Combinational Logic Design

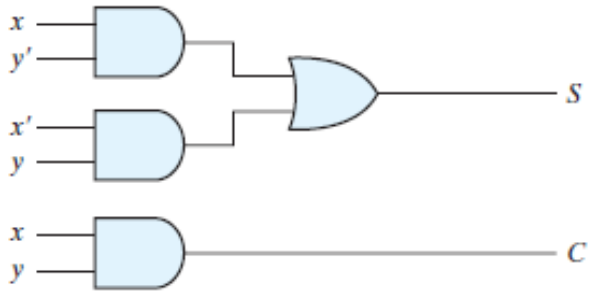
Dr. Chandan Karfa
CSE IIT Guwahati

- Adder

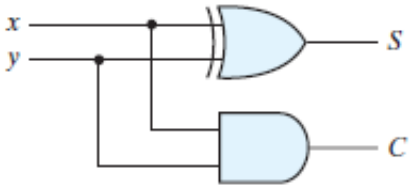
Half Adder

Table 4.3
Half Adder

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



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Full adder

- For example, addition of the binary numbers 1011 and 0011

0 1 1 carry-in

1 0 1 1 augend

0 0 1 1 addend

---

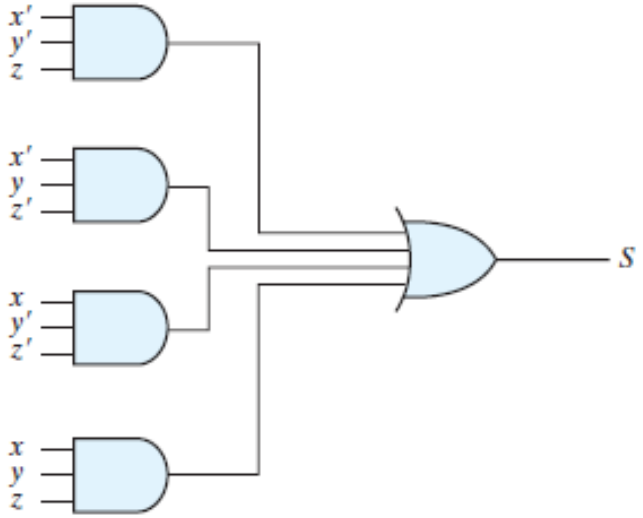
1 1 1 0 sum

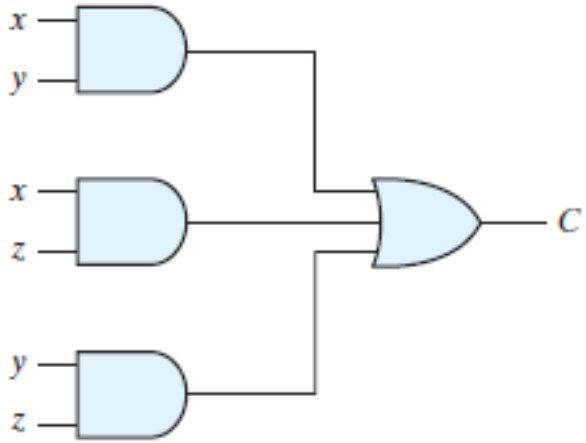
- The carry-out produced in the addition of the ith significant digits must be incorporated, as a carry-in, in the addition process for the (i + 1)th significant digit.

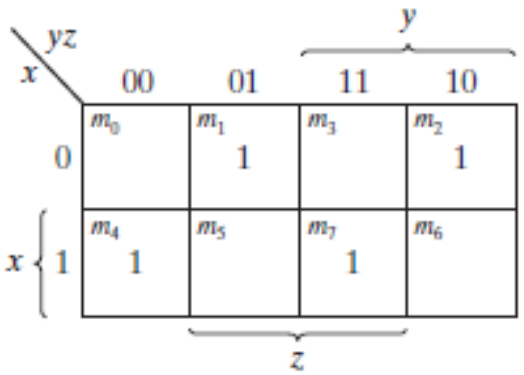
Full Adder

Table 4.4
Full Adder

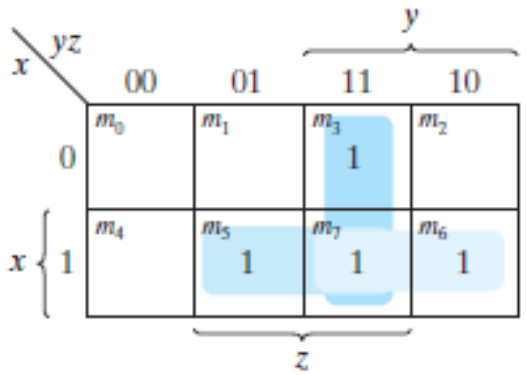
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





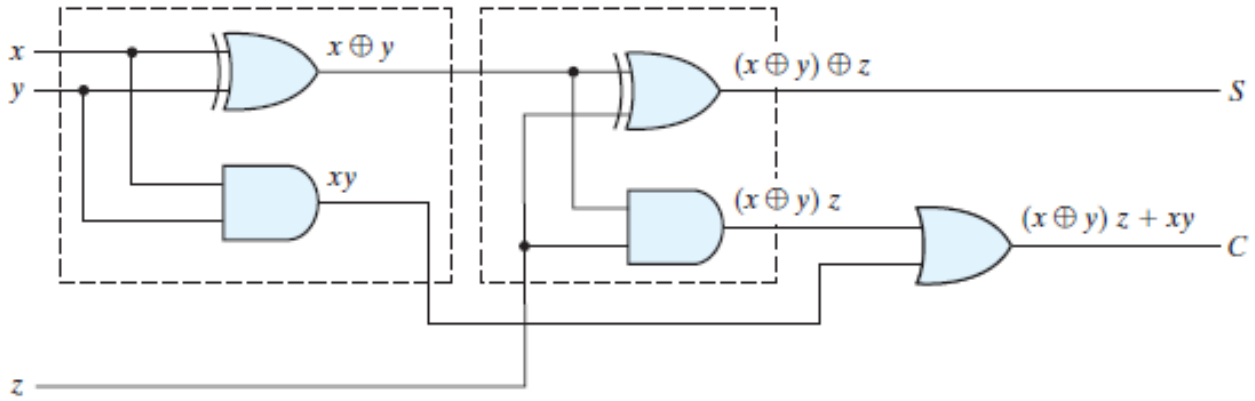


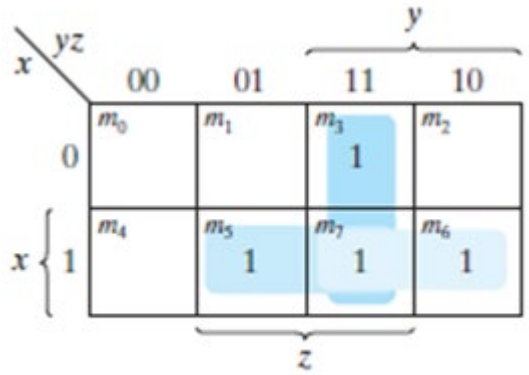
(a) $S = x'y'z + x'yz' + xy'z' + xyz$



(b) $C = xy + xz + yz$

Full Adder – area optimized design





(b) $C = xy + xz + yz$

Table 4.4
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

$$= xy'z' + x'yz' + xyz + x'y'z$$

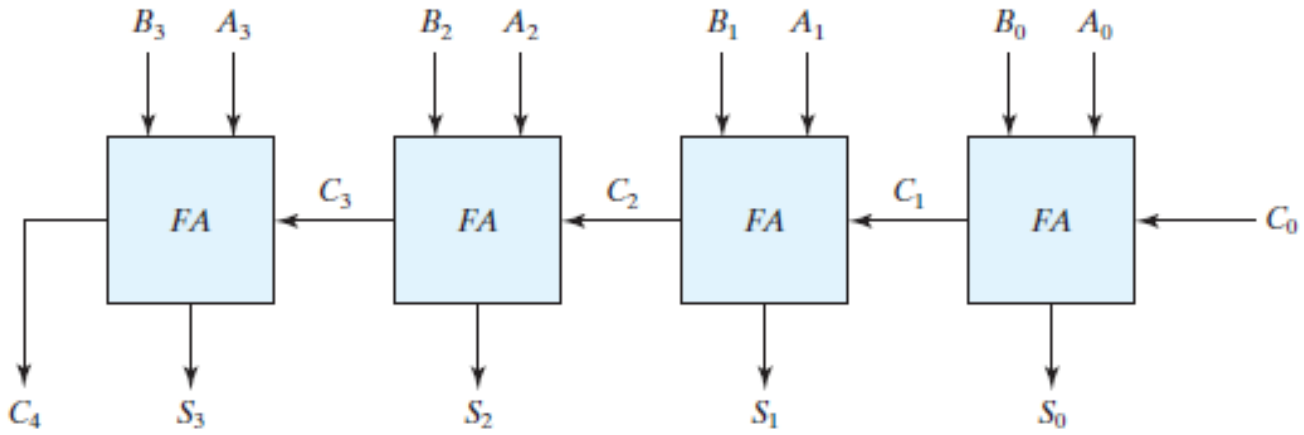
$$= z'(xy' + x'y) + z(xy + x'y')$$

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

$$S = z \oplus (x \oplus y)$$

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

Four bit binary adder



- Also known as ripple carry adder.
- Large combinational/propagation delay
  - 2n for normal design for n bits addition
  - 2n for area optimized design for n bits addition as well

Carry Look ahead adder

- G_i is called a *carry generate*, and it produces a carry of 1 when both A_i and B_i are 1, regardless of the input carry C_i.

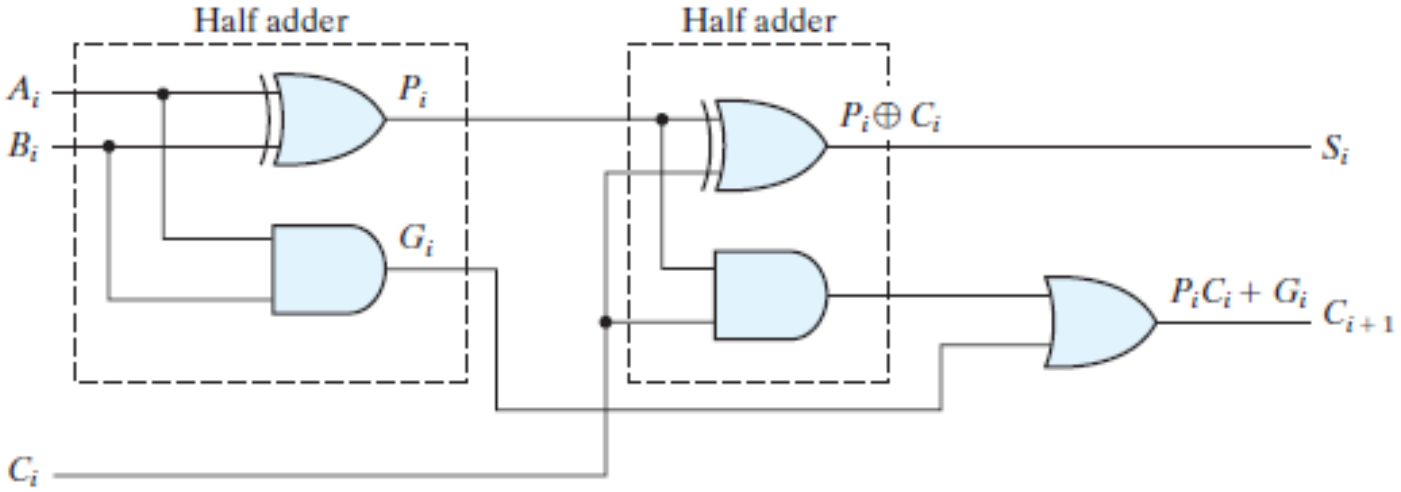


Table 4.4
Full Adder

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

Carry Look ahead adder

- P_i is called a *carry propagate* because it determines whether a carry into stage *i* will propagate into stage *i* + 1 (i.e., whether an assertion of C_i will propagate to an assertion of C_{i+1})

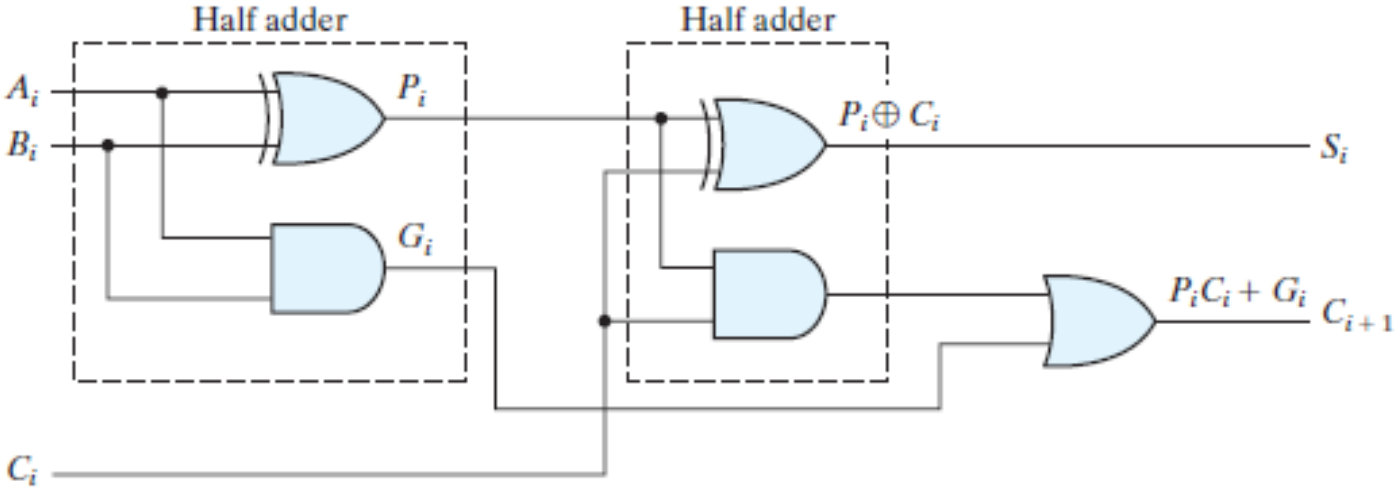


Table 4.4
Full Adder

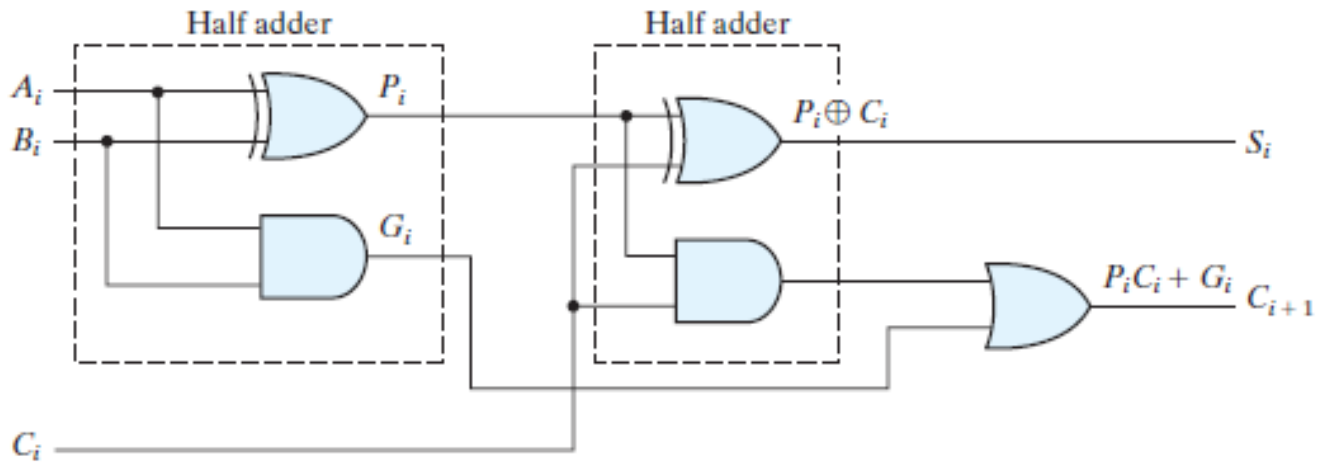
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Carry Look ahead adder

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

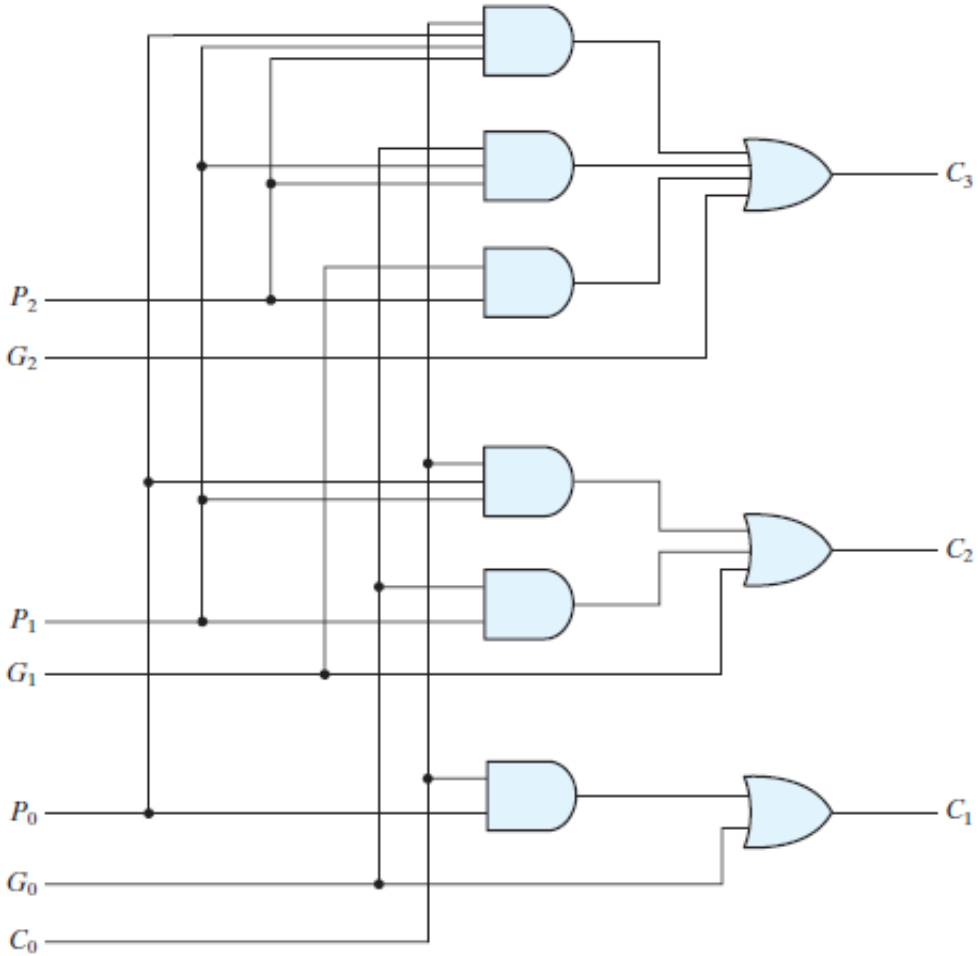
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

C_0 = input carry
$$C_1 = G_0 + P_0 C_0$$



$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

Carry Look ahead adder



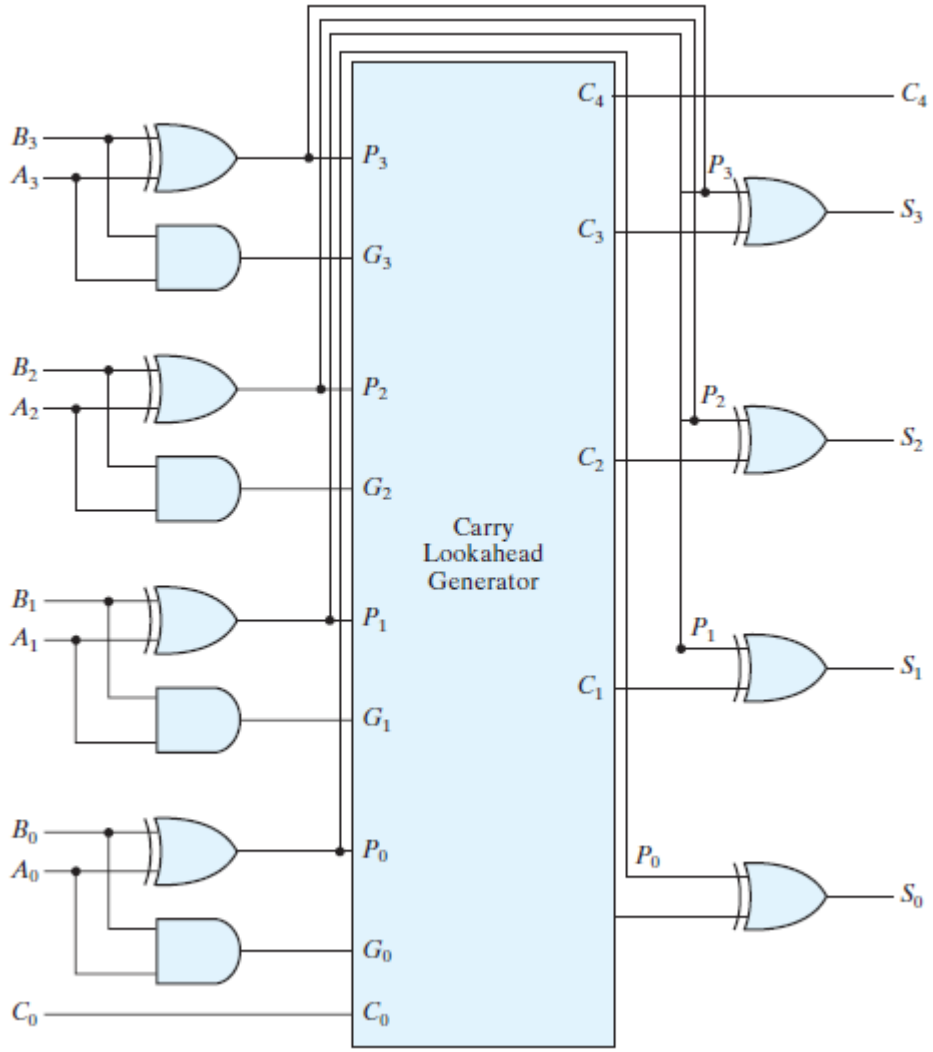
C_0 = input carry

$C_1 = G_0 + P_0C_0$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

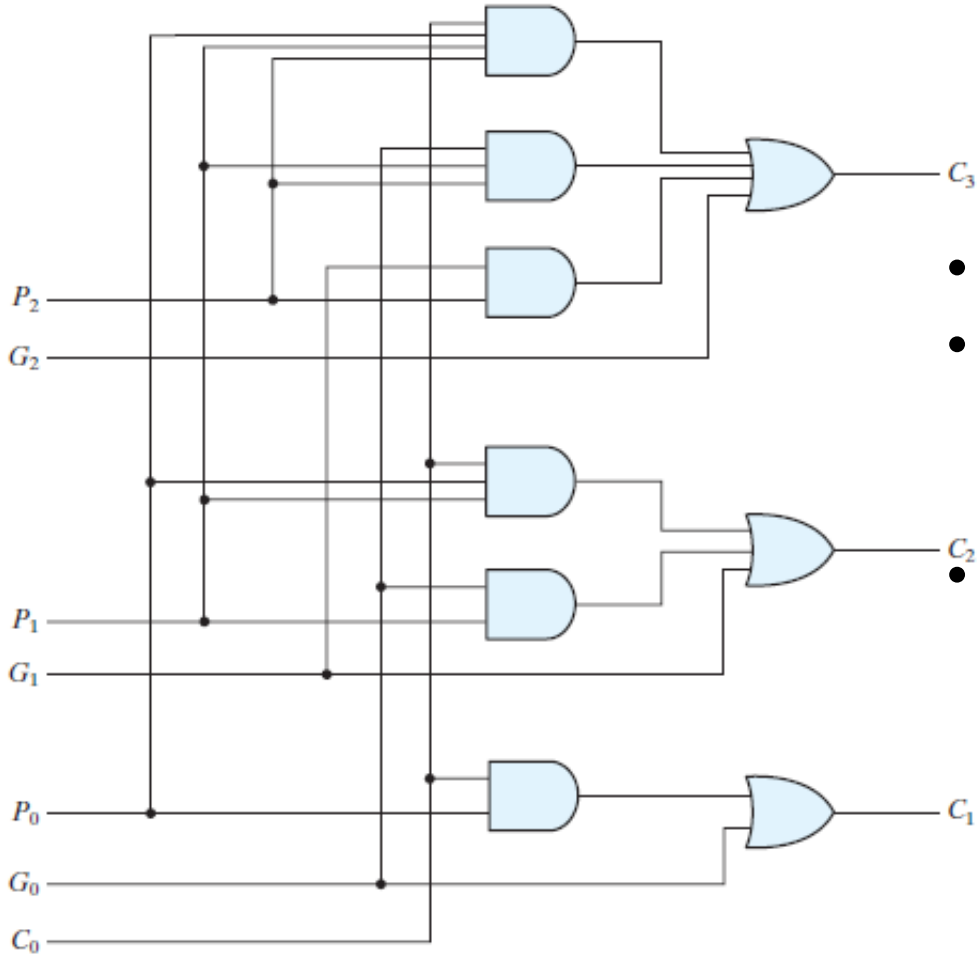
Four bit adder with carry Look ahead adder



What is the delay of a n-bit adder?

Constant 4

Area overhear of Carry look ahead adder



$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

- It requires a very large number of gates.
- For each stage of the adder it is necessary to have an OR gate with *n* inputs and *n* AND gates with 1 through *n* inputs
- Modern day computer is 64-bit word

Carry look ahead adder

- The limitation can be overcome, though at the expense of computation speed.
- By dividing the *n* stages of the adder into groups such that within each group a full carry look ahead is achieved while a ripple carry is maintained between groups.
- For group of size k-stages, delay:

  4 + (2n/k)
- Area overhead: