Combinational Logic Circuits

Lecture-17

By
Bhagyalaxmi Behera
Asst. Professor (Dept. of ECE)

Combinational Logic

Overview of previous lecture

- What is a Combinational Circuit.
- How is it different from Sequential Circuit.
- What is the analysis procedure for a Combinational Circuit.
- What is the design procedure for a Combinational Circuit.

Binary Adder

- The simple binary addition consists of four possible elementary operations:

  0+ 0 = 0

  0 + 1 = 1

  1 + 0 = 1

  1 + 1 = 10 (The higher significant bit of this result is called a *carry*) .

When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits.

- A combinational circuit that performs the addition of two bits is called a *half adder* .

- One that performs the addition of three bits (two significant bits and a previous carry) is a *full adder* .

- A *binary adder–subtractor* is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers.

➢ Half Adder

From the verbal explanation of a half adder, we find that this circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. We assign symbols x and y to the two inputs and S (for sum) and C (for carry) to the outputs.

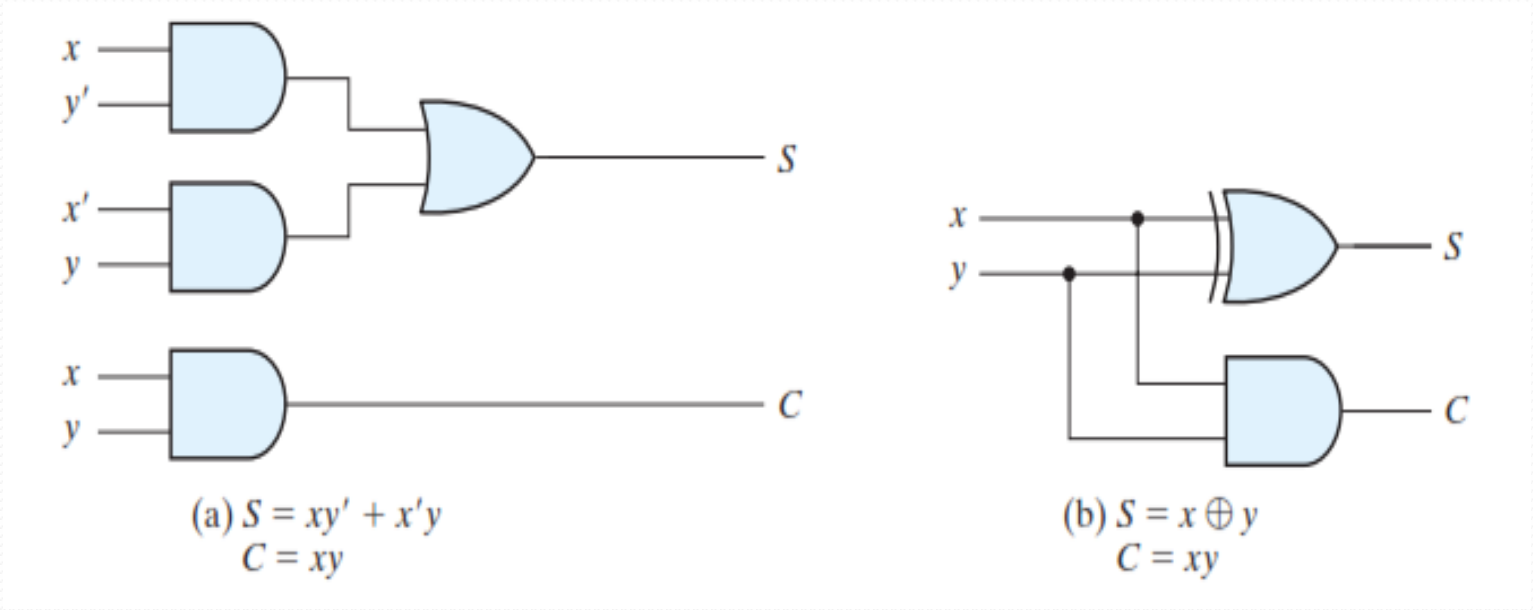
- The truth table for the half adder is listed below.
- The C output is 1 only when both inputs are 1.
- The S output represents the least significant bit of the sum.

Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum-of-products expressions are

$$S = x'y + xy'$$
$$C = xy$$

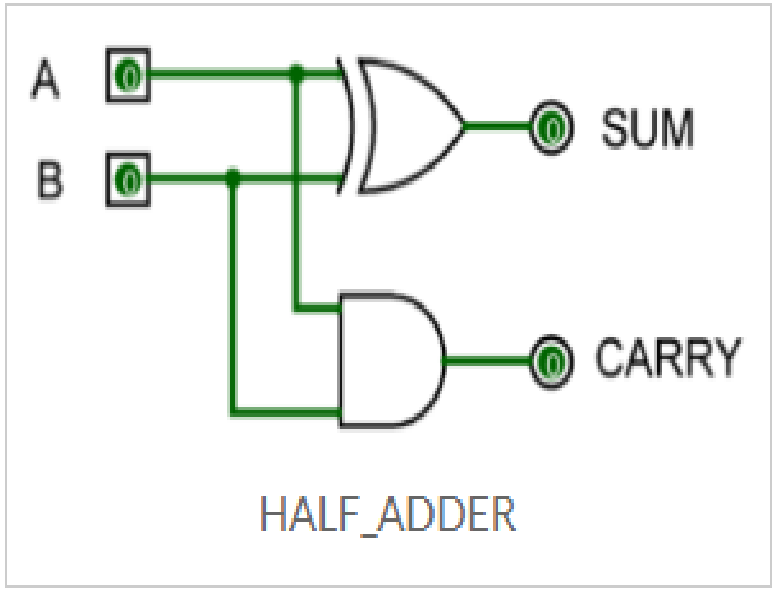


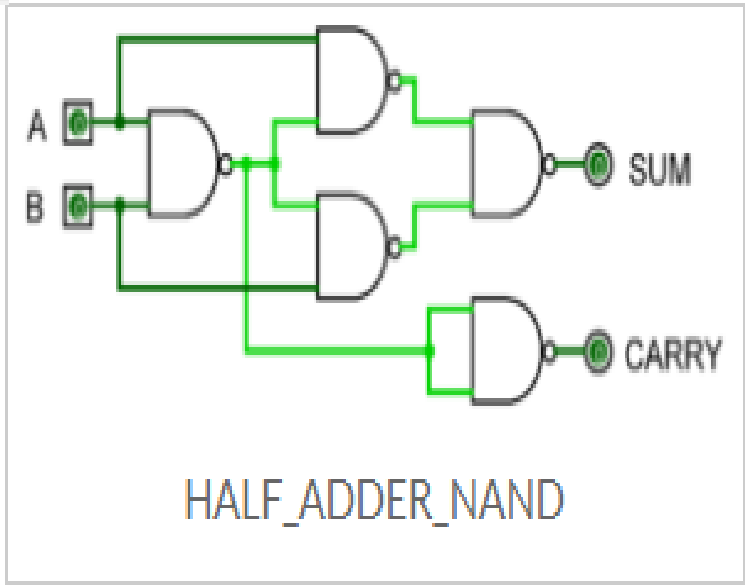
Implementation of half adder

Implement HALF ADDER using NAND GATES

$$S = A \oplus B$$

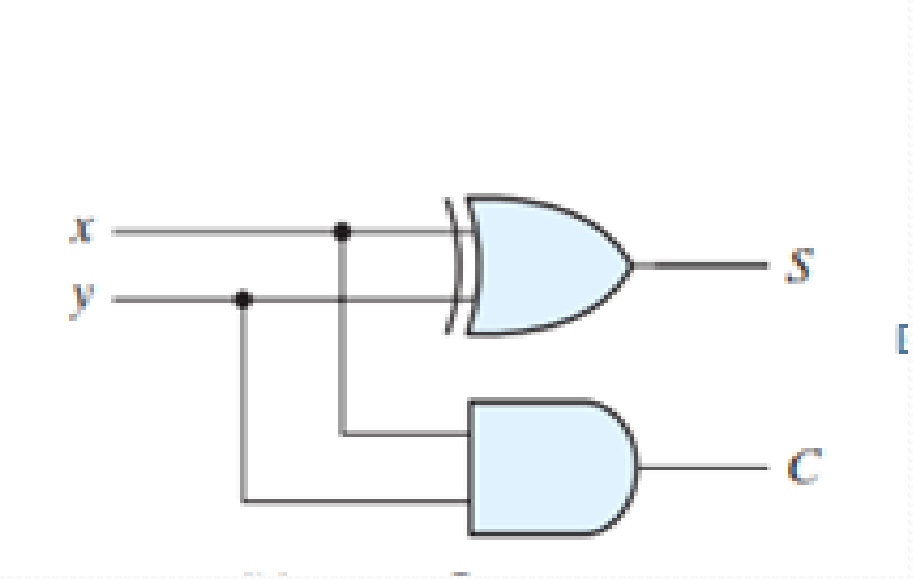
$$C = AB$$





- HDL for Half Adder

```
module  half_adder ( output  S, C,  input  x, y);
xor  (S, x, y);
and  (C, x, y);
endmodule
```



Full Adder

- A full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input, z, represents the carry from the previous lower significant position.

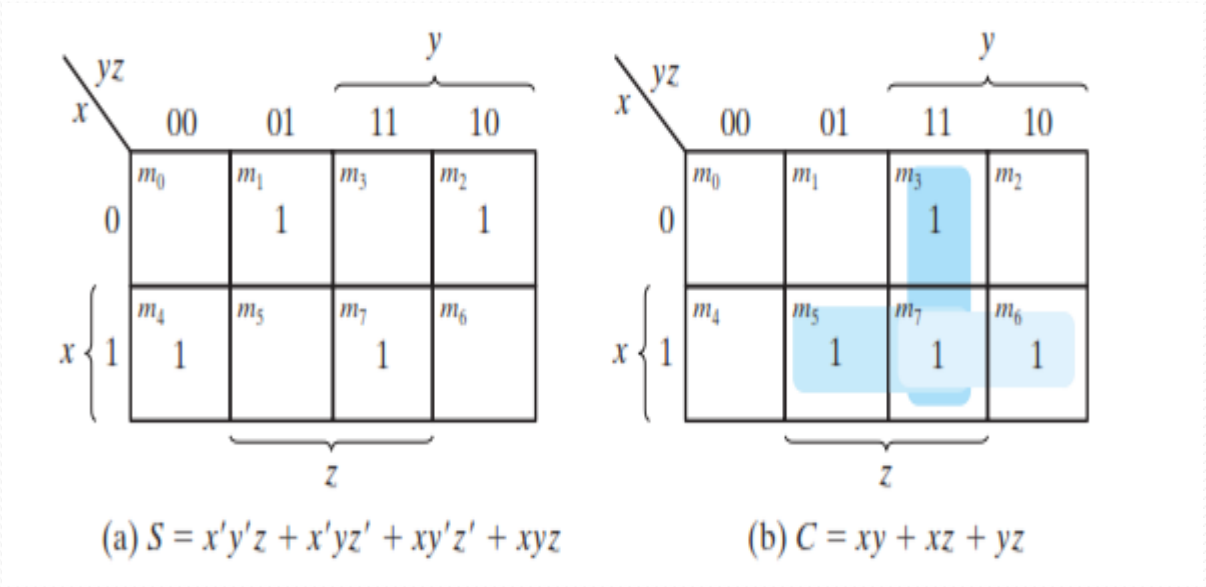
- The two outputs are designated by the symbols S for sum and C for carry. The binary variable S gives the value of the least significant bit of the sum. The binary variable C gives the output carry formed by adding the input carry and the bits of the words.

The truth table of the full adder is listed below.

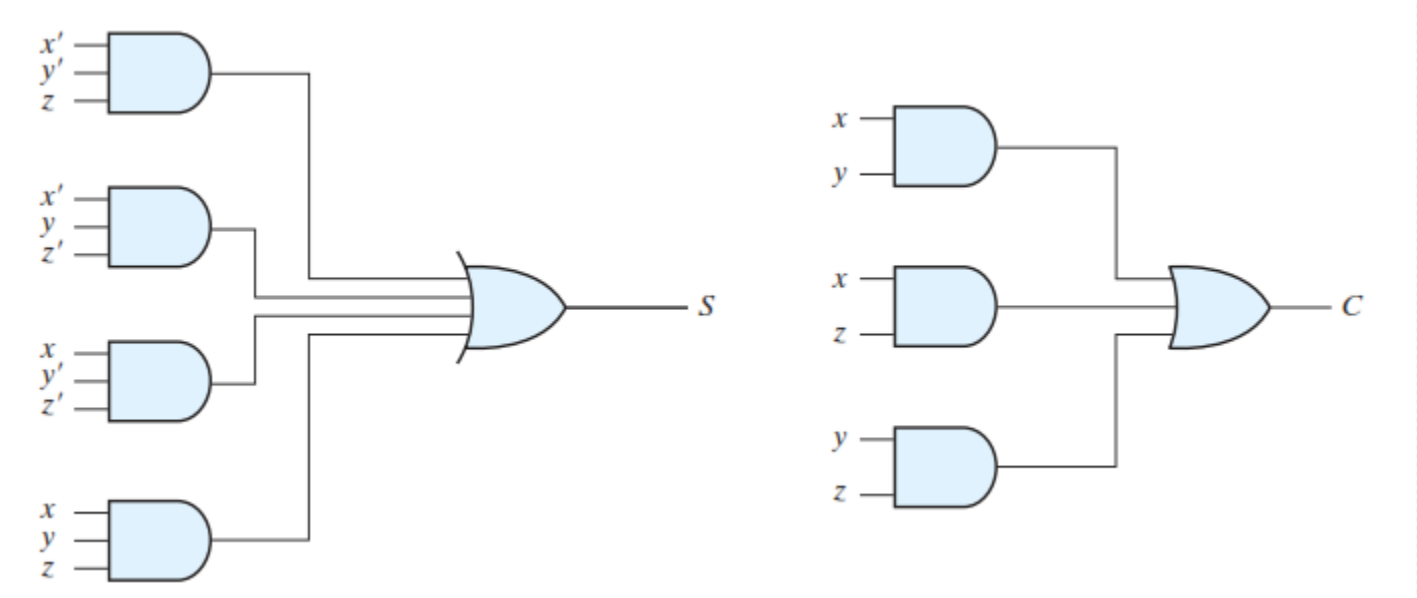
Full Adder

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-Maps for full adder

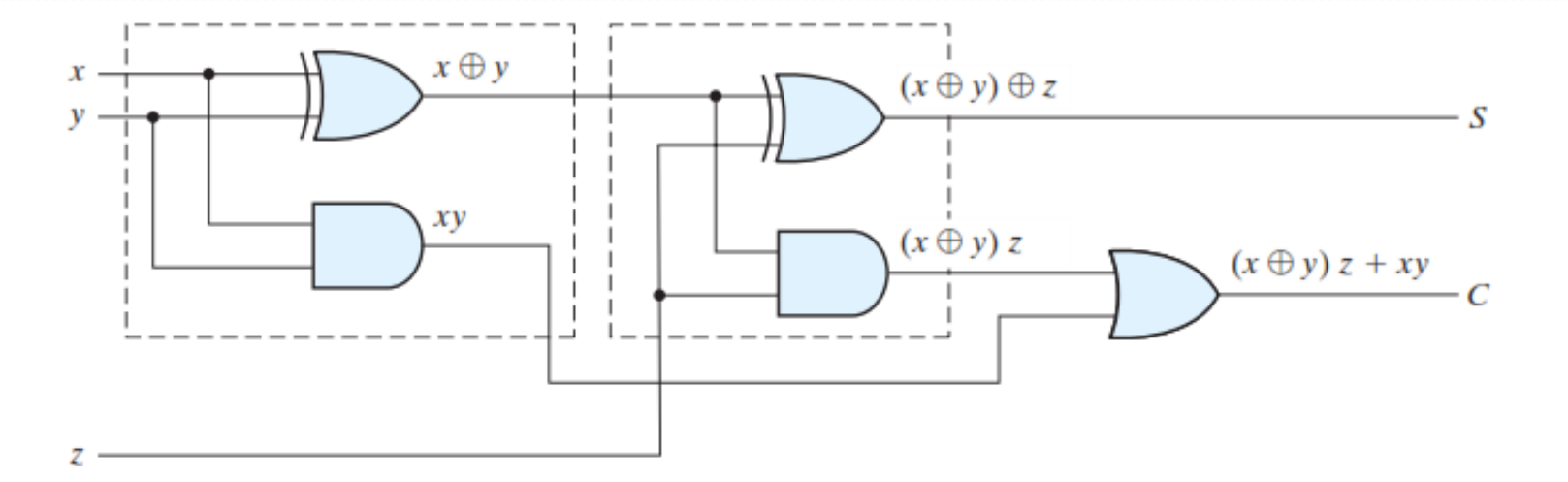


The logic diagram for the full adder implemented in sum-of-products form.



Implementation of full adder in sum-of-products form

It can also be implemented with two half adders and one OR gate,



The S output from the second half adder is the exclusive-OR of z and the output of the first half adder, giving

$$
\begin{aligned}
S &= z \oplus (x \oplus y) \\
&= z'(xy' + x'y) + z(xy' + x'y)' \\
&= z'(xy' + x'y) + z(xy + x'y') \\
&= xy'z' + x'yz' + xyz + x'y'z
\end{aligned}
$$

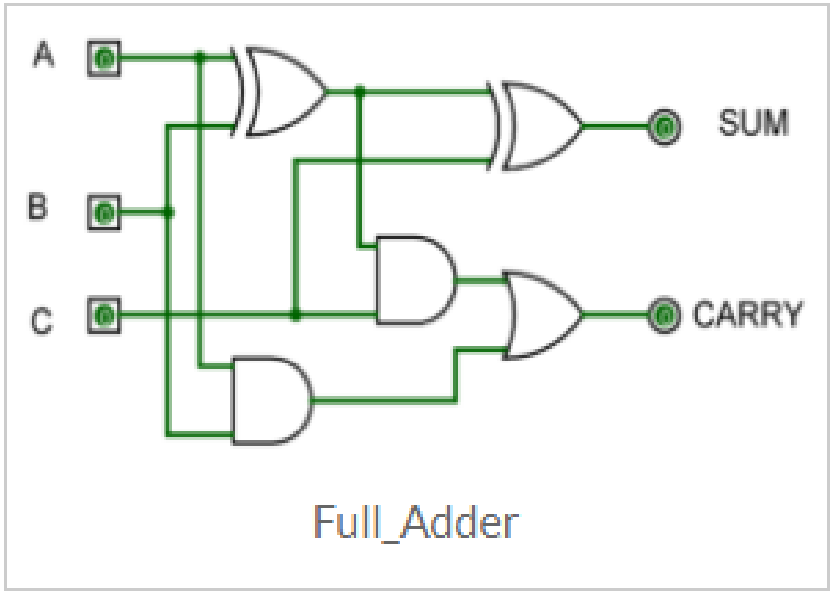
The carry output is

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

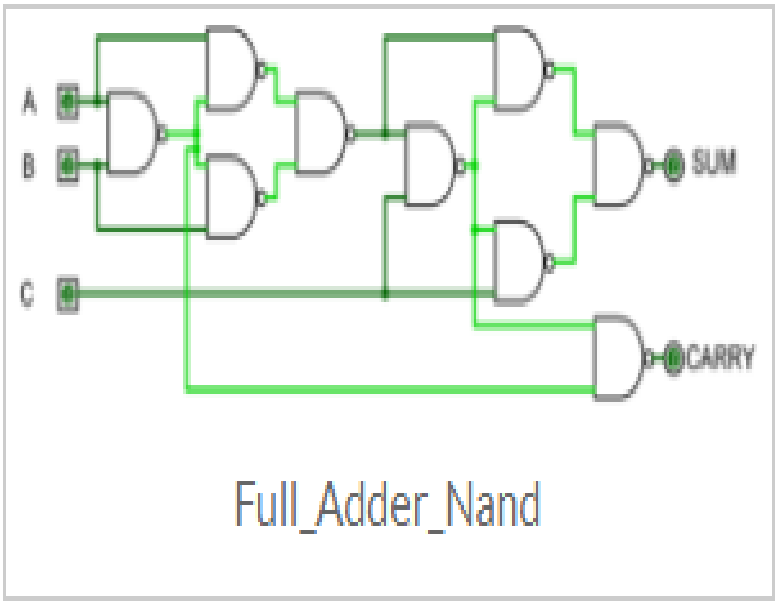
Realize Full ADDER using NAND gates only

$$Sum = A \oplus B \oplus C$$

$$Carry = A.B + (A \oplus B)C$$



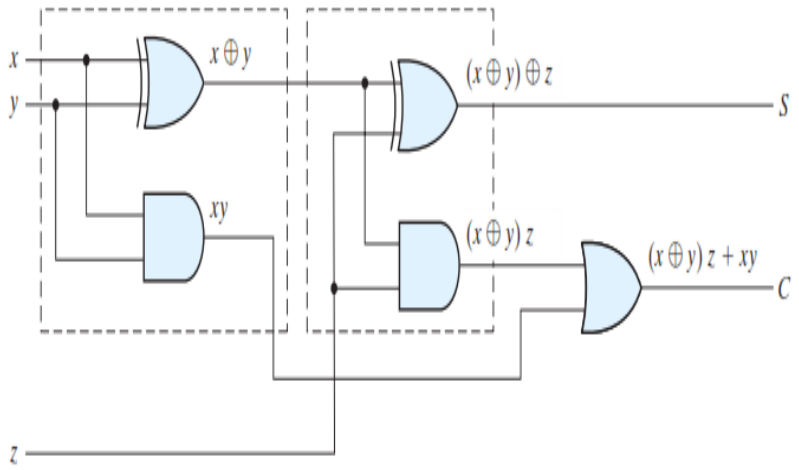
Full_Adder



Full_Adder_Nand

- HDL for Full Adder

```
// Description of full adder
//  module  full_adder (S, C, x, y, z);
//  output  S, C;
 //  input  x, y, z;
 module  full_adder ( output  S, C,  input  x, y, z);
wire S1, C1, C2;
// Instantiate half adders
half_adder HA1 (S1, C1, x, y);
 half_adder HA2 (S, C2, S1, z);
or G1 (C, C2, C1);
 endmodule
```



Binary adder

- This is also called Ripple Carry Adder ,because of the construction with full adders are connected in cascade.

Subscript i:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

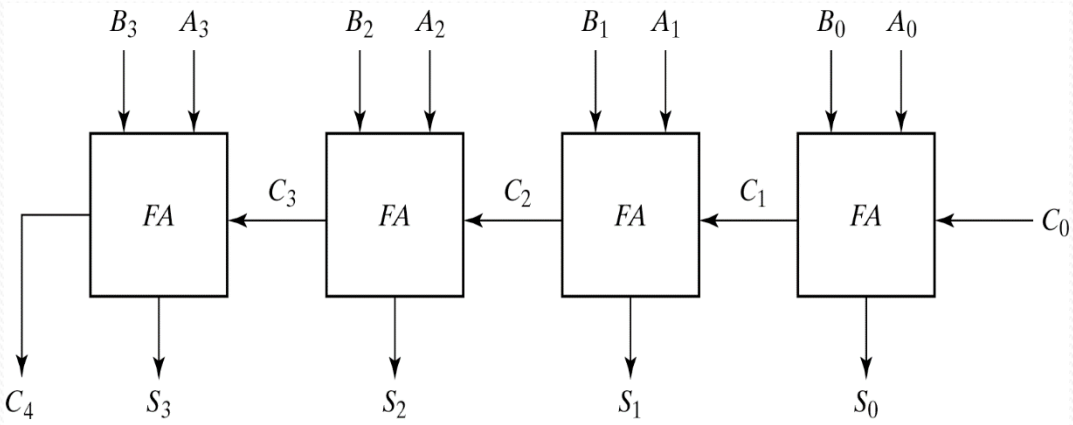


Fig. 4-9 4-Bit Adder

HDL for Ripple Carry Adder

```
module ripple_carry_4_bit_adder ( output [3: 0] Sum, output C4,
input [3: 0] A, B,
input C0);
wire C1, C2, C3;
 // Intermediate carries
// Instantiate chain of full adders
full_adder   FA0 (Sum[0], C1, A[0], B[0], C0),
                  FA1 (Sum[1], C2, A[1], B[1], C1),
                 FA2 (Sum[2], C3, A[2], B[2], C2),
                 FA3 (Sum[3], C4, A[3], B[3], C3);
 endmodule
```

Carry Propagation

- It causes a unstable factor on carry bit, and produces a longest propagation delay.
- The signal from C_i to the output carry C_{i+1}, propagates through an AND and OR gates, so, for an n-bit RCA, there are 2n gate levels for the carry to propagate from input to output.

Carry Propagation

- Because the propagation delay will affect the output signals on different time, so the signals are given enough time to get the precise and stable outputs.
- The most widely used technique employs the principle of carry look-ahead to improve the speed of the algorithm.

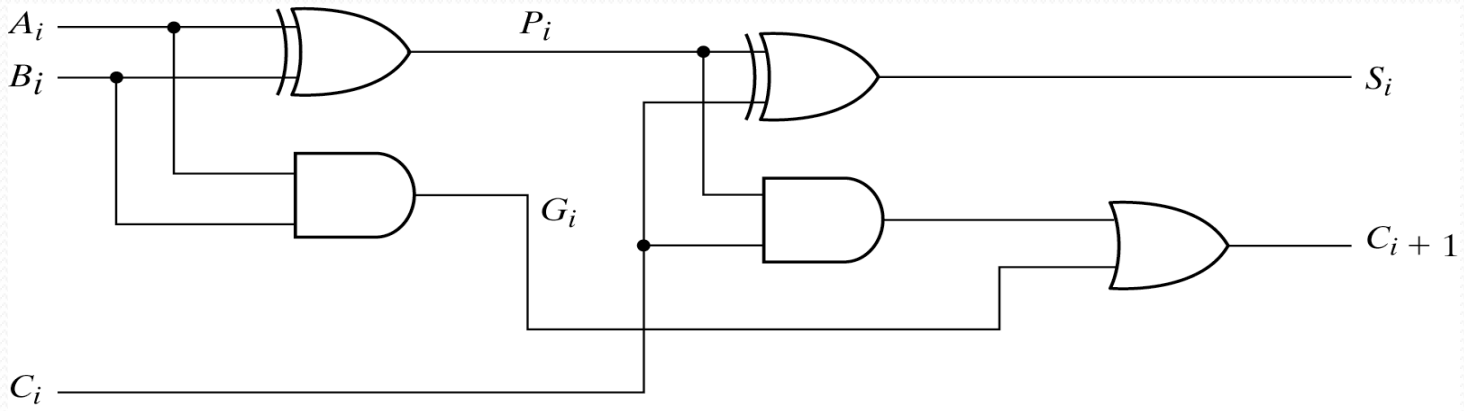


Fig. 4-10 Full Adder with P and G Shown

Boolean functions

$P_i = A_i \oplus B_i$ steady state value

$G_i = A_i B_i$ steady state value

Output sum and carry

$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

G_i : carry generate P_i : carry propagate

C_0 = input carry

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

- C_3 does not have to wait for C_2 and C_1 to propagate.

Logic diagram of carry look-ahead generator

- C_3 is propagated at the same time as C_2 and C_1.

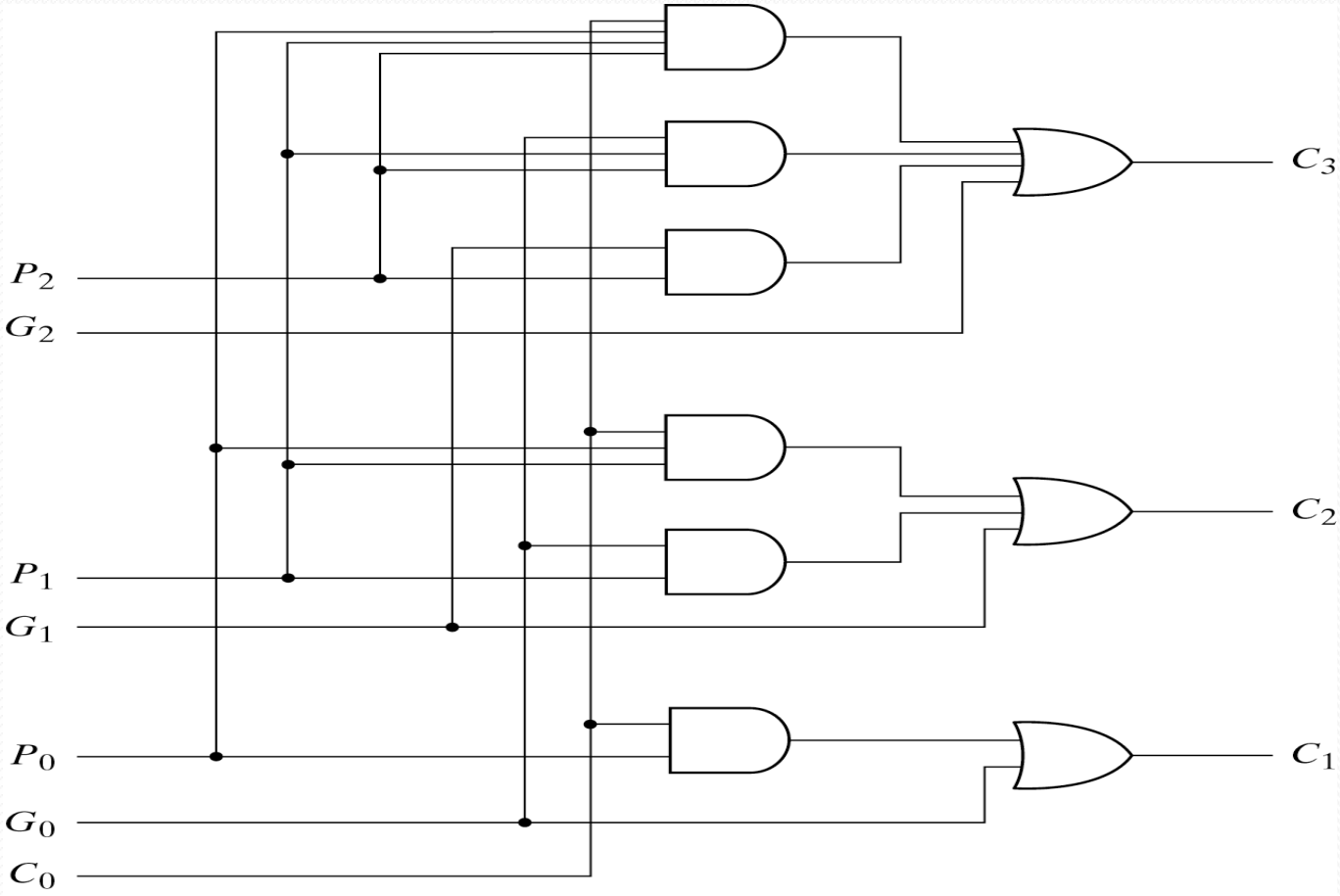


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

4-bit adder with Carry Look ahead

- Delay time of n-bit CLAA = XOR + (AND + OR) + XOR

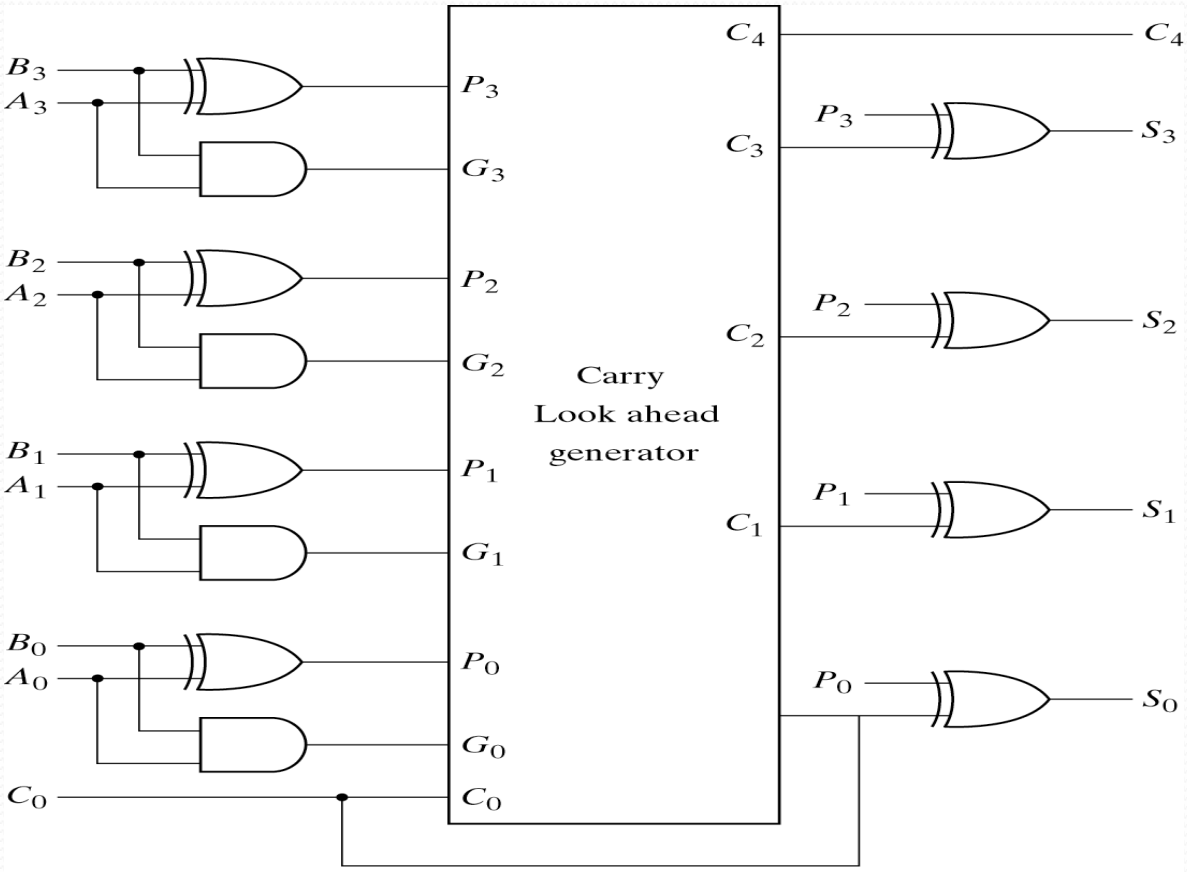


Fig. 4-12 4-Bit Adder with Carry Lookahead

THANK YOU