propagation delay.

Disadvantages of Carry Look-Ahead Adder:

- It involves complex hardware.
- It is costlier since it involves complex hardware.
- It gets more complicated as the number of bits increases.