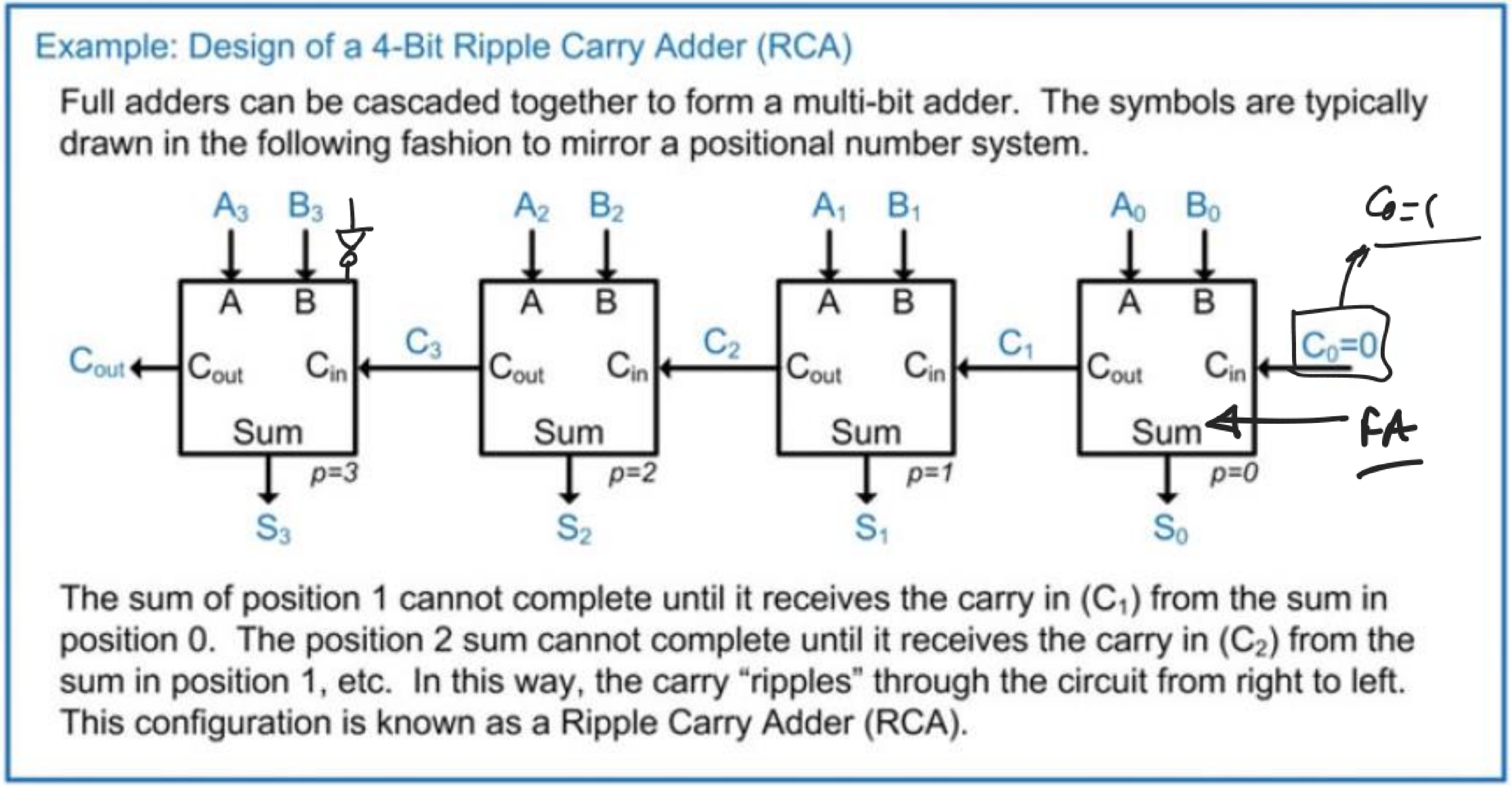
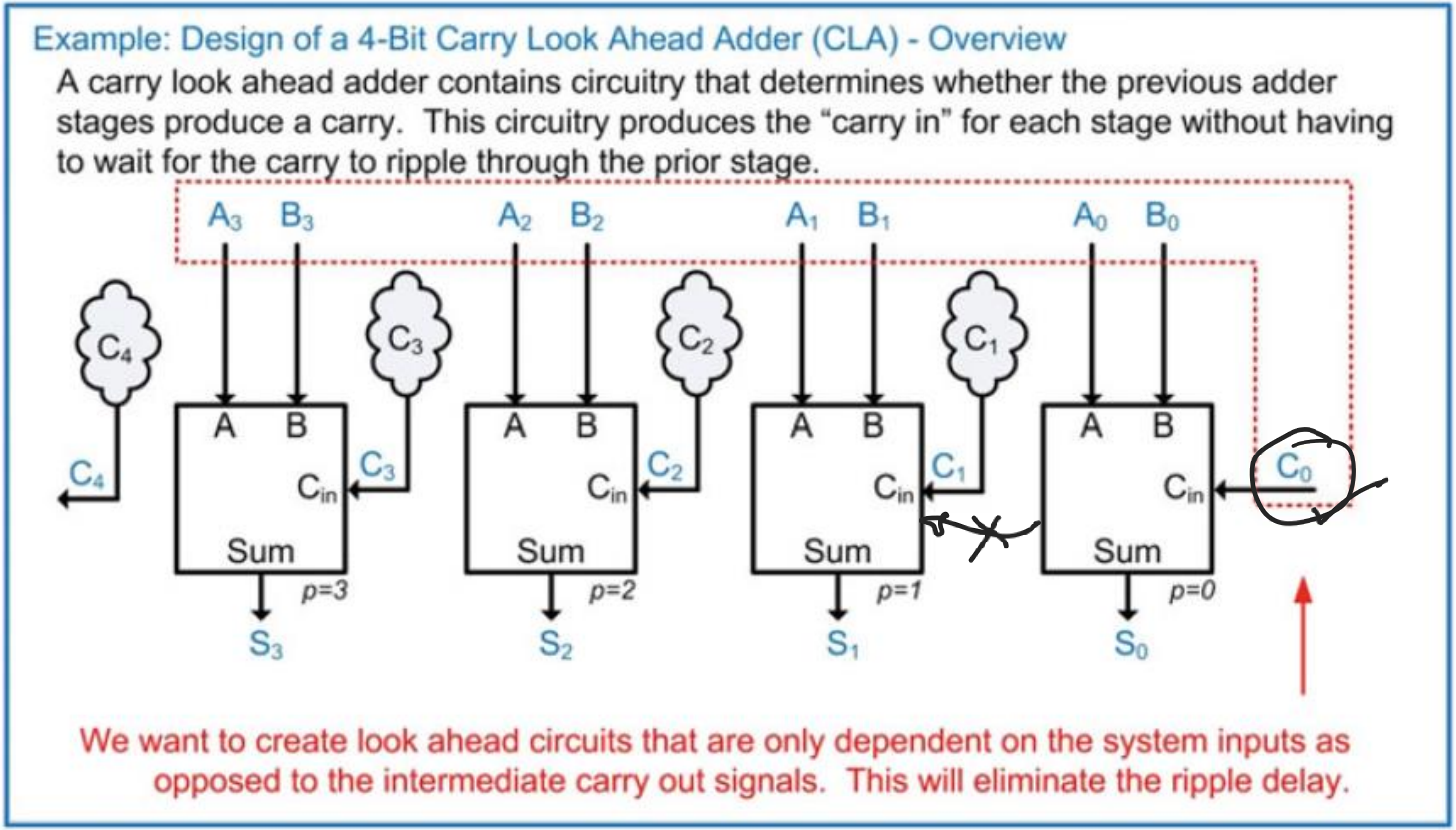
A - B



Example: Design of a 4-Bit Ripple Carry Adder (RCA)

Full adders can be cascaded together to form a multi-bit adder. The symbols are typically drawn in the following fashion to mirror a positional number system.

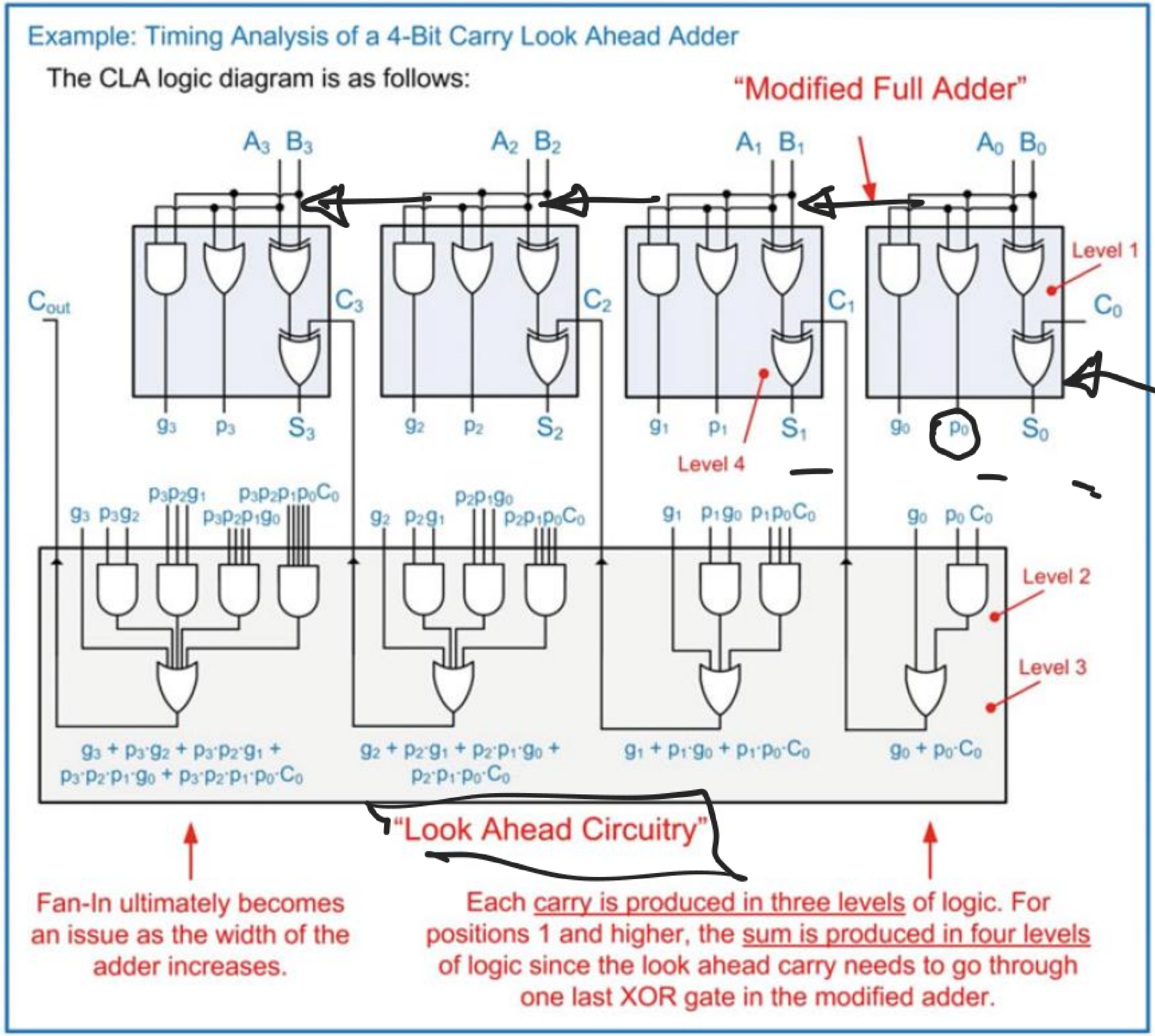
The sum of position 1 cannot complete until it receives the carry in (C_1) from the sum in position 0. The position 2 sum cannot complete until it receives the carry in (C_2) from the sum in position 1, etc. In this way, the carry "ripples" through the circuit from right to left. This configuration is known as a Ripple Carry Adder (RCA).



Example: Design of a 4-Bit Carry Look Ahead Adder (CLA) - Overview

A carry look ahead adder contains circuitry that determines whether the previous adder stages produce a carry. This circuitry produces the "carry in" for each stage without having to wait for the carry to ripple through the prior stage.

We want to create look ahead circuits that are only dependent on the system inputs as opposed to the intermediate carry out signals. This will eliminate the ripple delay.



Example: Timing Analysis of a 4-Bit Carry Look Ahead Adder

The CLA logic diagram is as follows:

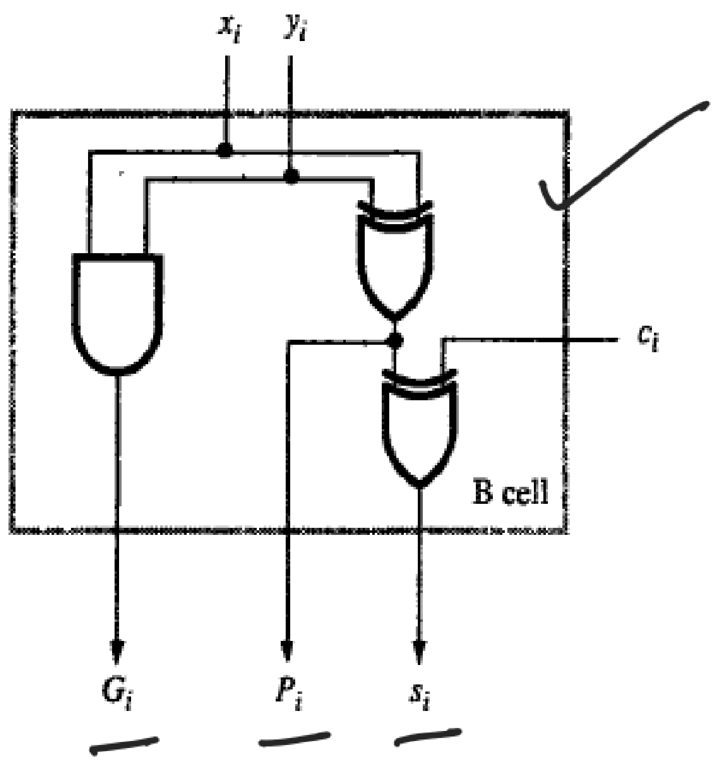
"Modified Full Adder"

A_3 B_3
A_2 B_2
A_1 B_1
A_0 B_0
Level 1
C_{out}
C_3
C_2
C_1
C_0
g_3 p_3 S_3
g_2 p_2 S_2
g_1 p_1 S_1
g_0 p_0 S_0
Level 4
g_3 p_3g_2 $p_3p_2g_1$ $p_3p_2p_1g_0$ $p_3p_2p_1p_0C_0$
g_2 p_2g_1 $p_2p_1g_0$ $p_2p_1p_0C_0$
g_1 p_1g_0 $p_1p_0C_0$
g_0 p_0 C_0
Level 2
Level 3
$g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
$g_0 + p_0 \cdot C_0$

"Look Ahead Circuitry"

Fan-In ultimately becomes an issue as the width of the adder increases.

Each carry is produced in three levels of logic. For positions 1 and higher, the sum is produced in four levels of logic since the look ahead carry needs to go through one last XOR gate in the modified adder.



t=0 x_i, y_i, c_0

4-bit CLA based adder

1τ ✓ $\boxed{P_i = x_i \oplus y_i}$

✓ $\boxed{G_i = x_i y_i}$

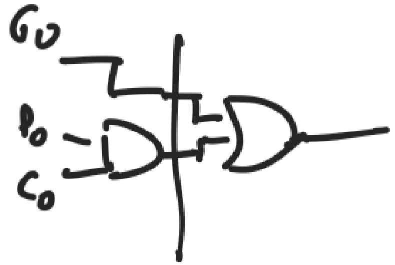
2τ

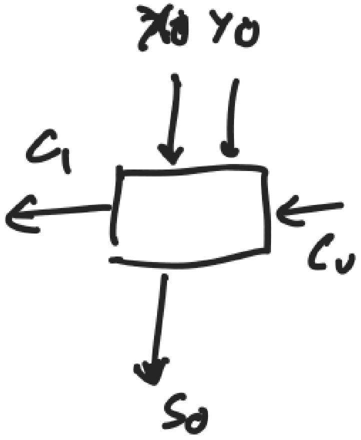
$$c_1 = G_0 + P_0 c_0$$

$$c_2 = G_1 + P_1 G_0 + P_1 P_0 c_0$$

$$c_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0$$

$$c_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$$





$i=1,2,3,4$

c_is

$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \cdots + P_i P_{i-1} \cdots P_1 G_0 + P_i P_{i-1} \cdots P_0 c_0$$

3τ

1τ

$$\boxed{S_i = P_i \oplus C_i}$$

Delay Analysis of CLA based adder

Logic Design of a subtractor

A = 0 → addition

A = 1 → subtraction.

$\boxed{\overline{A}/S}$

$\boxed{F = A \oplus B}$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Signed representation.

2's complement

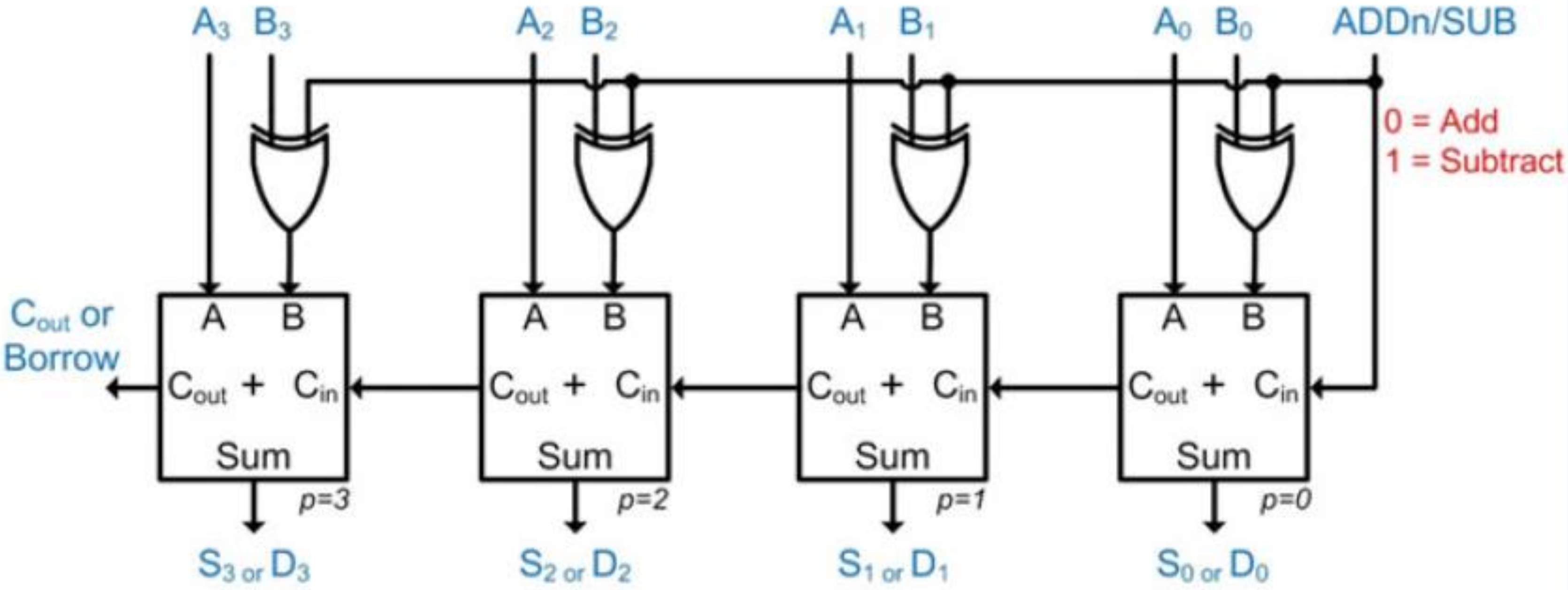
$\underline{A} - \underline{B}$

$$\begin{array}{r} A \\ \underline{-B} \end{array} \qquad \begin{array}{r} A \\ \underline{+(-B)} \end{array}$$

Making the adder/subtractor programmable

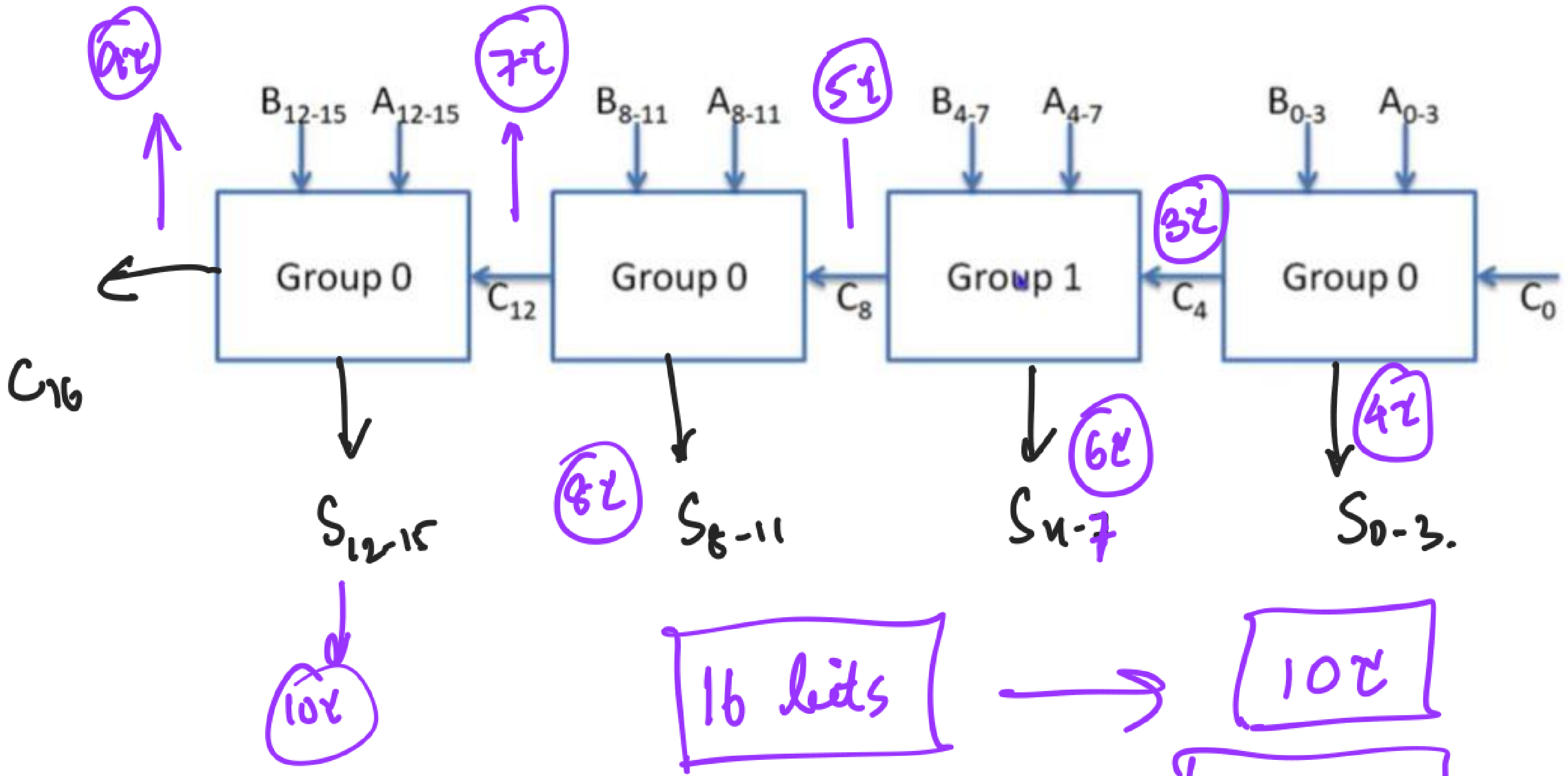
Example: Design of a 4-Bit Programmable Adder/Subtractor

The control signal “ADDn/SUB” is used to select whether the circuit performs addition (ADDn/SUB=0) or subtraction (ADDn/SUB=1). When in subtraction mode, the XOR gates invert the subtrahend B and add 1 to the first adder stage. These steps take the two’s complement of B and allow an add operation to conduct subtraction.



16 bit CLA adder

motivation : fan-in restriction.



16 bits → 10τ

RCA → 32τ

Timing analysis

n-bit adder using 4-bit CLA blocks.

c_1-c_4	3τ
c_5-c_8	5τ
c_9-c_{12}	7τ
$\vdots$	
c_{n-3}-c_n	
Sum.	1τ

$$\boxed{2\times\left\lceil \frac{n}{l} \right\rceil + 2}$$

no. of bits

block size

$$\boxed{\text{Speedup factor} = l}$$

w.r.t RCA

S_i

Sum.

$$\left(\frac{n}{2}+2\right)$$

n bits.

$$1 + 2\left\lceil \frac{n}{4} \right\rceil + 1$$

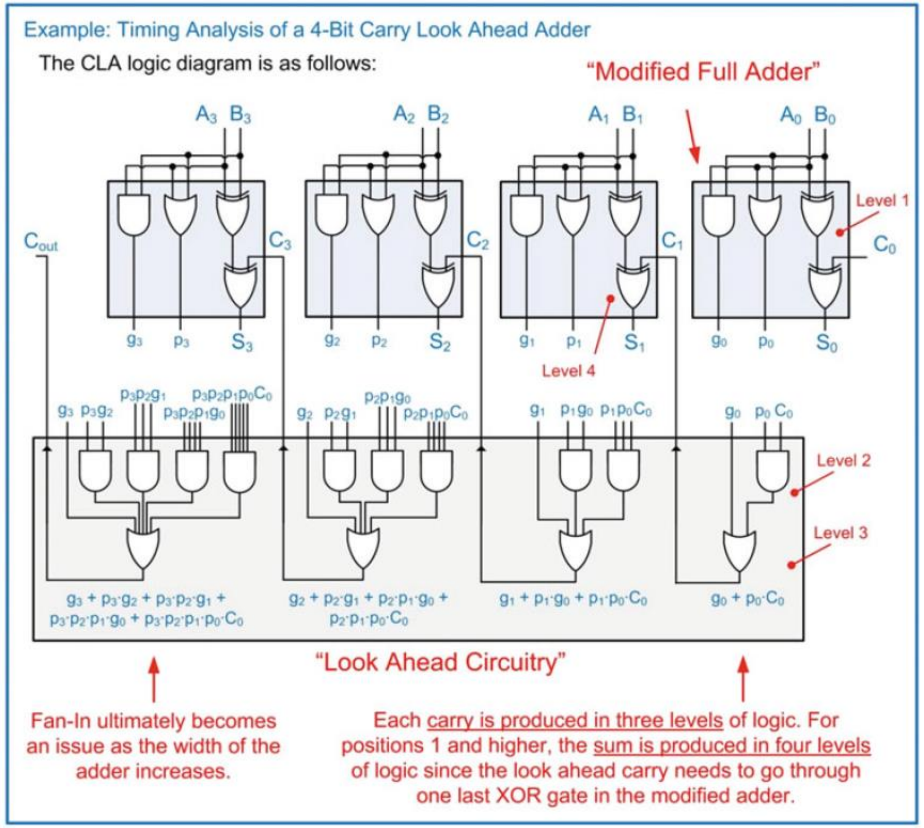
P_i, G_i

$$\text{Speedup ratio} = \frac{2n}{\left(\frac{n}{2}+2\right)}$$

$$\left(2+\frac{n}{2}\right)$$

$$\approx \boxed{4}$$

Revisiting the CLA based adder



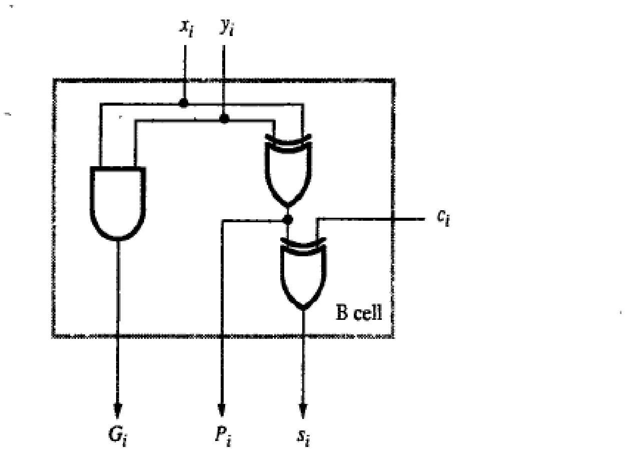
Example: Timing Analysis of a 4-Bit Carry Look Ahead Adder

The CLA logic diagram is as follows:

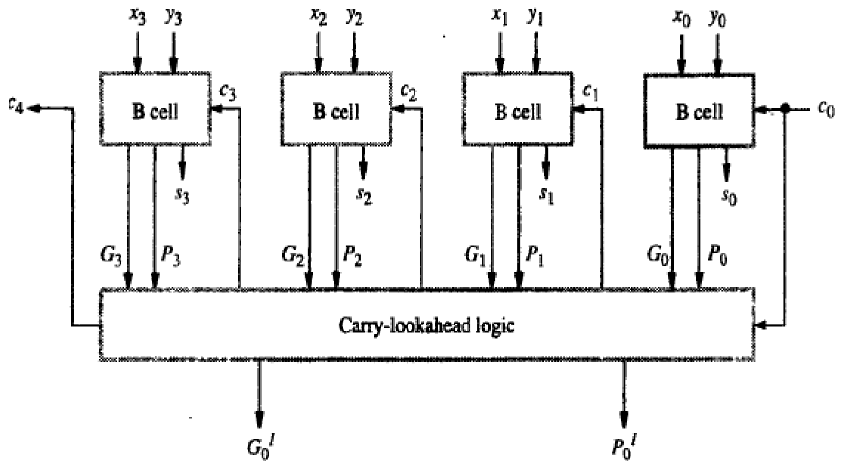
"Modified Full Adder"
A_3 B_3
A_2 B_2
A_1 B_1
A_0 B_0
Level 1
C_{out}
C_3
C_2
C_1
C_0
g_3 p_3 S_3
g_2 p_2 S_2
g_1 p_1 S_1
g_0 p_0 S_0
Level 4
g_3 p_3g_2 $p_3p_2g_1$ $p_3p_2p_1g_0$ $p_3p_2p_1p_0C_0$
g_2 p_2g_1 $p_2p_1g_0$ $p_2p_1p_0C_0$
g_1 p_1g_0 $p_1p_0C_0$
g_0 p_0 C_0
Level 2
Level 3
$g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
$g_0 + p_0 \cdot C_0$
"Look Ahead Circuitry"

Fan-In ultimately becomes an issue as the width of the adder increases.

Each carry is produced in three levels of logic. For positions 1 and higher, the sum is produced in four levels of logic since the look ahead carry needs to go through one last XOR gate in the modified adder.



(a) Bit-stage cell



(b) 4-bit adder

Higher-Level Generate and Propagate Functions

- Problem : Bottleneck due to serial dependency

Acknowledgements for Figures in Slides

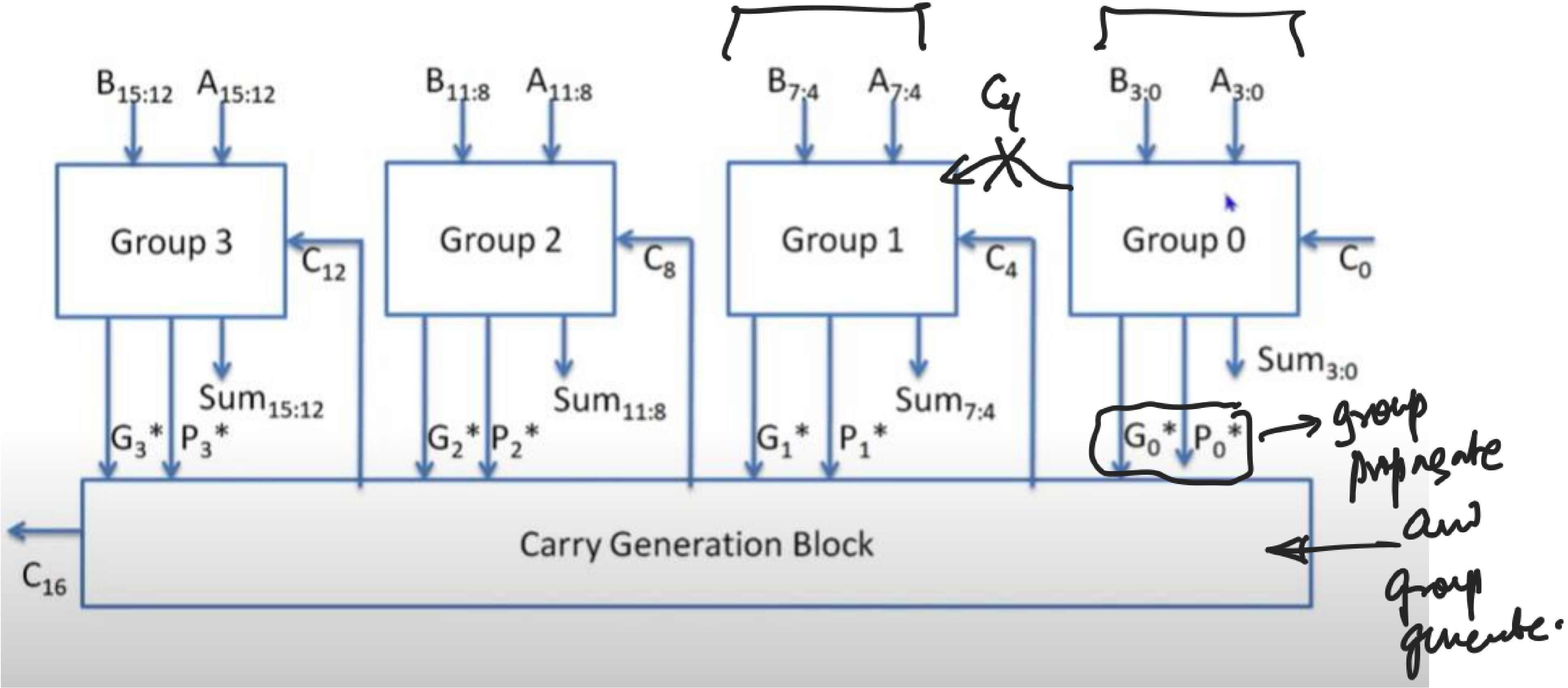
① Textbook by Brock J. LaMeres ["Intro to Logic Circuits & Logic Design" – Springer (2023)]

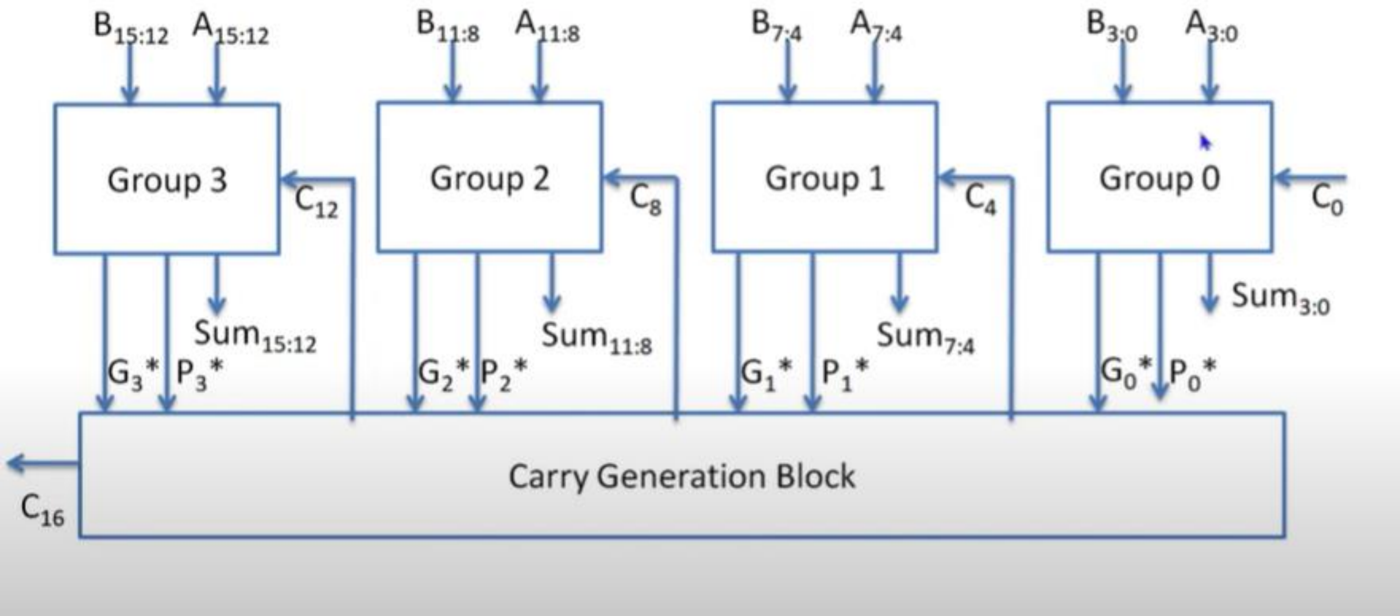
② Textbook by Hamacher et al. ("Computer Organization" – Mc Graw Hill (2002).

③ For the two slides below:

Prof. Neeraj Goel

Two level CLA





$$c_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0$$

- Use group carry and group propagate
  - $G^* = G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3$ ✓
  - $P^* = P_0P_1P_2P_3$ ✓
  - $\boxed{C_4 = G^*_0 + C_0P^*_0}$
  - $C_8 = G^*_1 + G^*_0P^*_1 + C_0P^*_0P^*_1$
  - $C_{12} = G^*_2 + G^*_1P^*_2 + G^*_0P^*_2P^*_1 + C_0P^*_2P^*_1P^*_0$

$\boxed{C_1 = P_0 C_0 + G_0}$