INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

# Carry Lookahead Adder

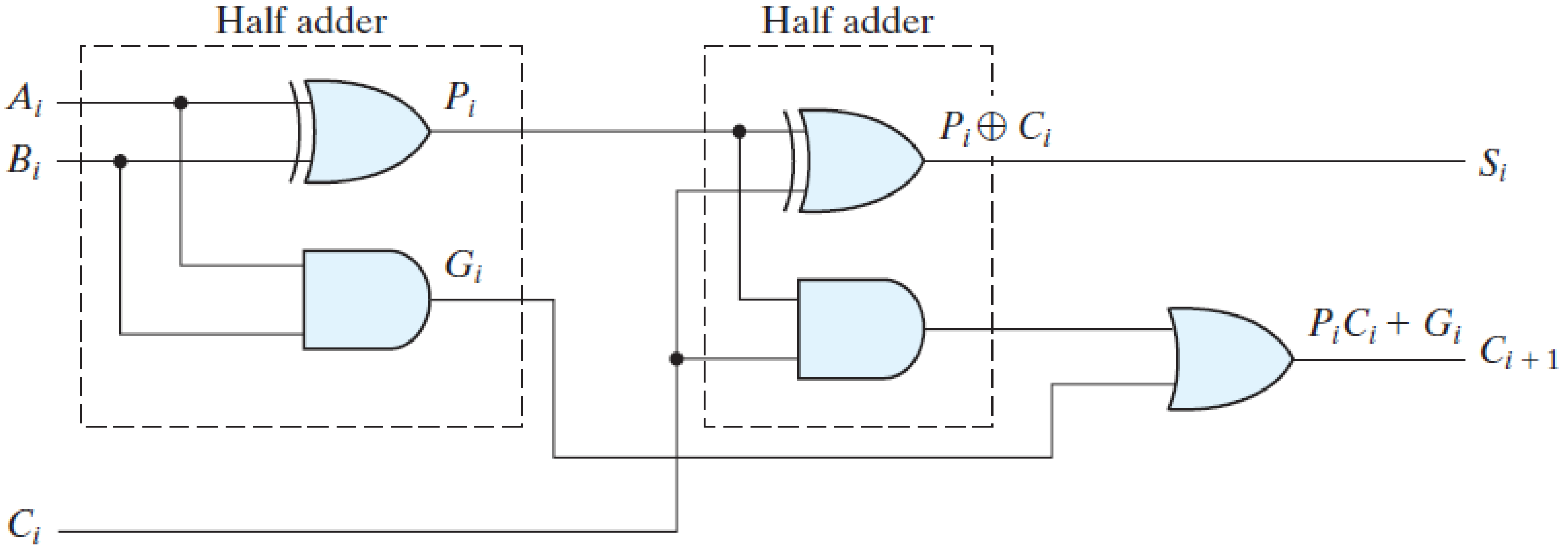
Sparsh Mittal

# Carry Propagation

- In any combinational circuit, the signal must propagate through the gates before the correct output is available in the output terminals.
- The total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the circuit.
- The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders.
- Since each bit of the sum output depends on the value of the input carry, the value of $S_i$ at any given stage in the adder will be in its steady-state final value only after the input carry to that stage has been propagated.

- The number of gate levels for the carry propagation can be found from the circuit of the full adder.

- $P_i$ and $G_i$ depend only on input.
- The signal from the input carry $C_i$ to the output carry $C_{i+1}$ propagates through an AND gate and an OR gate, which constitute two gate levels.



Redrawn circuit for each stage (i) of adder

# Total delay

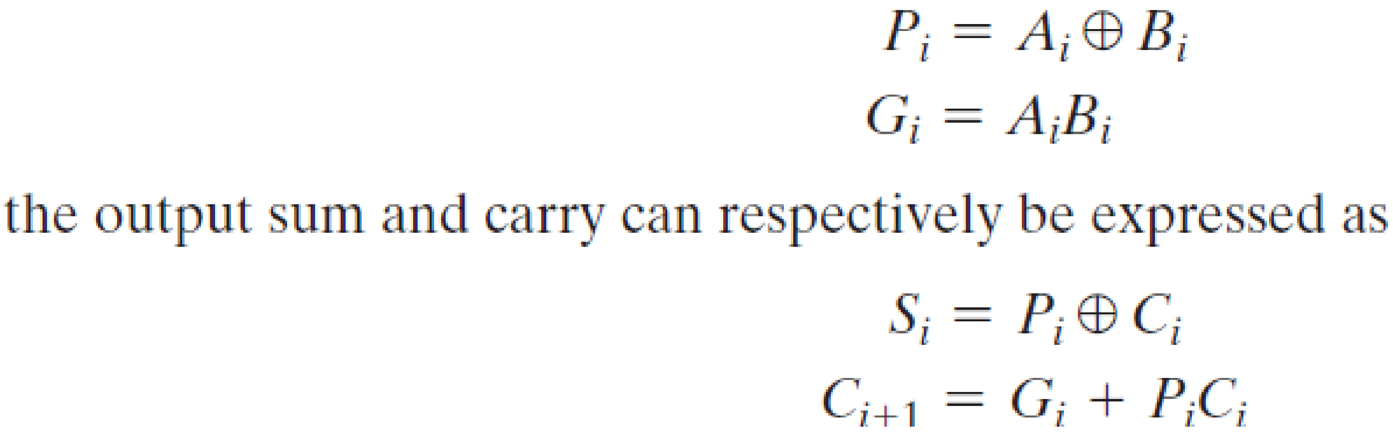
- If there are four full adders in the adder, the output carry $C4$ would have 2 * 4 = 8 gate levels from $C0$ to $C4$.
- For an $n$ -bit adder, there are $2n$ gate levels for the carry to propagate from input to output.

# Reducing carry propagation delay

- The carry propagation time of the adder limits the speed with which two numbers are added.
- Although the adder will always have some value at its output terminals, the outputs will not be correct unless the signals are given enough time to propagate through the gates connected from the inputs to the outputs.
- **How to reduce carry propagation delay?**

1. Employ faster gates with reduced delays.
2. Increase equipment complexity such that the carry delay time is reduced. The most widely used technique employs the principle of carry lookahead logic.

- Define 2 new variables



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

the output sum and carry can respectively be expressed as

$$S_i = P_i \oplus C_i$$

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

- *Gi* is called a *carry generate* , and it produces a carry of 1 when both *Ai* and *Bi* are 1, regardless of the input carry *Ci*.
- *Pi* is called a *carry propagate* , because it determines whether a carry into stage *i* will propagate into stage *i* + 1

- We now write the Boolean functions for the carry outputs of each stage and substitute the value of each *Ci* from the previous equations:

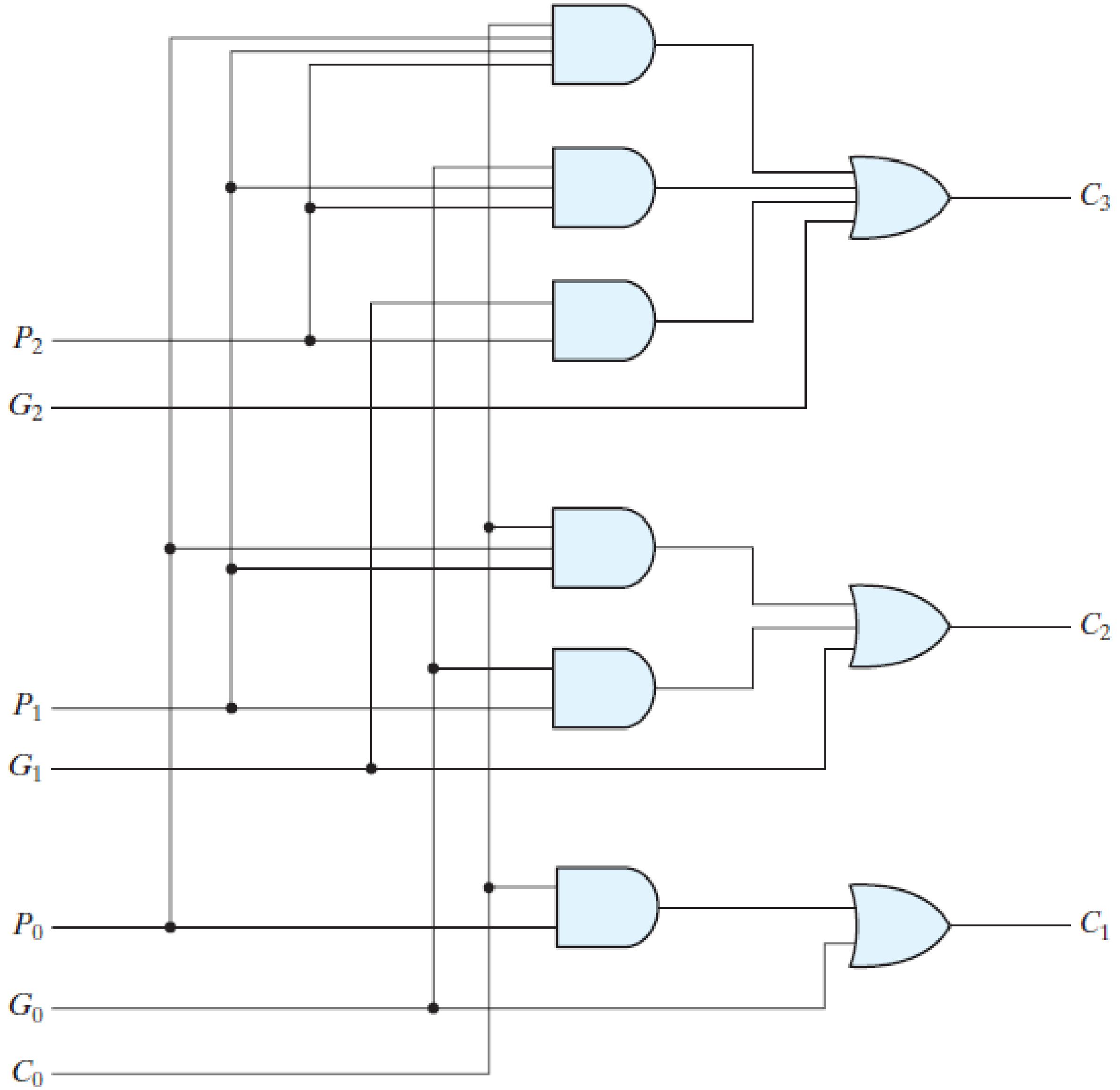
$$C_0 = \text{input carry}$$
$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

- The Boolean function for each output carry is expressed in sum-of-products form, it can be implemented with two-levels: AND and OR.

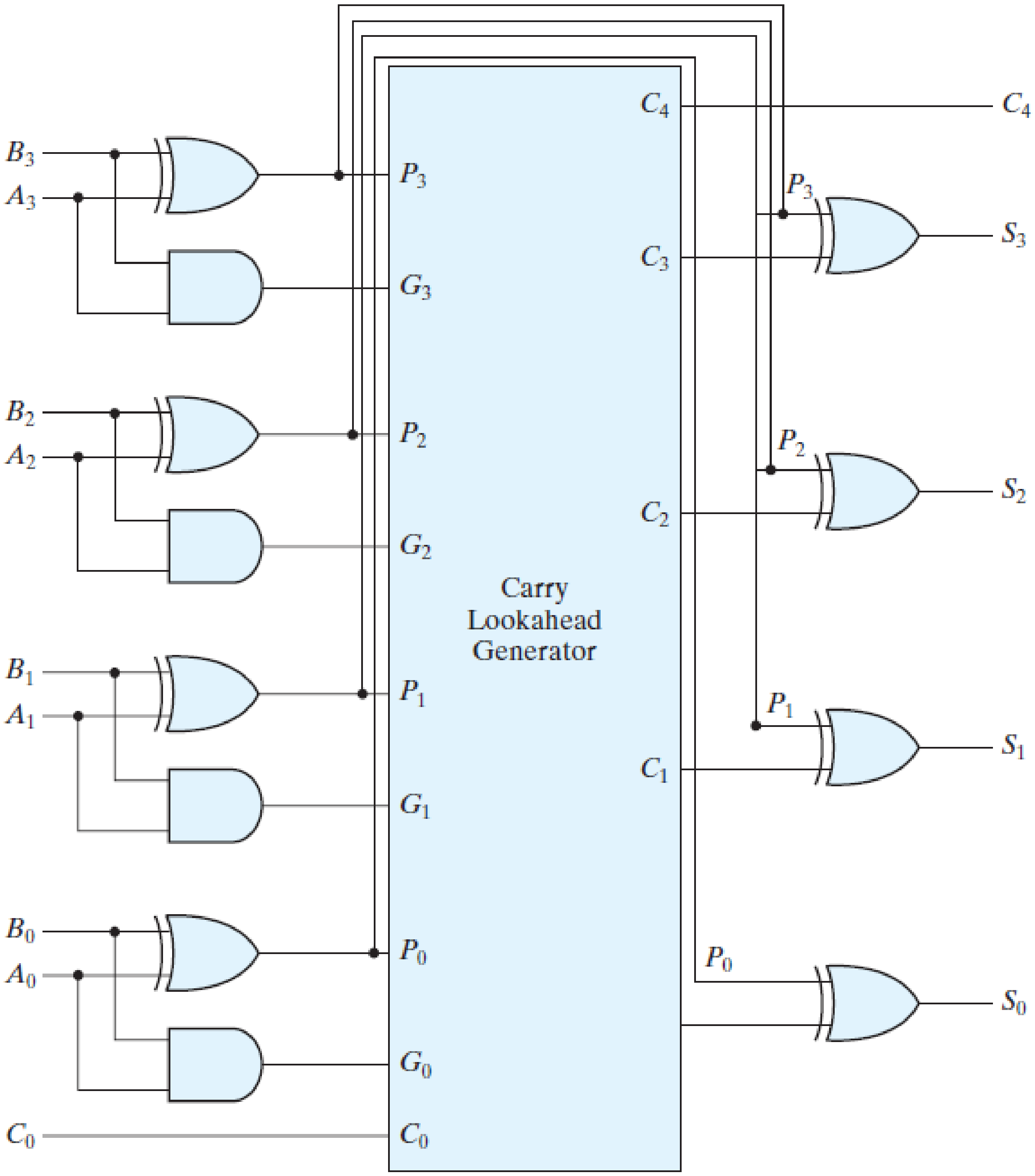
# Carry lookahead (CLA) generator

- Three Boolean functions for C1, C2, and C3 are implemented in CLA generator shown here
- This circuit can add in less time because C3 does not have to wait for C2 and C1 to propagate.
- In fact, C3 is propagated at the same time as C1 and C2.
- This gain in speed of operation is achieved at the expense of additional hardware.



# 4-bit CLA adder

- Each sum output requires two XOR gates.
- Output of first XOR gate generates *Pi* variable, and AND gate generates *Gi* variable.
- Carries are propagated through CLA and applied as inputs to the second XOR gate.
- All output carries are generated after a delay through 2 gates.
- Outputs *S*1 through *S*3 have equal propagation delay times.


$B_3$
$A_3$
$B_2$
$A_2$
$B_1$
$A_1$
$B_0$
$A_0$
$C_0$
$P_3$
$G_3$
$P_2$
$G_2$
$P_1$
$G_1$
$P_0$
$G_0$
$C_0$
Carry Lookahead Generator
$C_4$
$C_3$
$C_2$
$C_1$
$C_4$
$P_3$
$S_3$
$P_2$
$S_2$
$P_1$
$S_1$
$P_0$
$S_0$