CSE 201: Digital Logic Design

Prepared By

Lec Sumaiya Afroz Mila

CSE, MIST

BINARY ADDER

- The most basic arithmetic operation is the addition of two binary digits.
- A combination circuit that performs the addition of two bits is **half adder**
- A adder performs the addition of 2 significant bits and a previous carry is called a **full adder**

BINARY ADDER

- Half Adder
  - Adds 1-bit plus 1-bit
  - Produces Sum and Carry

$$\begin{array}{rr} & x \\ + & y \\ \hline & C\ \ S \end{array}$$

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

BINARY ADDER

- Full Adder
  - Adds 1-bit(x) plus 1-bit(y) plus 1-bit(z)
  - Let the bit z be the carry-in bit
  - Produces Sum and Carry Out

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

$$\begin{array}{rr} & x \\ + & y \\ + & z \\ \hline C_{out} & S \end{array}$$

BINARY ADDER

- For the combination x=1, y=1 and z=1
  - If we perform the addition operation without considering the input carry (z bit)

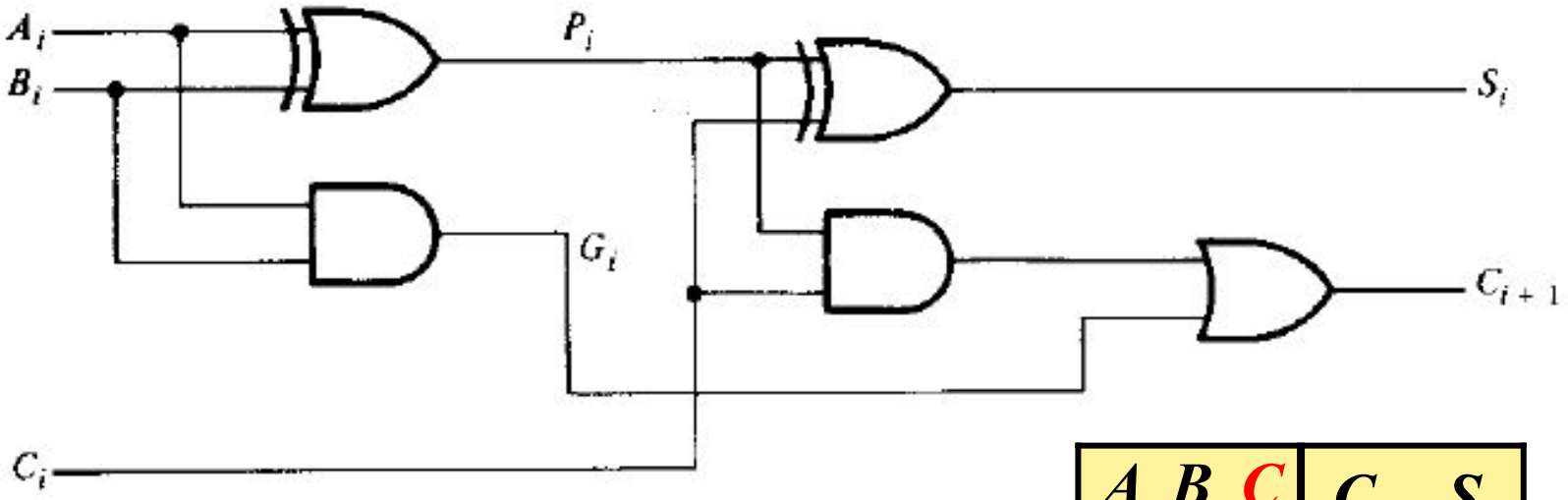
$$\begin{array}{cc} & 1 \\ + & 1 \\ \hline C_{out} & S \\ 1 & 0 \end{array}$$

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

  - If we perform the addition operation considering the input carry (z bit)

$$\begin{array}{cc} & 1 \\ & 1 \\ + & 1 \\ \hline C_{out} & S \\ 1 & 1 \end{array}$$

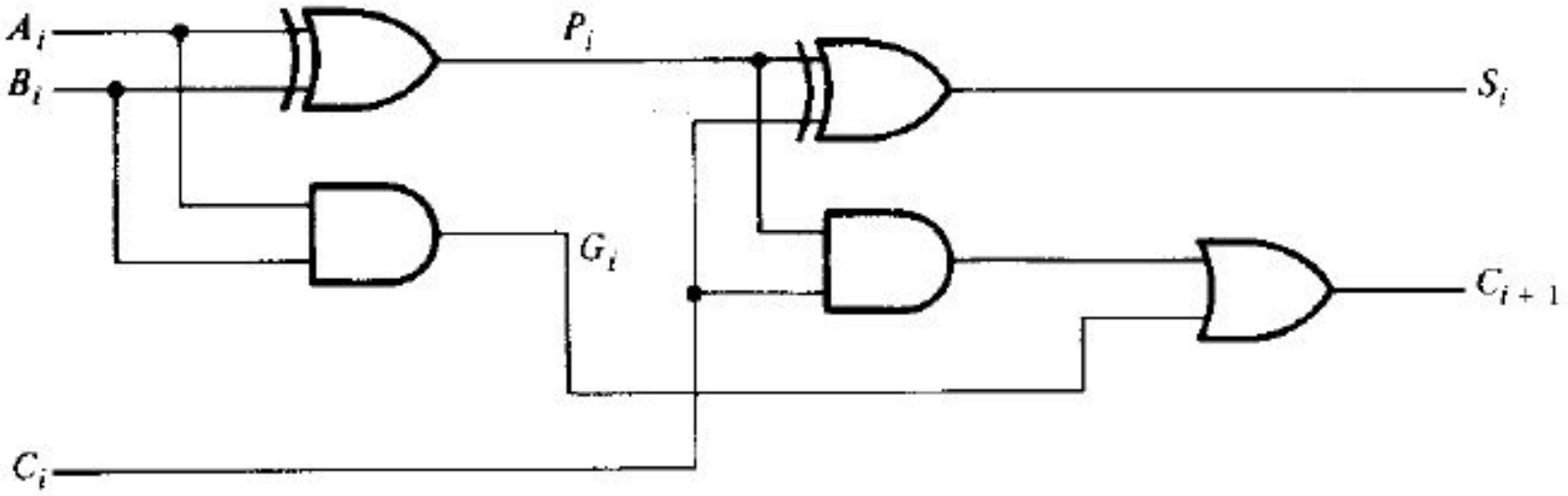
4- bit Parallel Adder using Full Adders



	0	1	1	0	
	1	0	1	1	
	0	0	1	1	
	1	1	1	0	

A_i	B_i	C_i	C_{i+1}	S_i
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

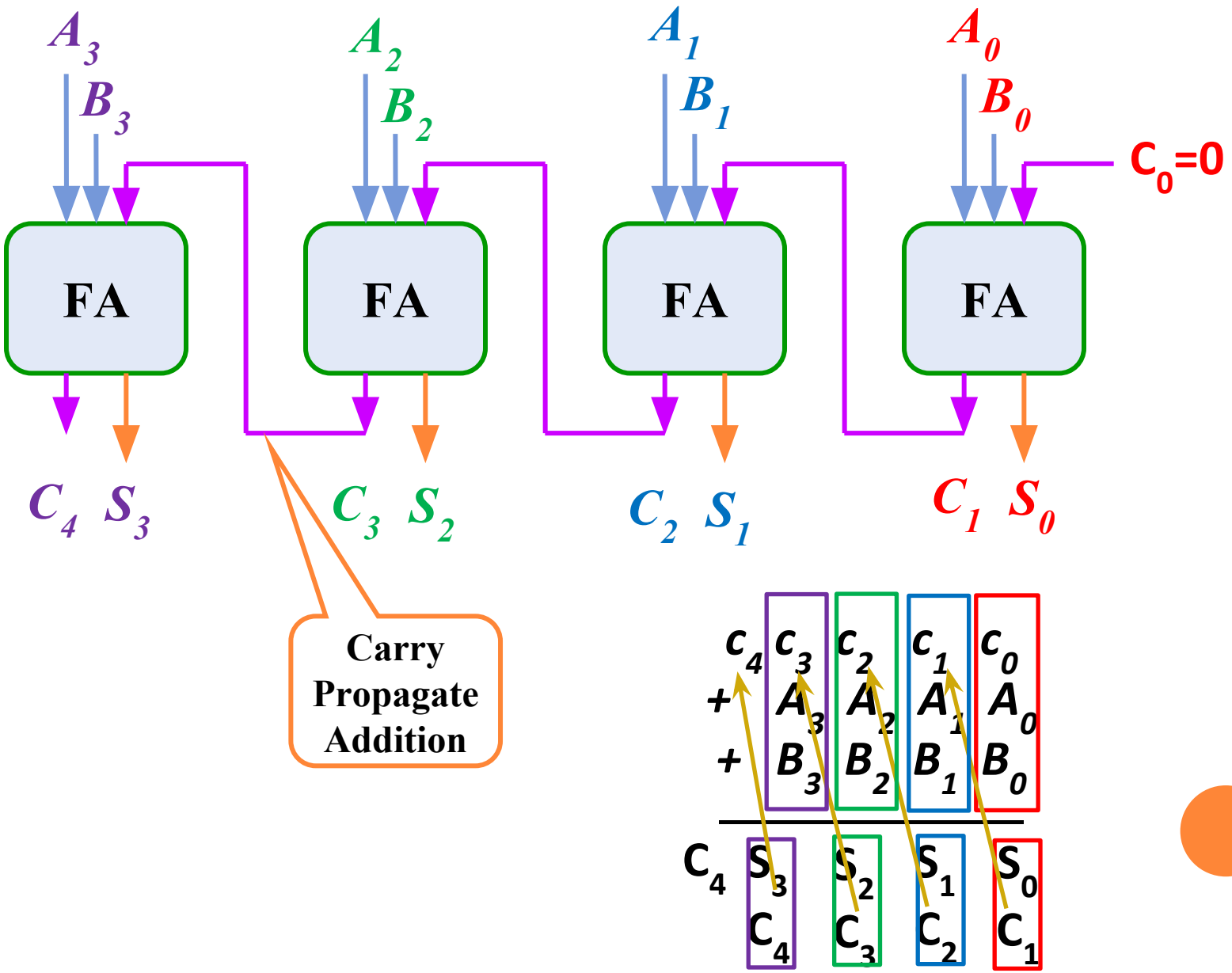
4- bit Parallel Adder using Full Adders



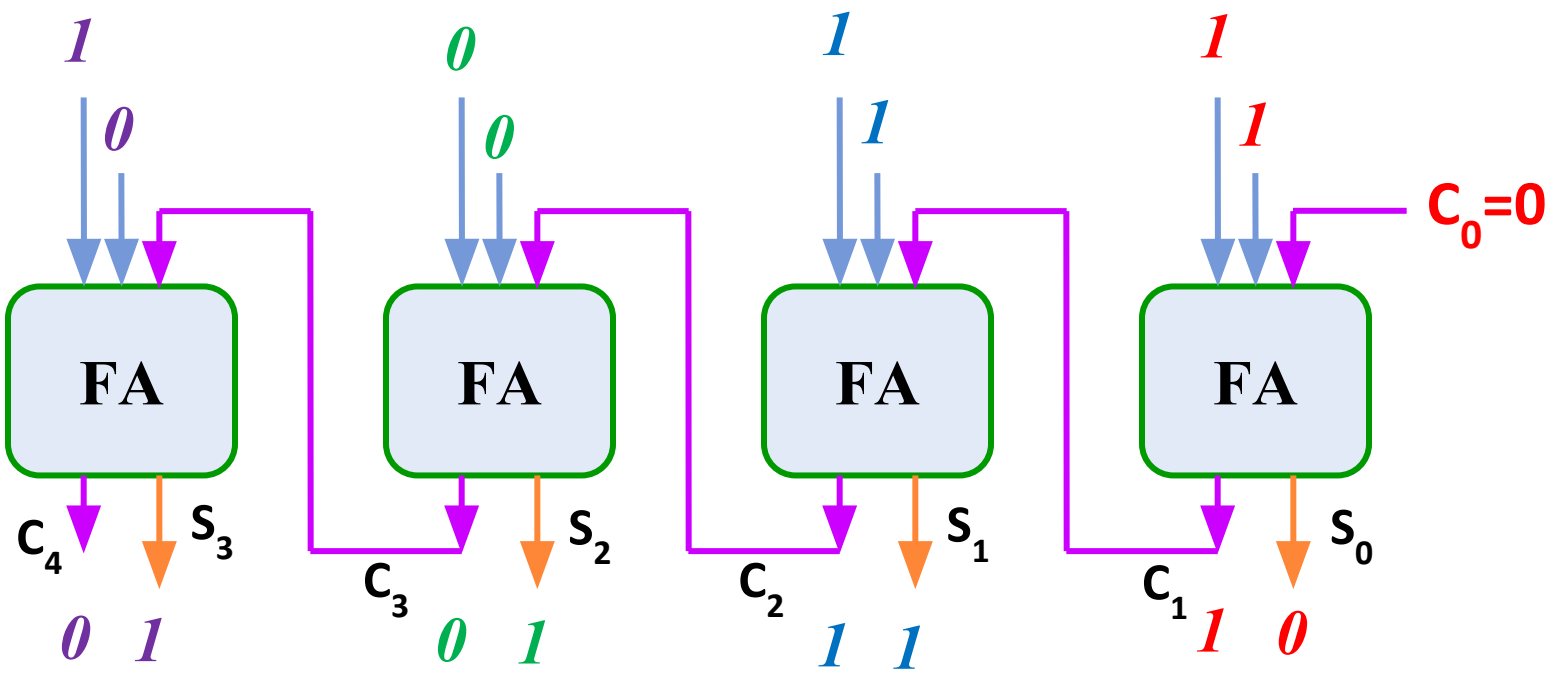
$$\begin{array}{cccc} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 \end{array}$$

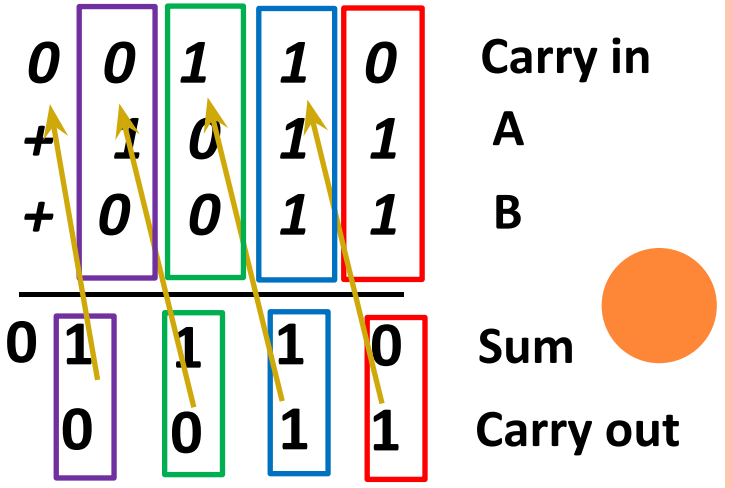
$$\begin{array}{ccccc} C_4 & C_3 & C_2 & C_1 & C_0 \\ + & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline C_4 & S_3 & S_2 & S_1 & S_0 \\ & C_4 & C_3 & C_2 & C_1 \end{array}$$

4- bit Parallel Adder using Full Adders

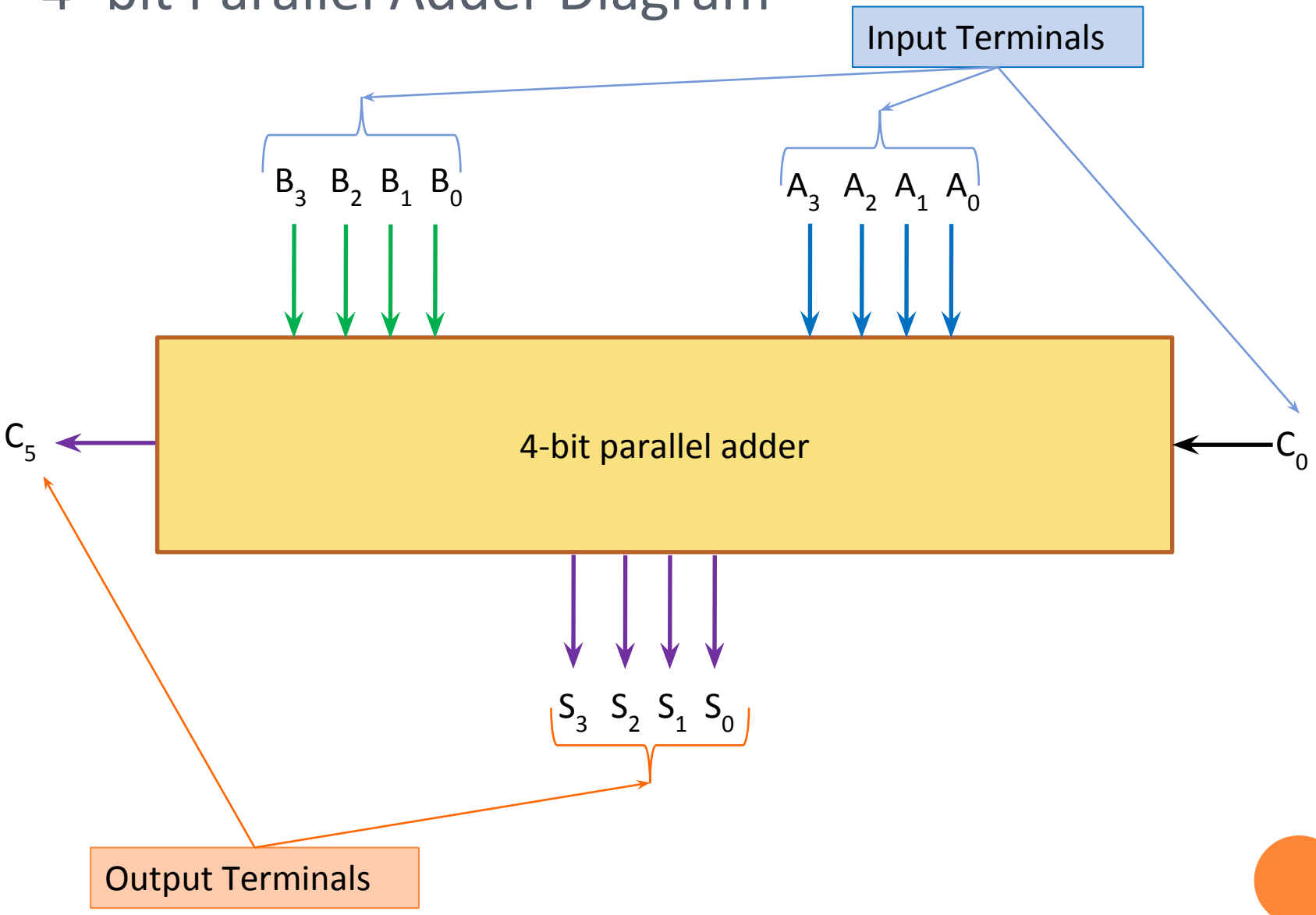


4- bit Parallel Adder using Full Adders





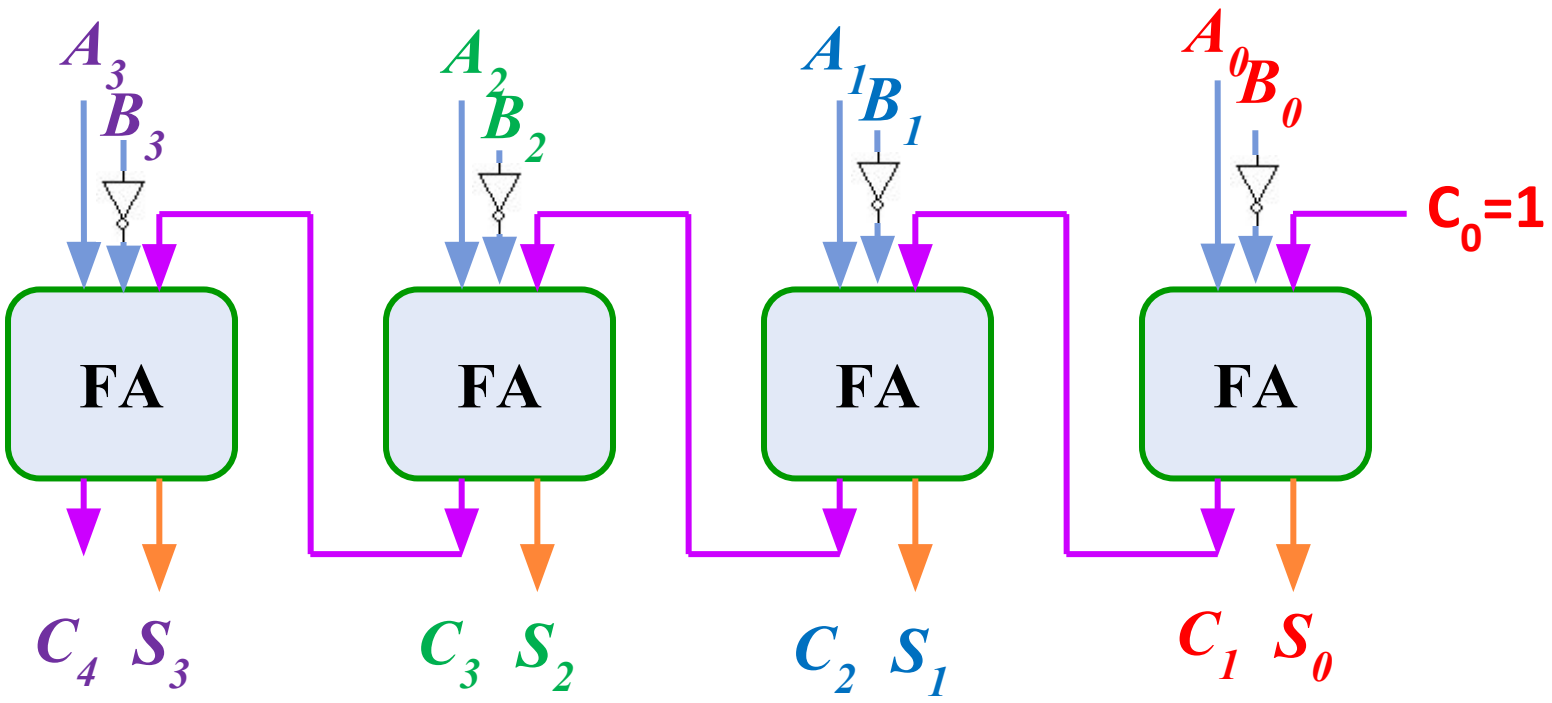
4- bit Parallel Adder Diagram



BINARY SUBTRACTOR

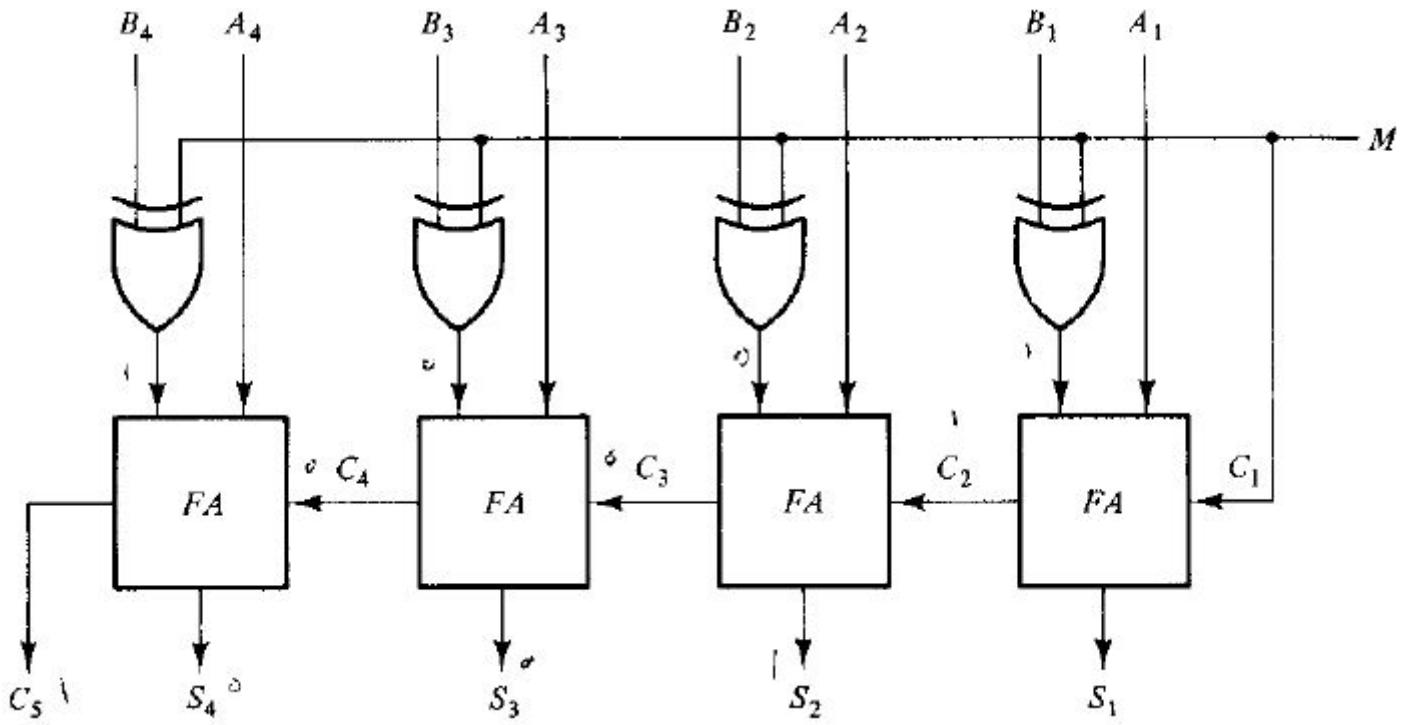
- The subtraction operation (A-B) can be done by (A+ 2's complement of B)
  1. Take 1's complement or invert of B
  2. Add A with 1's complement of B
  3. Add 1 with the least significant bits of the previous addition
- This can be achieved by modifying the n-bit parallel adder by
  1. adding an inverter in input of B
  2. Input carry C_0 must be equal to 1
- For unsigned numbers-
  - This operation gives (A-B) if $A \geq B$
  - And 2's complement of B-A if $A \leq B$

4- bit Parallel Subtractor using Full Adders



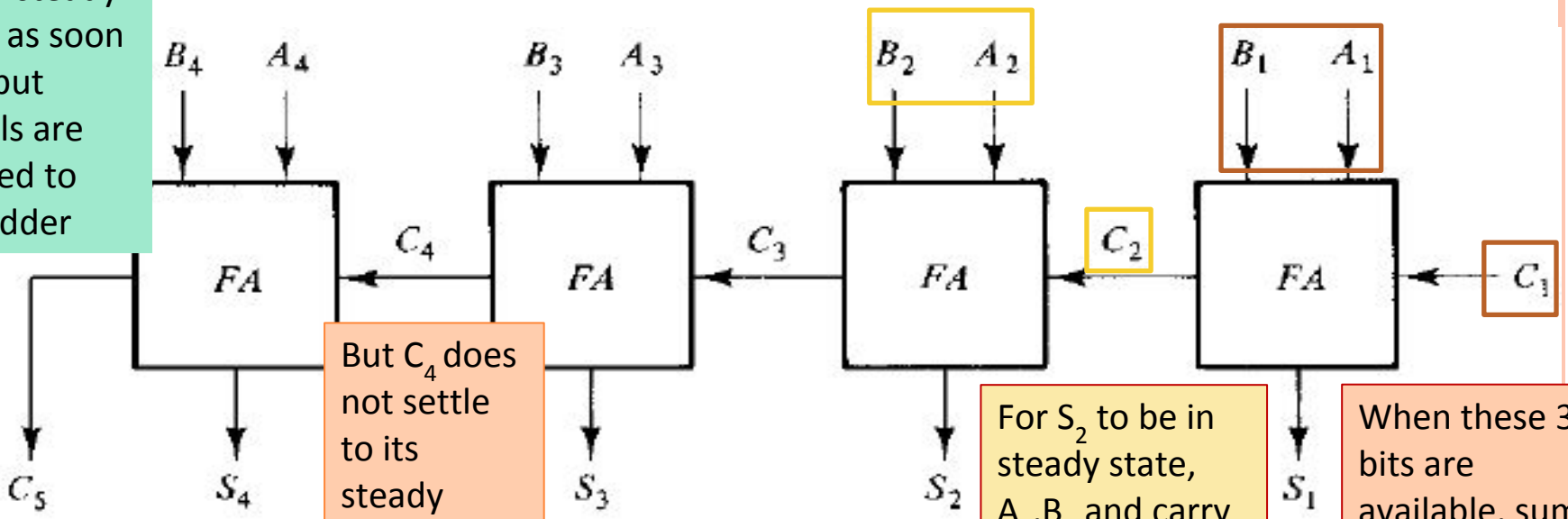
$$\begin{array}{l} A_3\ A_2\ A_1\ A_0 \\ +B'_3\ B'_2\ B'_1\ B'_0 \\ +\qquad\qquad 1 \end{array}$$

4- bit Adder - Subtractor



Limitations in Parallel Adder

Input A_4 & B_4 reach steady state as soon as input signals are applied to the adder



But C_4 does not settle to its steady state until C_3 is available

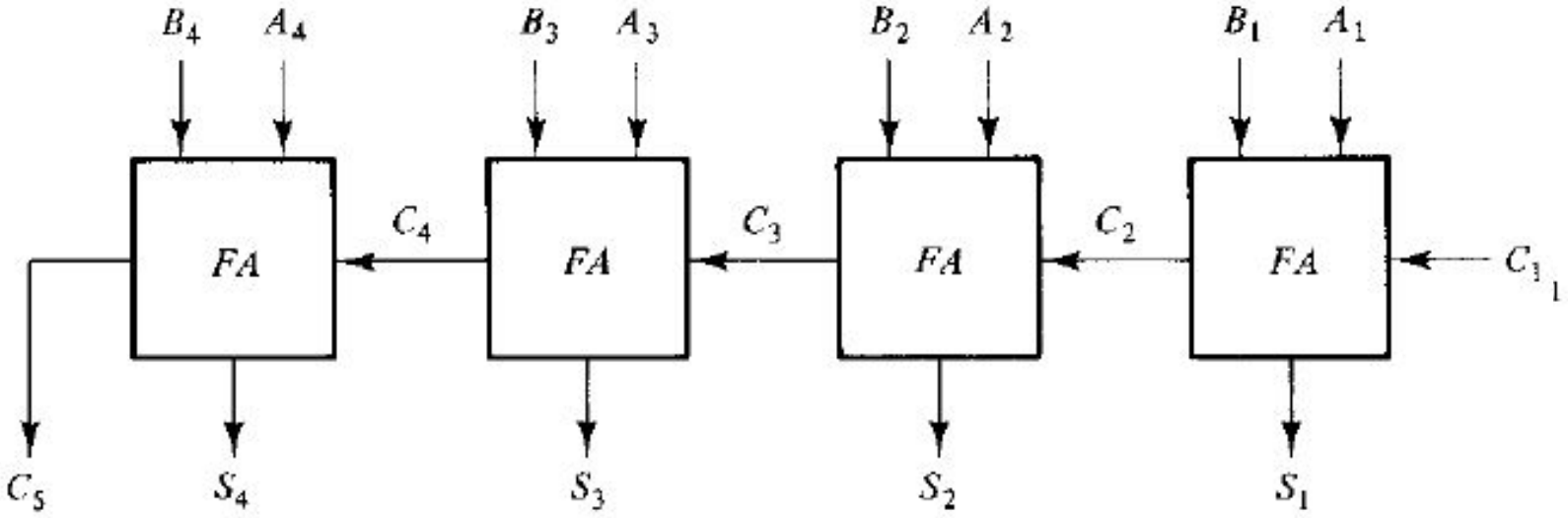
For S_4 to be in steady state, A_4,B_4 and carry C_4 must be available

For S_2 to be in steady state, A_2,B_2 and carry C_2 must be available

When these 3 bits are available, sum bit S_1 will be in steady state

- All bits of addend and augend should be available for computation at the same time
- Each bit of sum output depends on the value of the input carry
- Similarly C_3 has to wait for C_2, and C_2 has to wait for C_1
- Thus only after the carry propagates through all the stages, S_4 will be in its steady state

Limitations in Parallel Adder



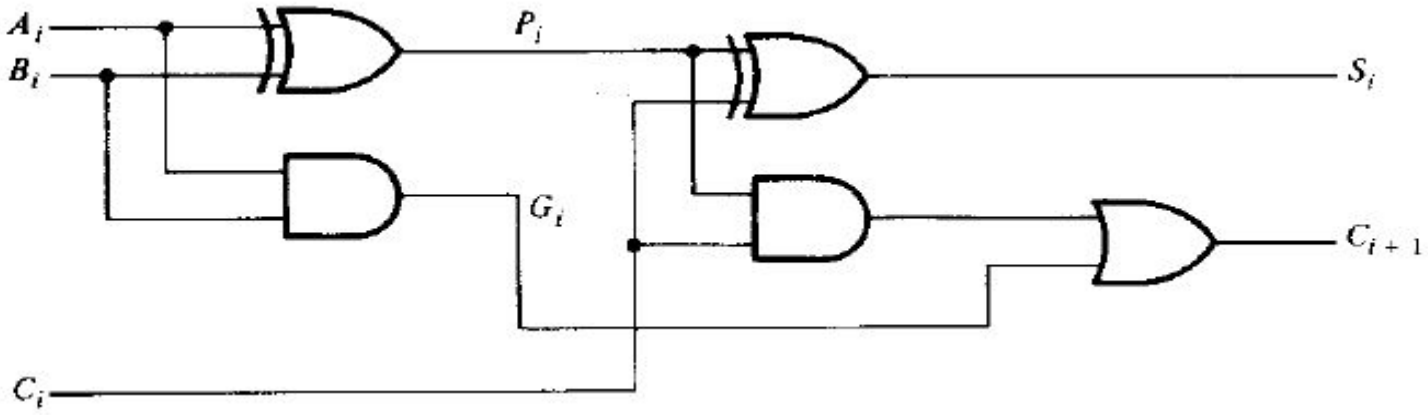
- Carry propagation from C1 to C4 causes propagation delay
- The signal from the input carry C_i to the output carry C_{i+1}, propagates through an **AND gate**+ an **OR gate**, means **two gate levels**.
- C_2 has 2 gate levels from C_1 to C_2
- C3 has 2 gate levels from C_2 to C_3 and 4 gate levels from C_1 to C_3
- Similarly, in the previous design with 4 full adders, C_5 has 4*2=8 gate levels from C_1 to C_5
- So, the total propagation time would be, propagation time in one half adder plus 8 gate levels

Limitations in Parallel Adder

- **Carry propagation time** is a limiting factor on the speed
- Though the parallel adder will always have some value in the output terminals, outputs will not be correct unless signals are given enough time to propagate through the gates connected from the inputs to the outputs

- One technique to **reduce** the carry propagation time is ***look-ahead carry***

LOOK-AHEAD CARRY



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

Output sum and carry can be expressed as:

$$S_i = P_i \oplus C_i$$

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

Booelan Function for carry output of each stage:

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

C_5 can be derived by the equation substitution method:

$C_5 = G_4 + P_4 C_4$

$= G_4 + P_4(G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1)$

LOOK-AHEAD CARRY

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

- Boolean function for each output carry is expressed in sum of products.
- Each function can be implemented with one level of **AND gate**, followed by an **OR gate**.
- Carry outputs depend on the variable P and G. P_i and G_i are functions of input A_i and B_i which are available as soon as inputs are applied to the adder.

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

- In this design, C_4 does not have to wait for C_3 and C_2 to propagate.
- In fact, C_4 is propagated at the same time as C_3 and C_2

LOOK-AHEAD CARRY

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

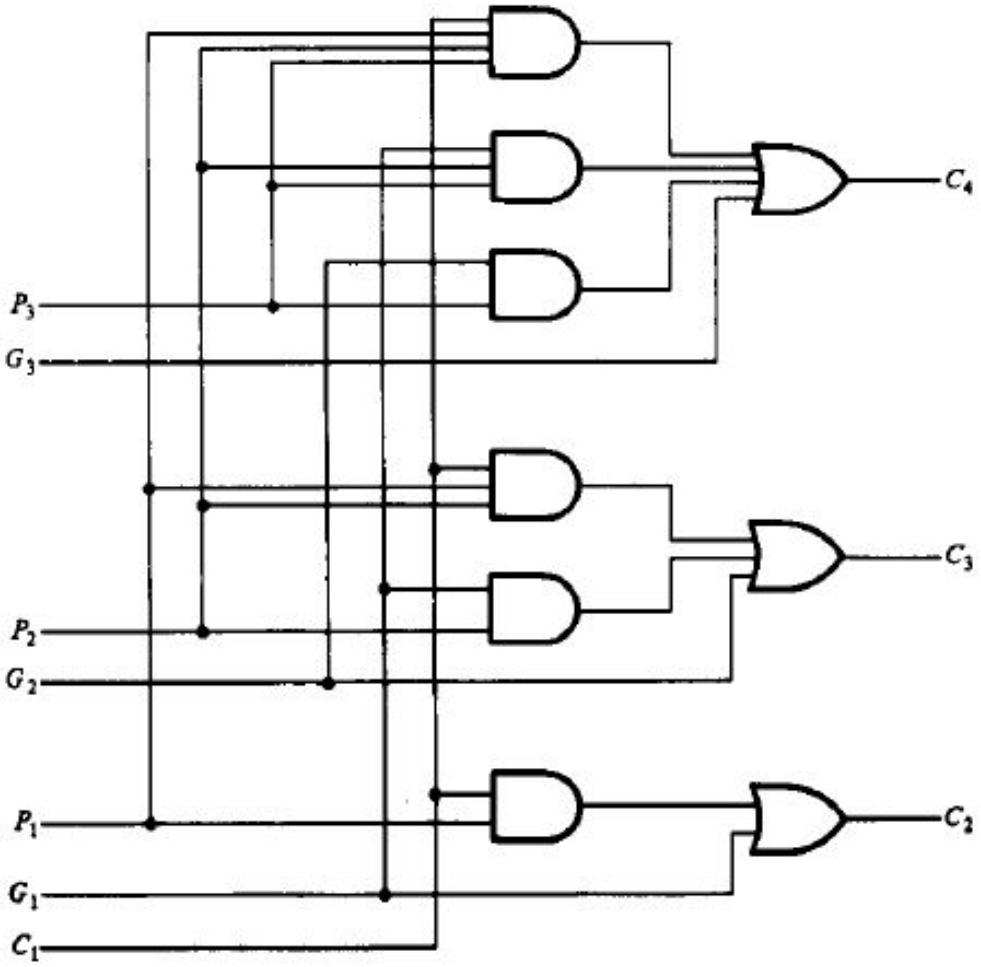
$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$C_5 = G_4 + P_4 C_4$

$= G_4 + P_4(G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1)$

Connection of C_5 is not shown here. However, we can add the connection for C_5



4-BIT FULL ADDERS WITH LOOK-AHEAD CARRY

$S_i = P_i \text{ xor } C_i$

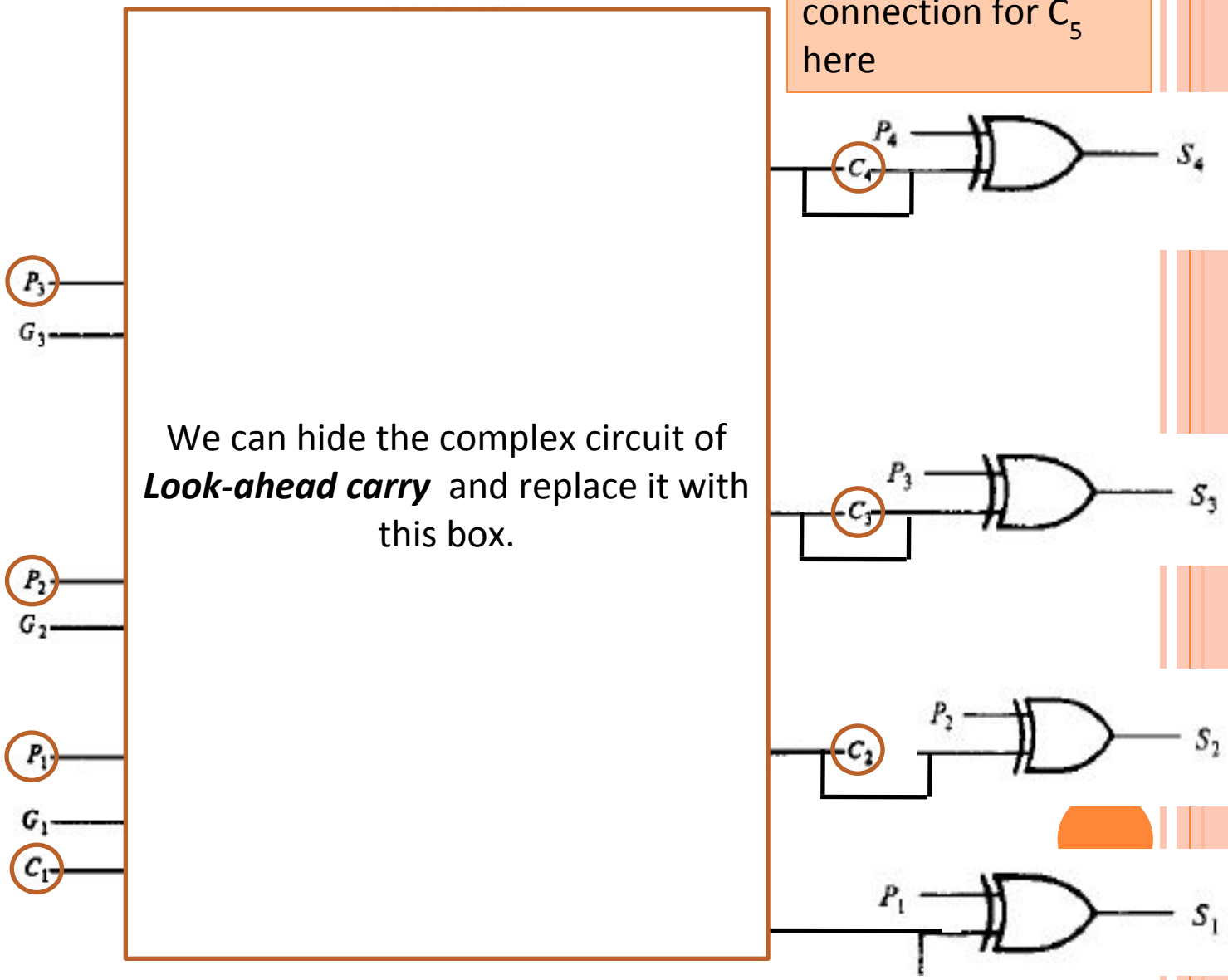
$S_1 = P_1 \text{ xor } C_1$

$S_2 = P_2 \text{ xor } C_2$

$S_3 = P_3 \text{ xor } C_3$

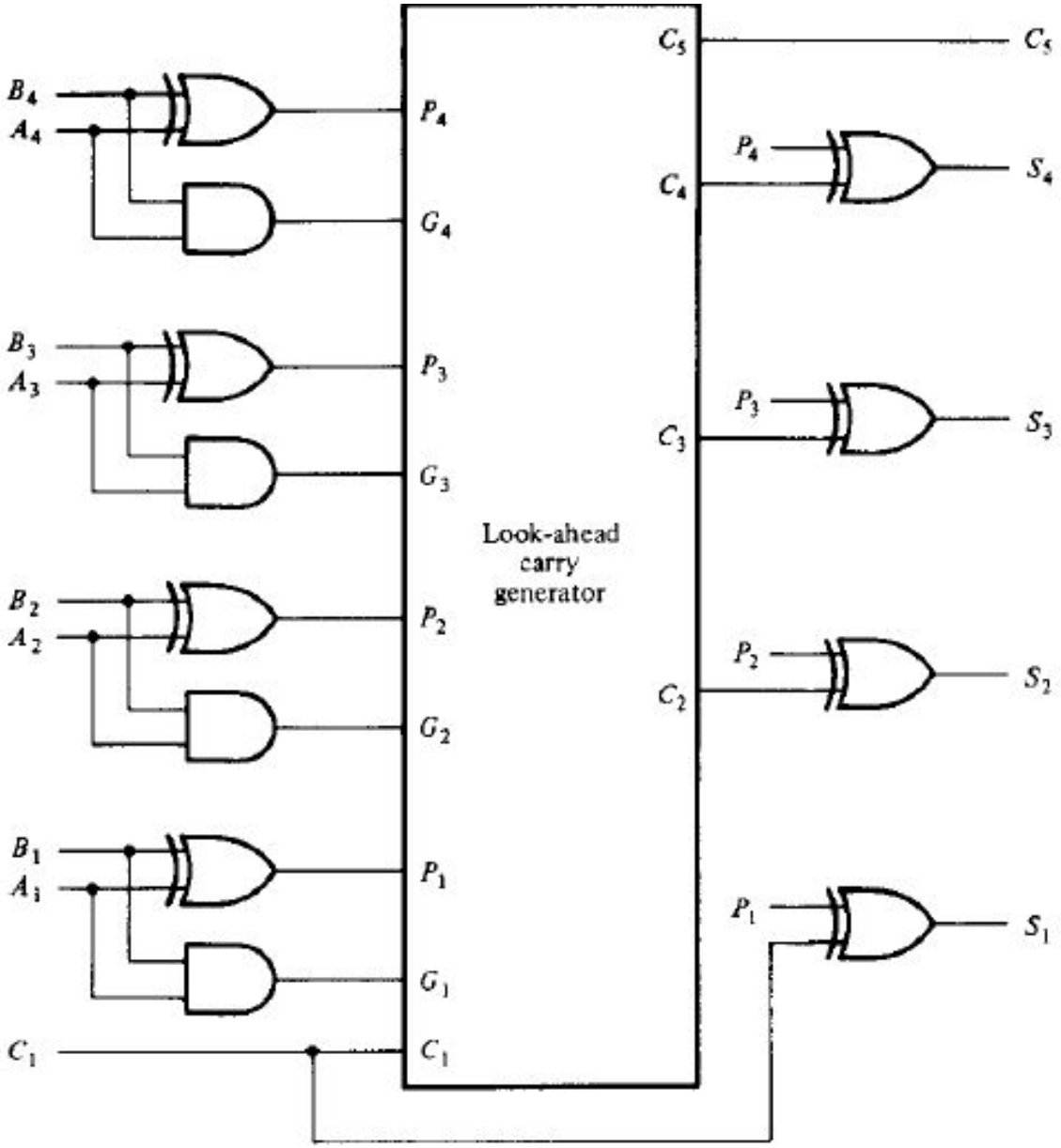
$S_4 = P_4 \text{ xor } C_4$

we have to add the connection for C_5 here



4-BIT FULL ADDERS WITH LOOK-AHEAD CARRY

So, the new design can be drawn like this



Logic Circuits

- Logic Circuits: Combinational and Sequential
- Combinational Circuits
  - A combinational circuit consists of logic gates whose outputs at any time are determined from **only the present combination of inputs.**
- Sequential Circuits
  - A sequential circuits employ storage elements and logic gates.
  - **The outputs are a function of the inputs and the state of the storage elements.**
  - The state of the storage elements, in turn, is a function of the previous inputs (and the previous state).