Carry Lookahead Adder (2A)

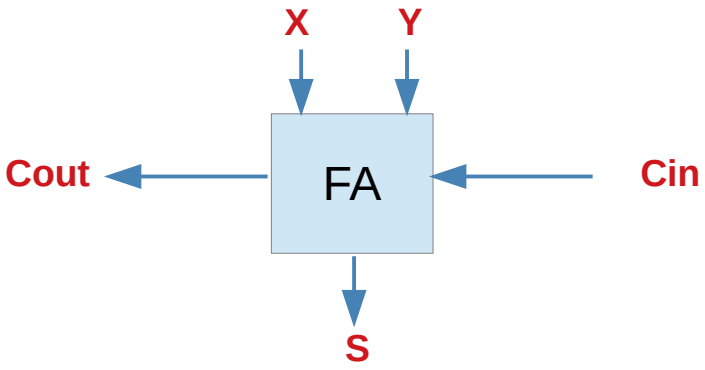
-
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Carry Kill, Propagate, Generate conditions (1)

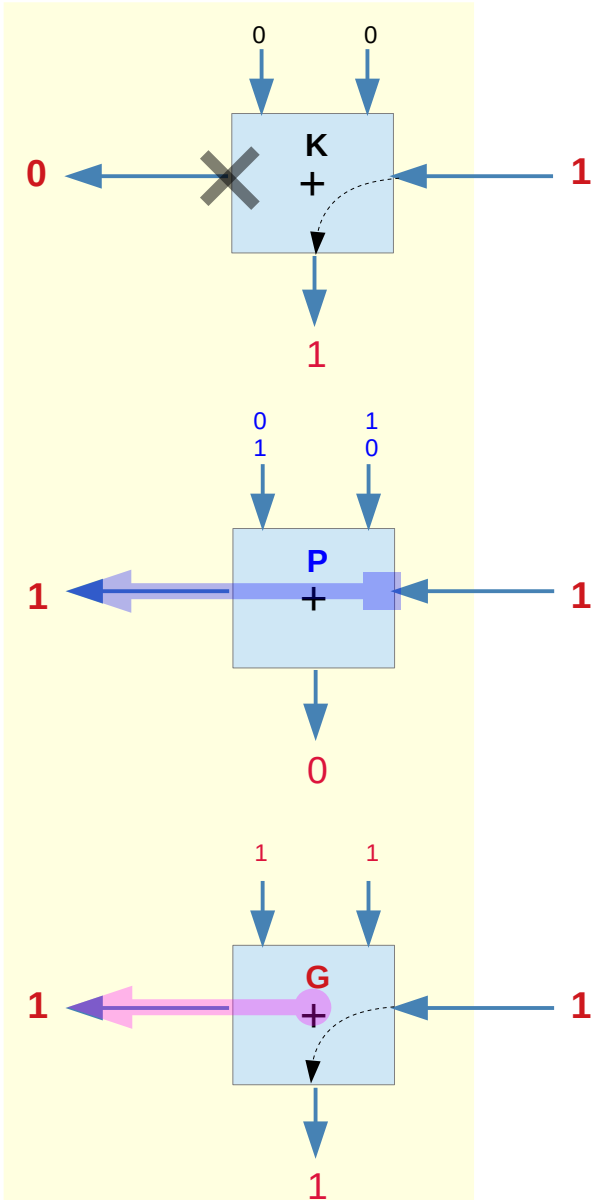
X	Y		
0	0	K	Kill (=$\overline{P}\overline{G}$)
0	1	P	Propagate
1	0	P	Propagate
1	1	G	Generate



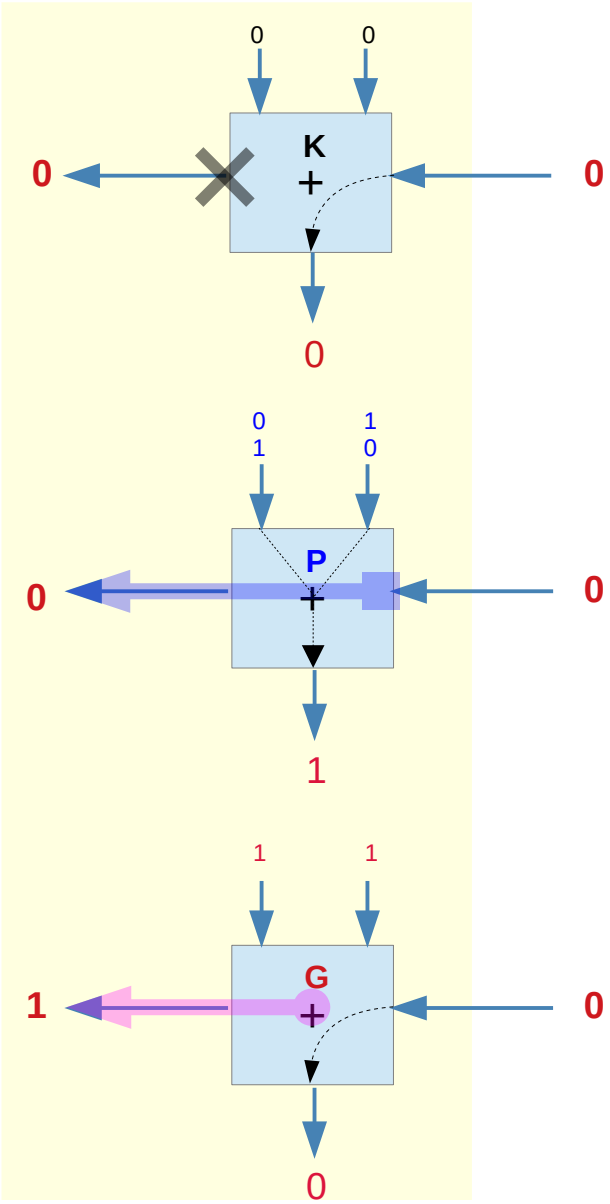
https::/electronics.stackexchange.com/questions/21251/critical-path-for-carry-skip-adder

Carry Kill, Propagate, Generate conditions (2)

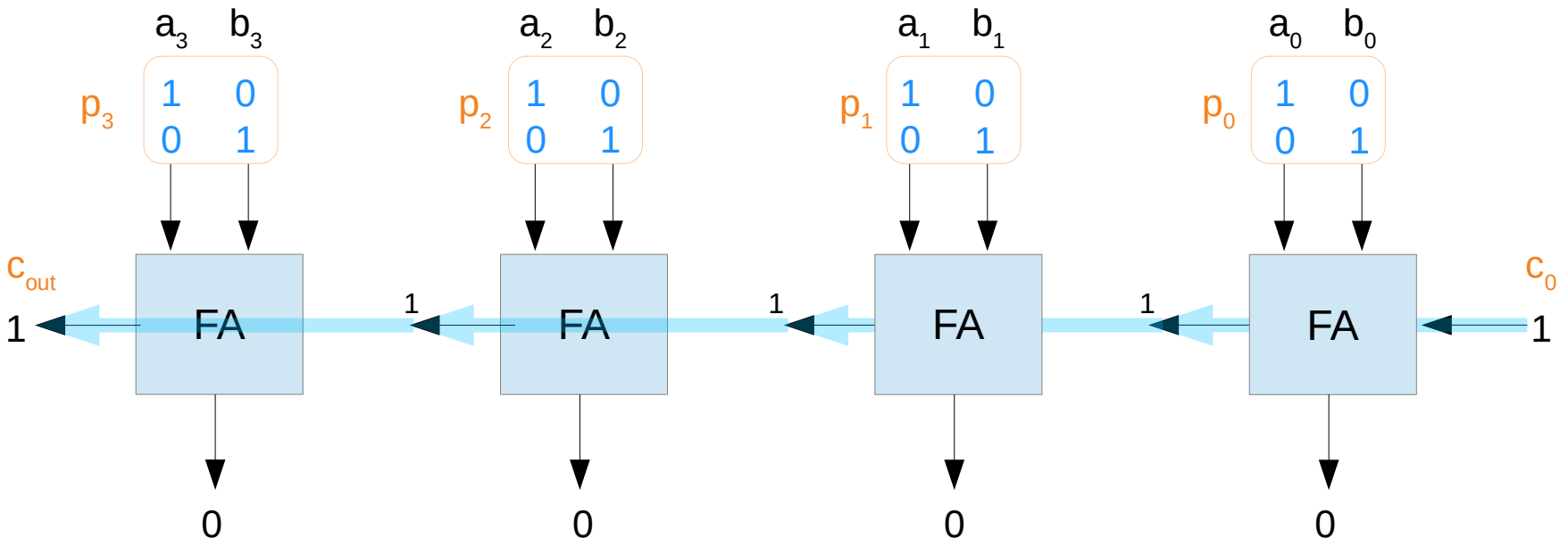
killed carry



killed carry



C_0 propagation condition

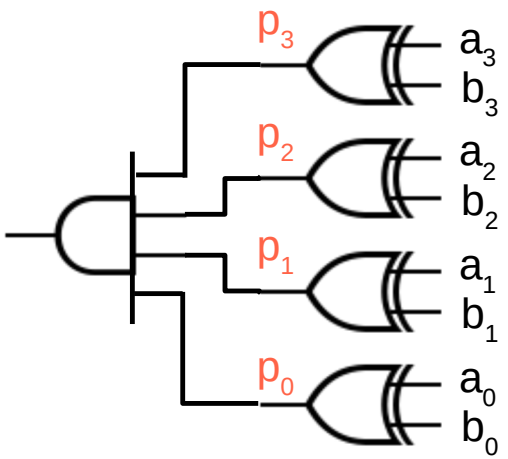


$s = p_3 \wedge p_2 \wedge p_1 \wedge p_0$

$= (a_3 \oplus b_3) \wedge (a_2 \oplus b_2) \wedge (a_1 \oplus b_1) \wedge (a_0 \oplus b_0)$

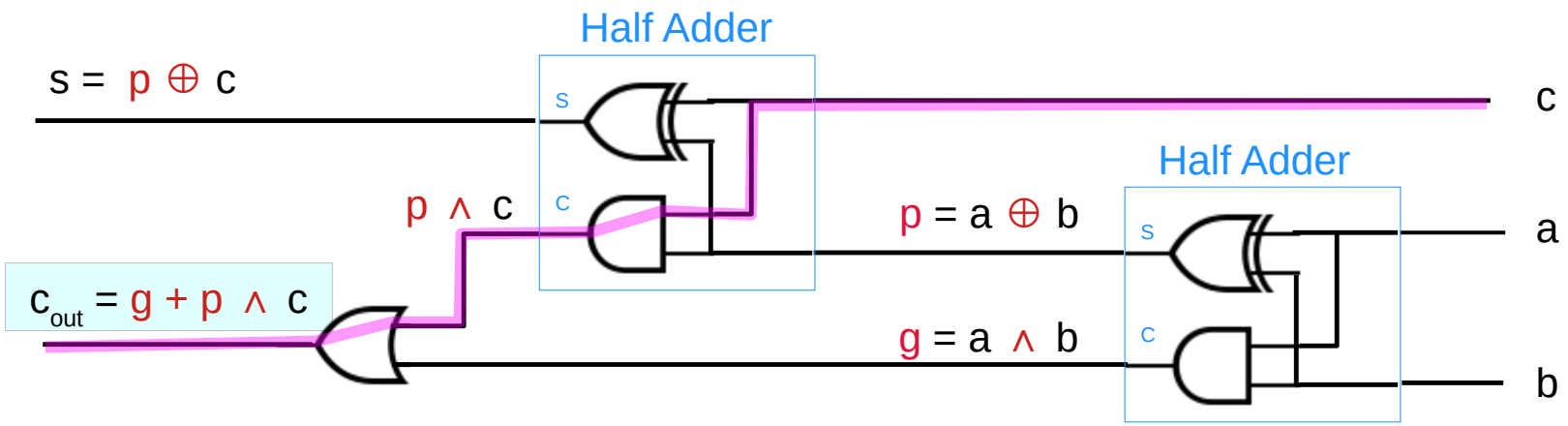
c_0 can be propagated to c_{out} only if $s = 1$

$s = p_3 p_2 p_1 p_0$



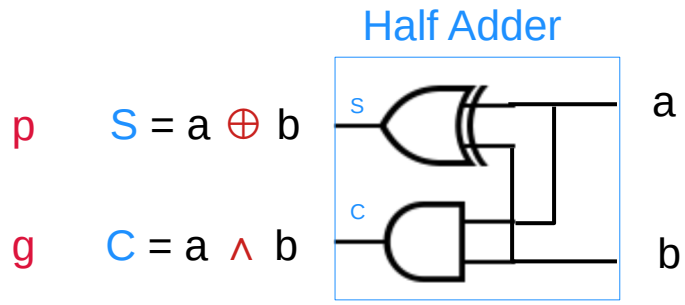
https://en.wikipedia.org/wiki/Carry-skip_adder

A FA in terms of **p** and **g**



Half Adder
S = a ⊕ b
C = a ∧ b

a	b	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



https://en.wikipedia.org/wiki/Carry-skip_adder

Full adder with additional generate and propagate signals.

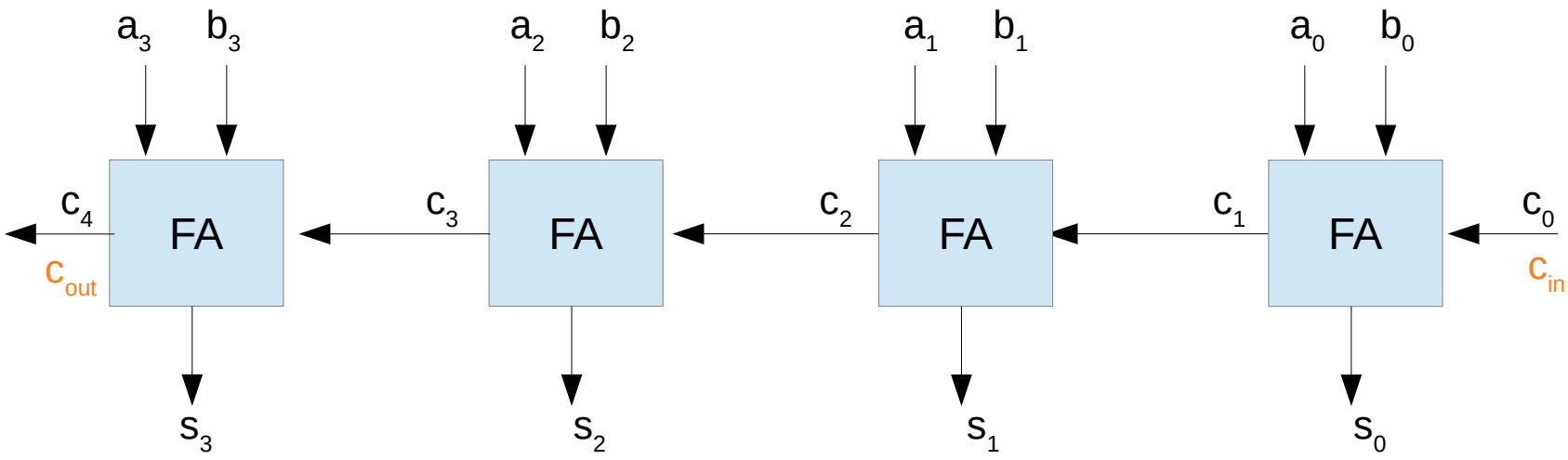
Carry Equations in a Ripple Carry Adder

$$c_1 = g_0 + p_0 \wedge c_0$$
$$c_2 = g_1 + p_1 \wedge c_1$$
$$c_3 = g_2 + p_2 \wedge c_2$$
$$c_4 = g_3 + p_3 \wedge c_3$$

generated carry

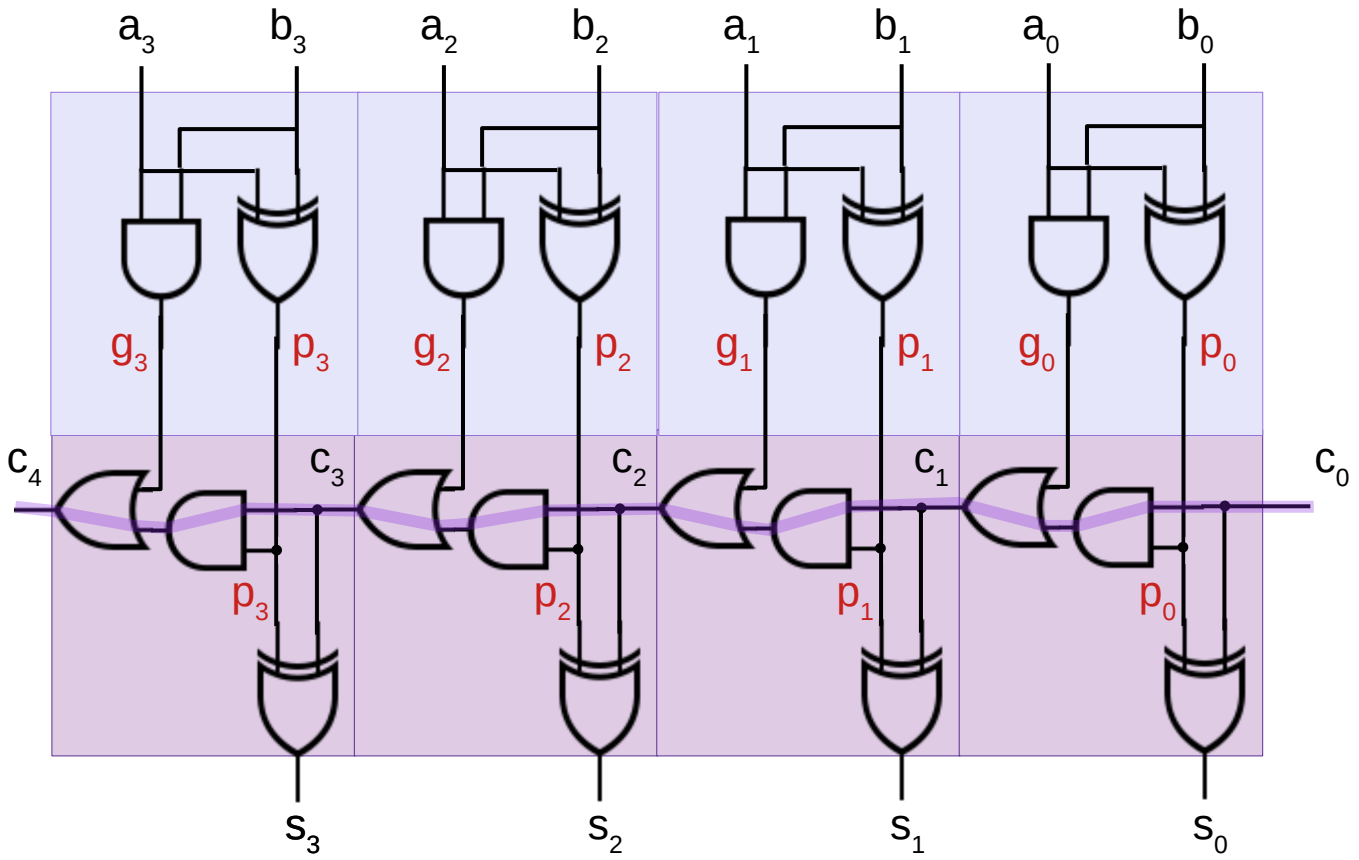
propagated carry

$$p_i = a_i \oplus b_i$$
$$g_i = a_i \wedge b_i$$



https://en.wikipedia.org/wiki/Carry-skip_adder

A 4-bit Ripple Carry Adder



Half Adder 1

$p_i = a_i \oplus b_i$

$g_i = a_i \wedge b_i$

Half Adder 2

$c_{i+1} = g_i + p_i \wedge c_i$

$s_i = p_i \oplus c_i$

Critical Path : $c_4 - c_3 - c_2 - c_1 - c_0$: 8 gate delay

https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

Expanding carry equations

$$c_{i+1} = g_i + p_i c_i$$

$$c_1 = g_0 + p_0 c_0 \quad \Rightarrow \quad c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 c_1 \quad \Rightarrow \quad c_2 = g_1 + p_1 \left[g_0 + p_0 c_0 \right]$$

$$c_3 = g_2 + p_2 c_2 \quad \Rightarrow \quad c_3 = g_2 + p_2 \left[g_1 + p_1 \left[g_0 + p_0 c_0 \right] \right]$$

$$c_4 = g_3 + p_3 c_3 \quad \Rightarrow \quad c_4 = g_3 + p_3 \left[g_2 + p_2 \left[g_1 + p_1 \left[g_0 + p_0 c_0 \right] \right] \right]$$

c_{out} c_{in}

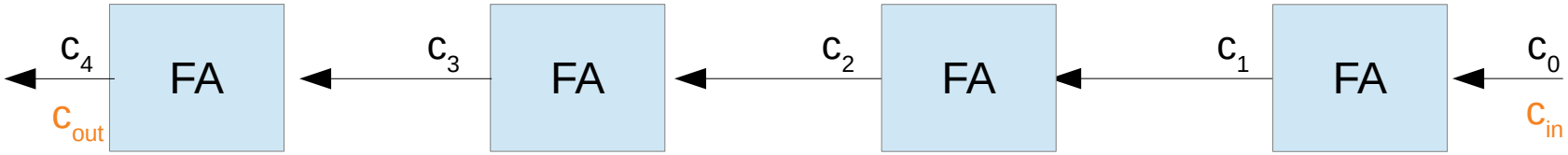
$$g_0 + p_0 c_0 = c_1$$

$$g_1 + p_1 g_0 + p_1 p_0 c_0 = c_2$$

$$g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 = c_3$$

$$g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0 = c_4$$

Carry Generate and Carry Propagate Signals



these carries can be computed in parallel

c_1

g_0 generated g_0

$p_0 c_0$ propagated c_0

c_2

g_1 generated g_1

$p_1 g_0$ propagated generated g_0

$p_1 p_0 c_0$ propagated propagated c_0

c_3

g_2 generated g_2

$p_2 g_1$ propagated generated g_1

$p_2 p_1 g_0$ propagated propagated generated g_0

$p_2 p_1 p_0 c_0$ propagated propagated propagated c_0

c_4

g_3 generated g_3

$p_3 g_2$ propagated generated g_2

$p_3 p_2 g_1$ propagated propagated generated g_1

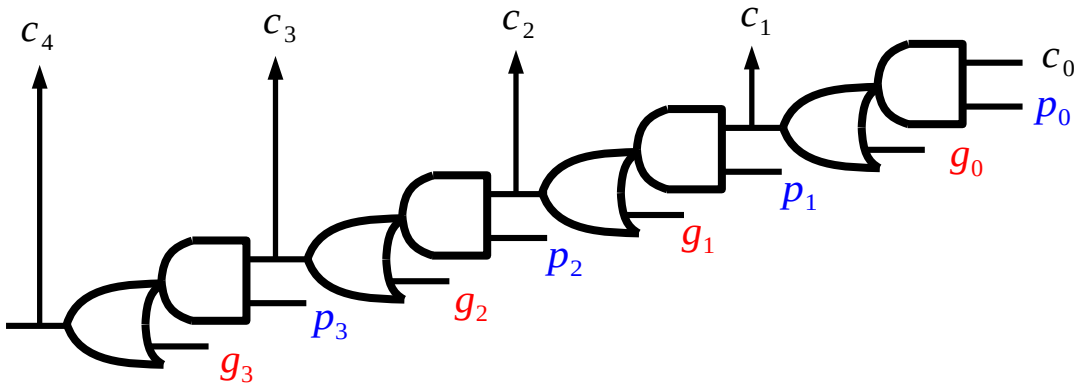
$p_3 p_2 p_1 g_0$ propagated propagated propagated generated g_0

$p_3 p_2 p_1 p_0 c_0$ propagated propagated propagated propagated c_0

Carry equations into Gates (1)

$$c_{i+1} = g_i + p_i c_i$$

$$c_4 = g_3 + p_3\left[g_2 + p_2\left[g_1 + p_1\left[g_0 + p_0 c_0\right]\right]\right]$$

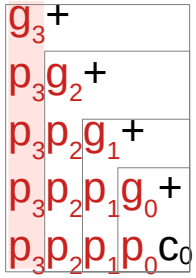


8 gate delay

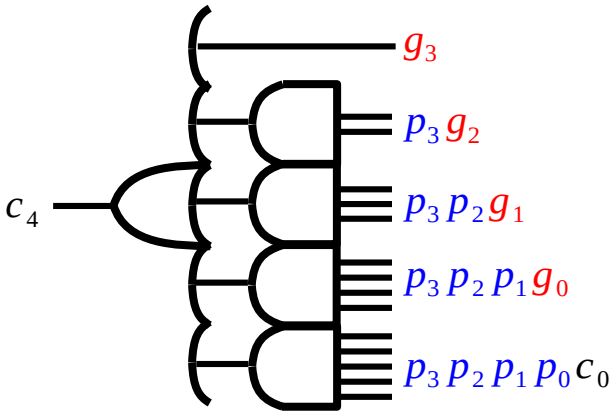
Fan-in number: small
Stage number: large

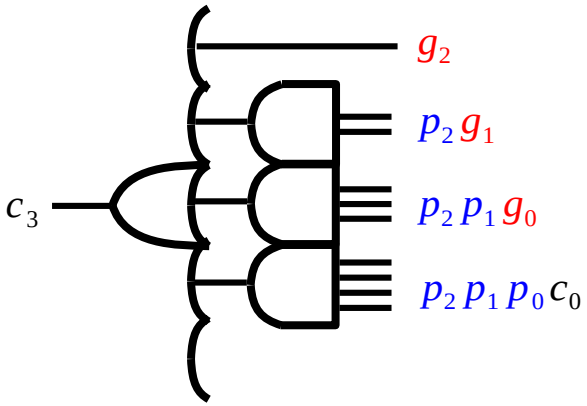
Ripple Carry Adder

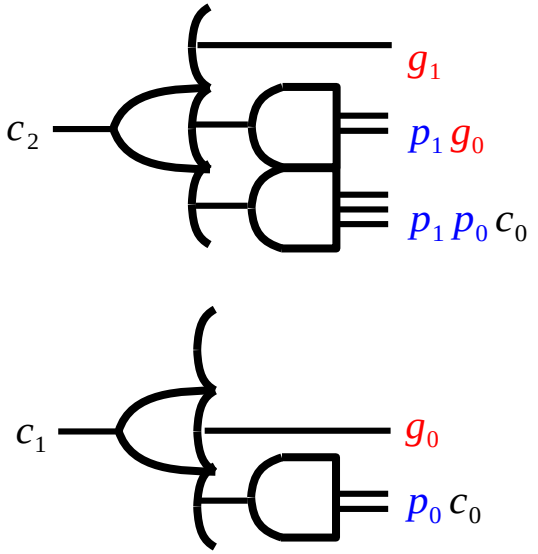
Carry equations into Gates (2)



$$g_0 + p_0 c_0 = c_1$$
$$g_1 + p_1 g_0 + p_1 p_0 c_0 = c_2$$
$$g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 = c_3$$
$$g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0 = c_4$$





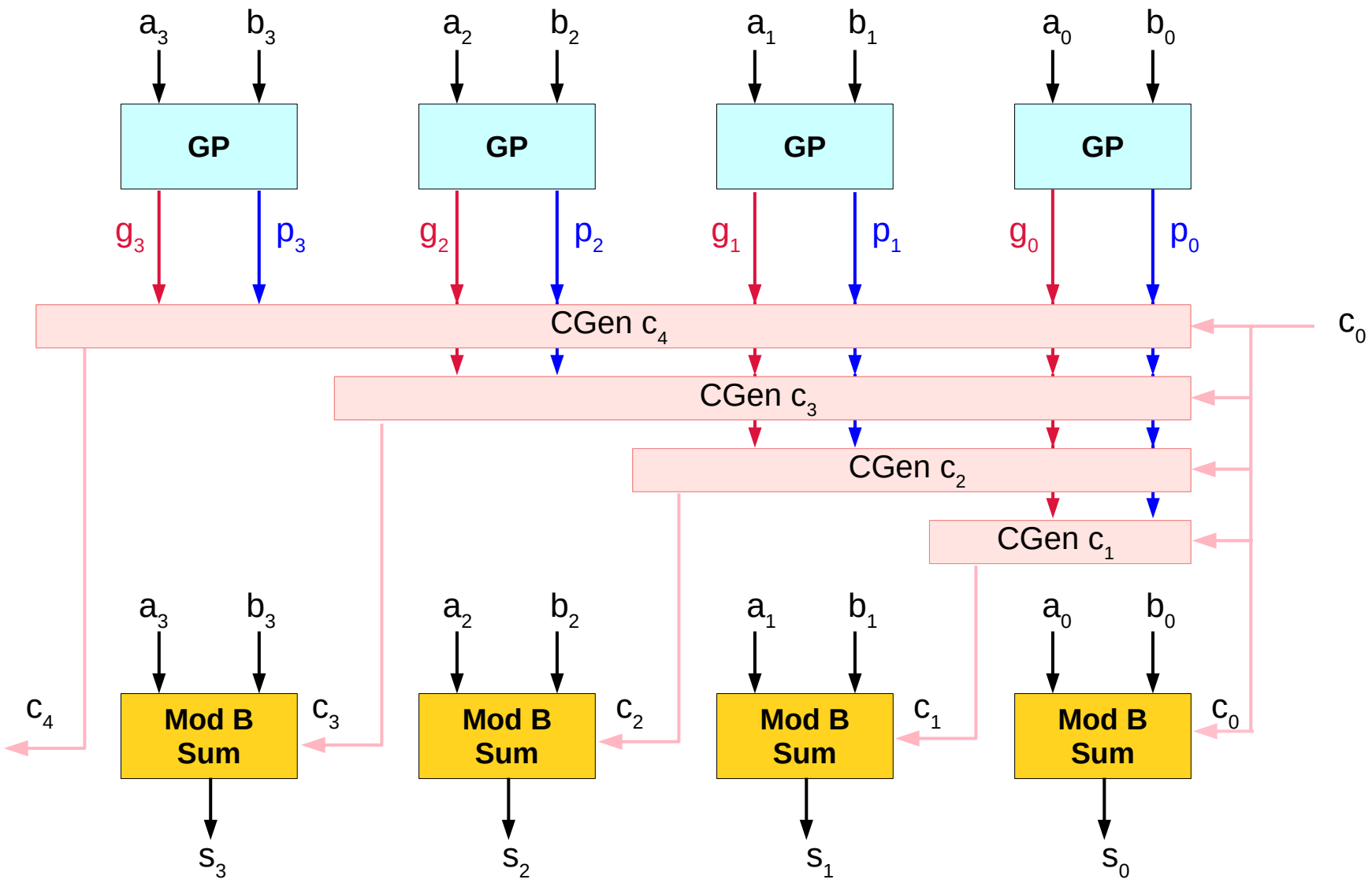


Sum of Product :
2 gate delay

Fan-in number: large
Stage number: small

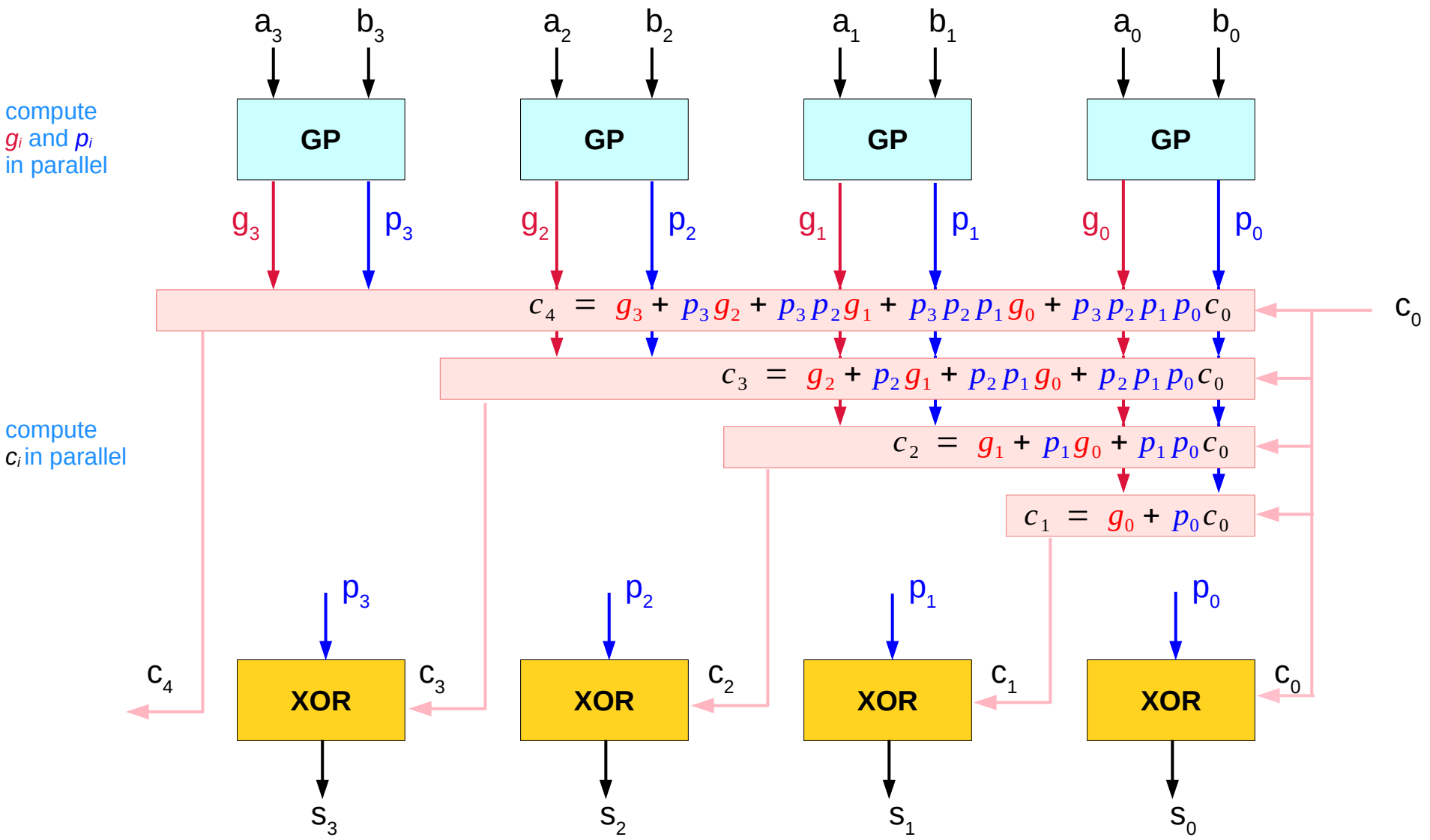
Carry Lookahead Adder

Carry Lookahead Adder



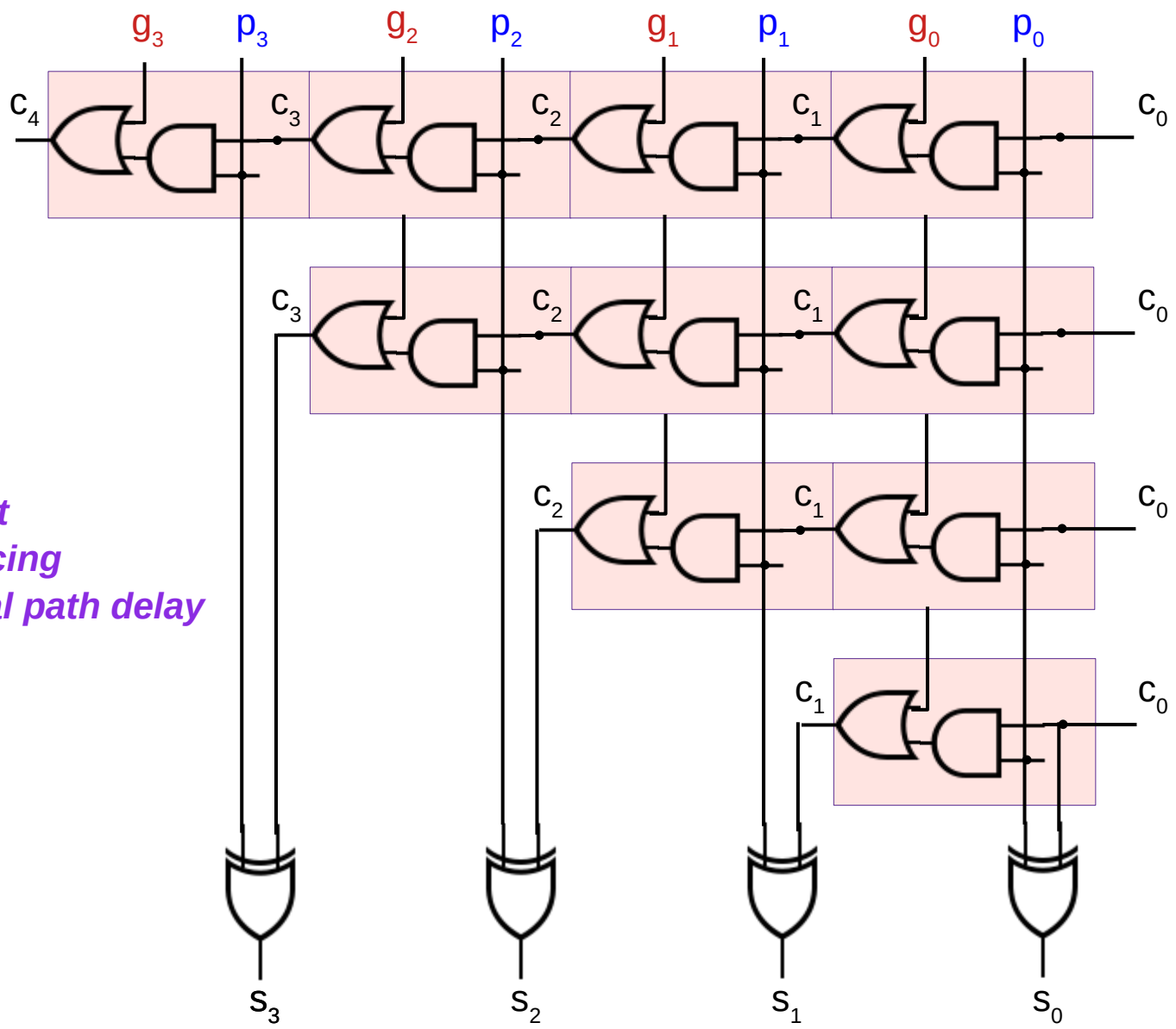
Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Carry Lookahead Adder



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Replicating carry circuits (1)



Half Adder

$p_i = a_i \oplus b_i$

$g_i = a_i \wedge b_i$

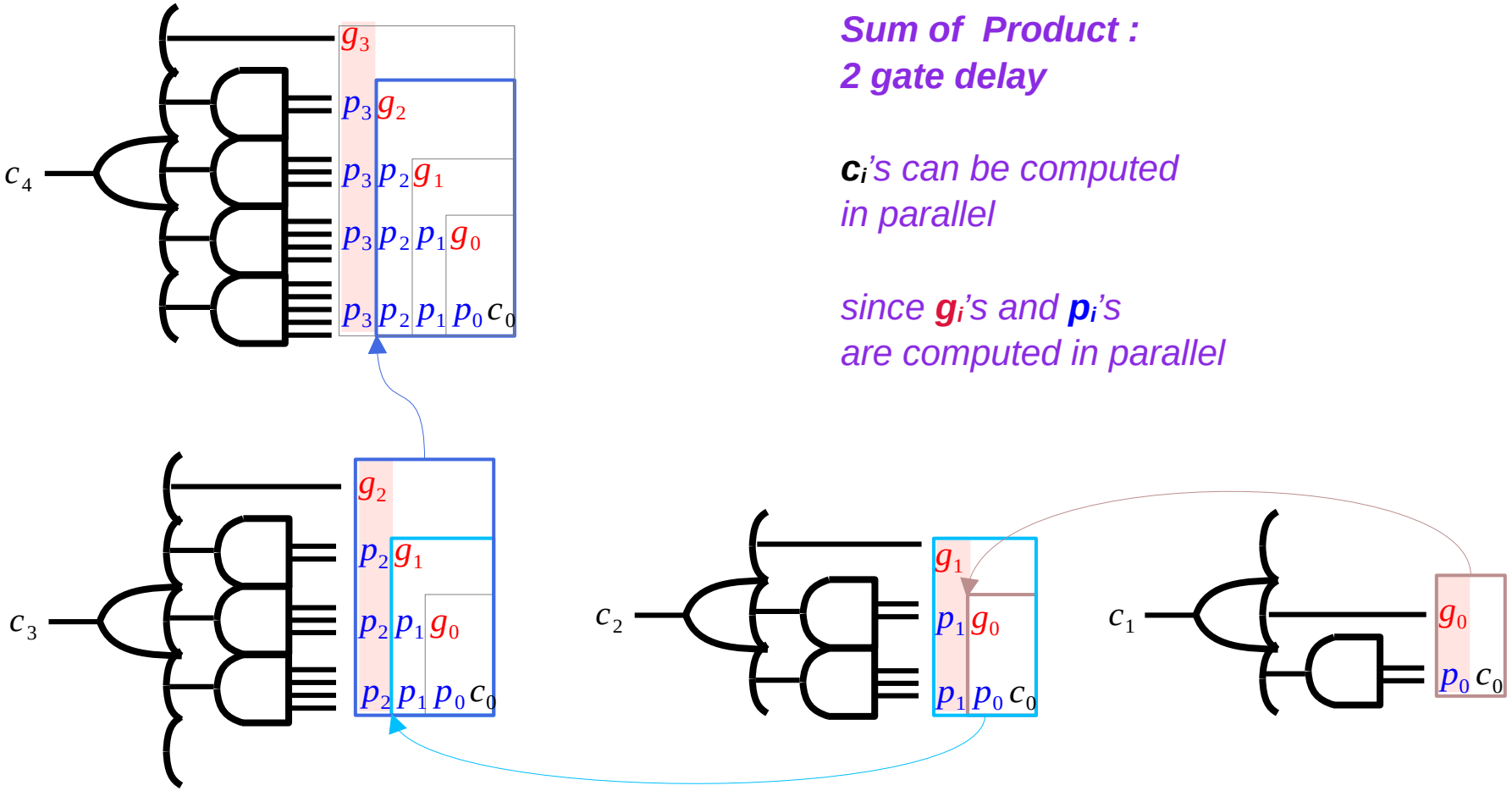
Half Adder

$c_{i+1} = g_i + p_i \wedge c_i$

$s_i = p_i \oplus c_i$

no merit in reducing a critical path delay

Replicating carry circuits (2)



Sum of Product :
2 gate delay

***c_i**'s can be computed*
in parallel

*since **g_i**'s and **p_i**'s*
are computed in parallel

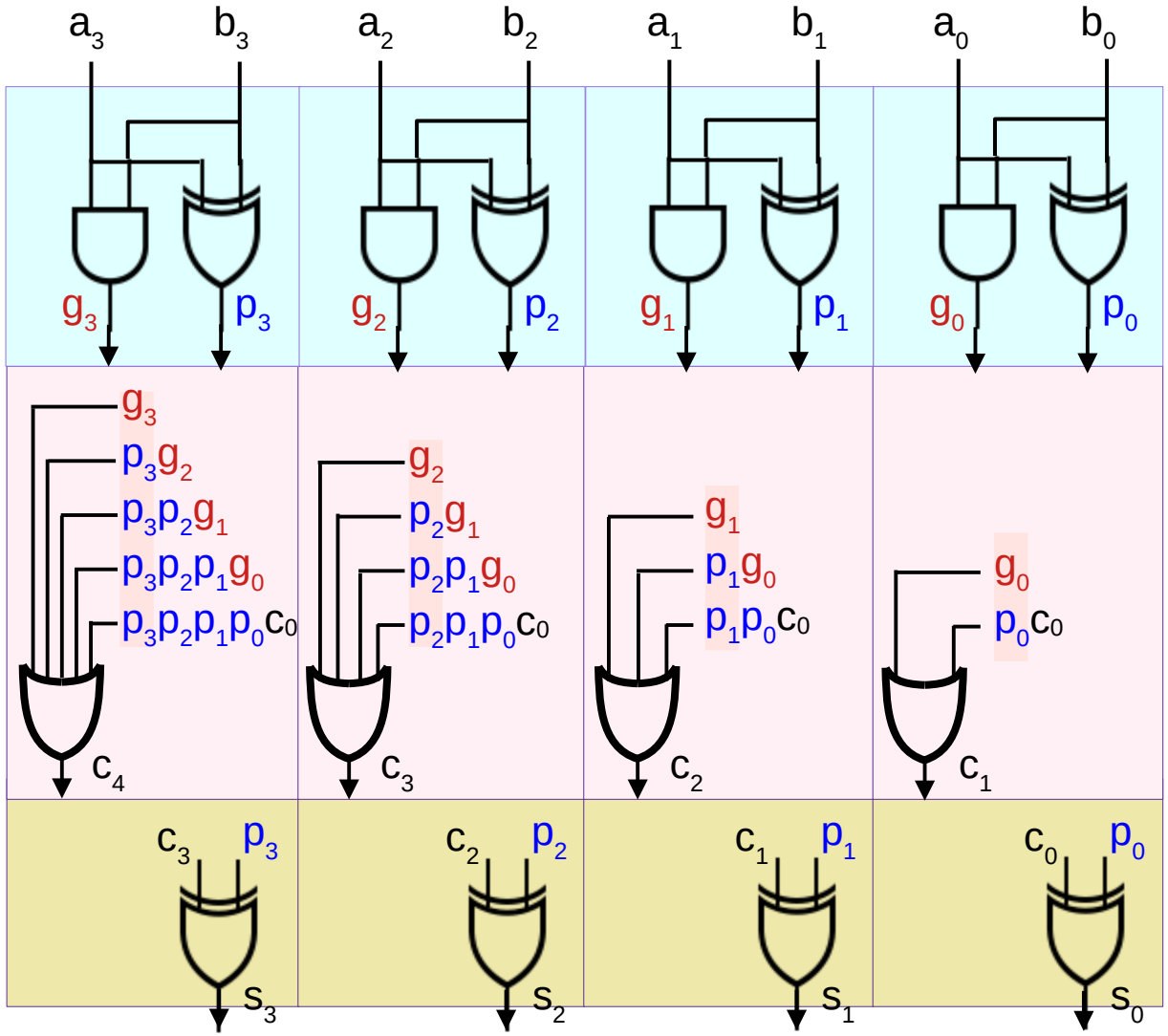
Limitations of Carry Lookahead

AND2, OR2		$g_0 + p_0 c_0 = c_1$
AND3, OR3		$g_1 + p_1 g_0 + p_1 p_0 c_0 = c_2$
AND4, OR4		$g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 = c_3$
AND5, OR5		$g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0 = c_4$
⋮ ⋮		
AND32, OR32	possible?	$= c_{31}$
AND33, OR33	possible?	$= c_{32}$

Large number of fan-in : *impractical*

- High Radix Addition (2^g)
- Multi-level Carry Lookahead

A 4-bit Carry Lookahead Adder



compute g_i and p_i in parallel

compute c_i in parallel

compute S_i in parallel

GP

g_i : 1 gate delay ||
p_i : 1 gate delay

CGen

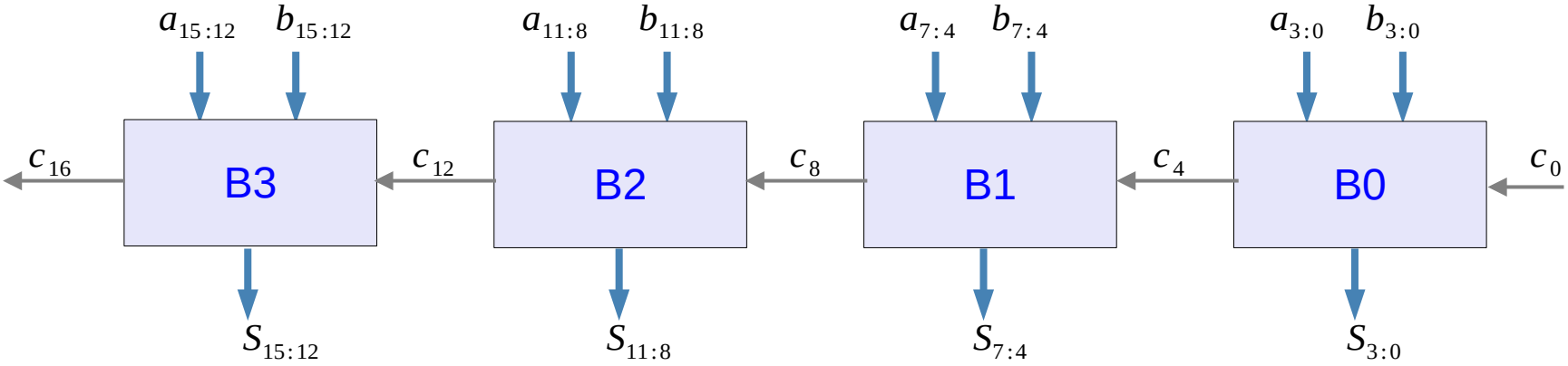
Sum of Product :
2 gate delay

c_i : 2 gate delay

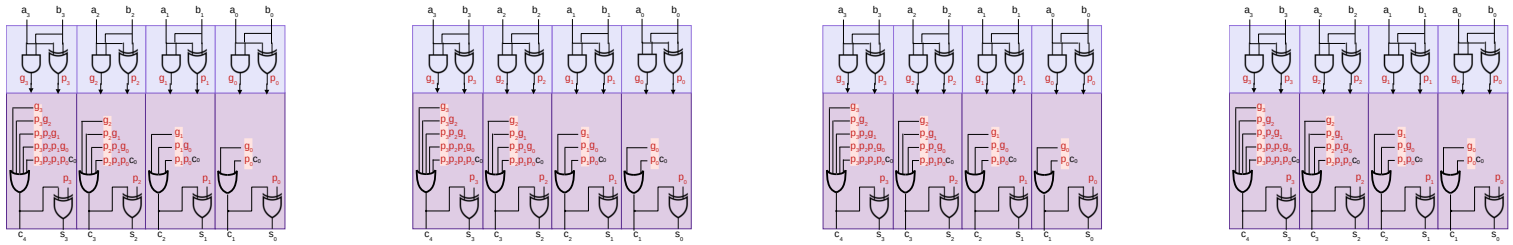
Sum

S_i : 1 gate delay

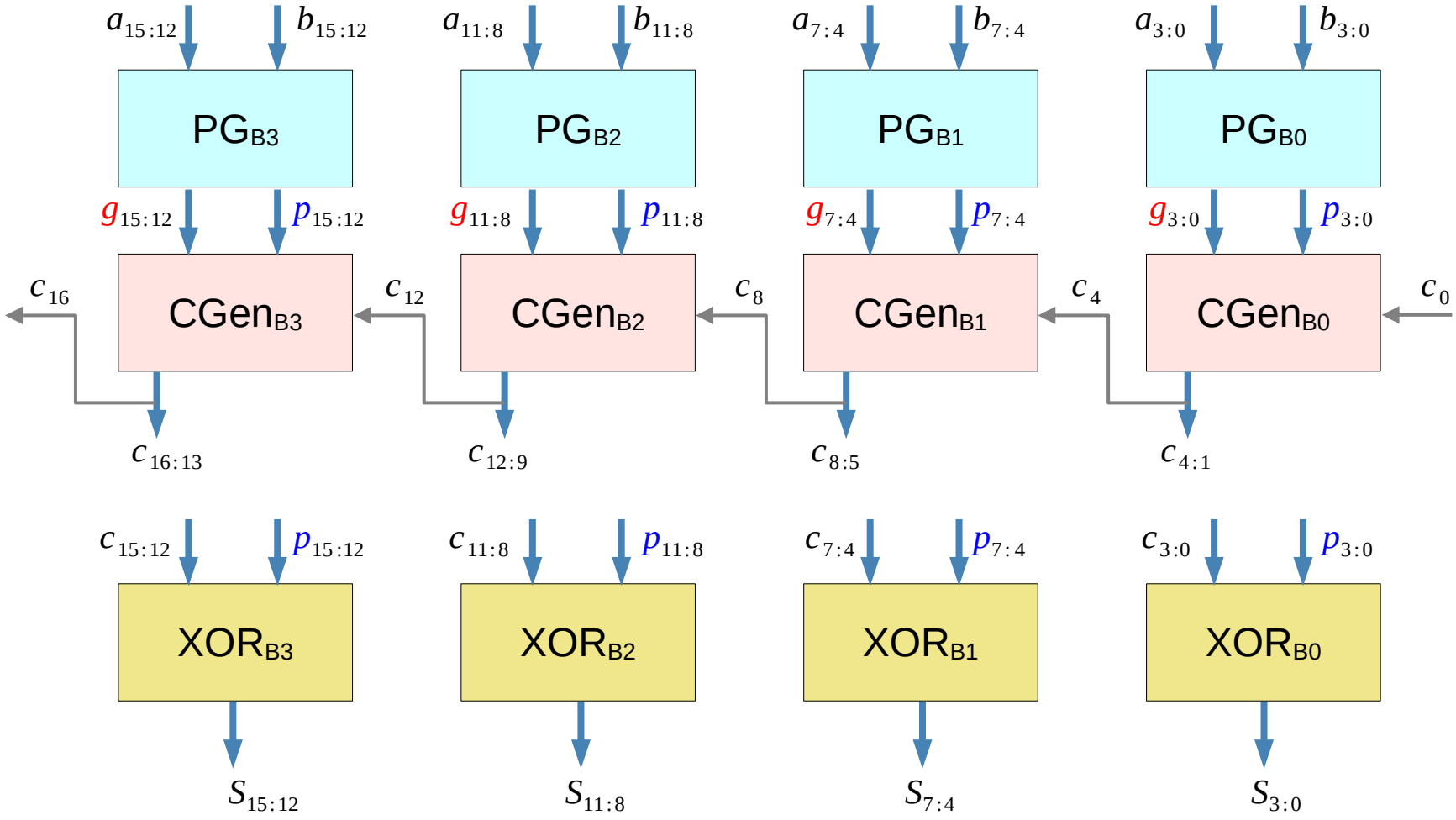
A 16-bit adder using four 4-bit CLA's



partition a 16-bit adder into four 4-bit **CLA** blocks : B3, B2, B1, B0



Block CLA Structures (1)



Block CLA Structures (2)

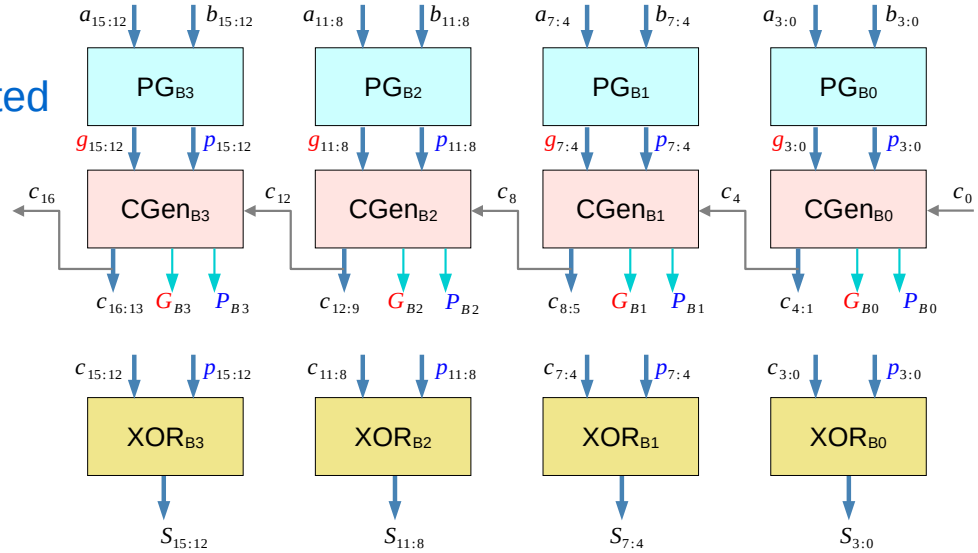
Each block computes 4 g's and 4 p's in parallel

As soon as c_{in}, 4 g's, and 4 p's of each block is available,
each block computes 4 carries in parallel

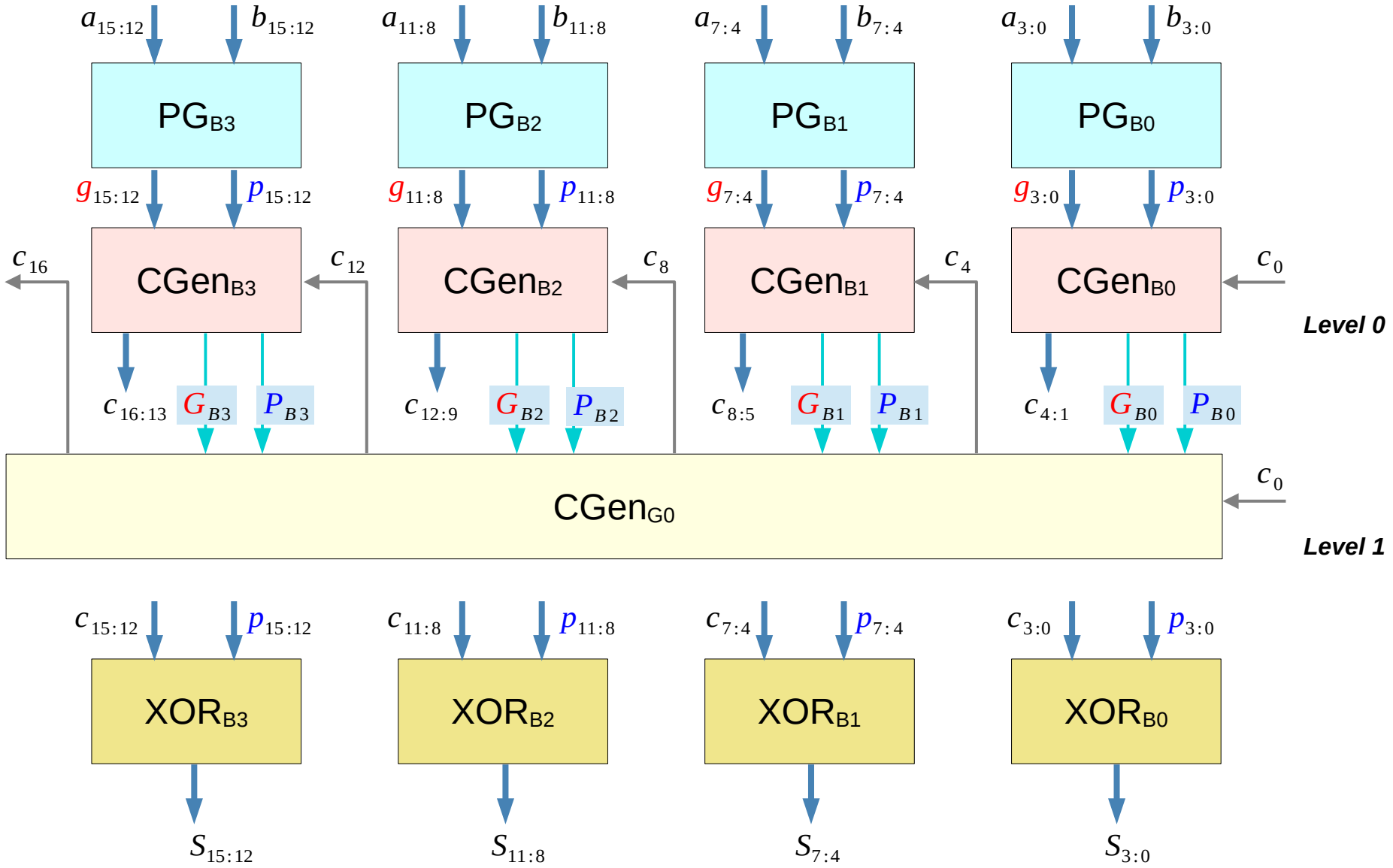
As soon as 4 carries and 4 p's of each block is available,
each block computes 4 sums in parallel

However,
only after c_{out} of B0/B1/B2 is available,
the four carries of B1/B2/B3 can be computed

These can be computed in parallel,
if block carry generate signal G_{Bi}
and block carry propagate signal P_{Bi}
are used

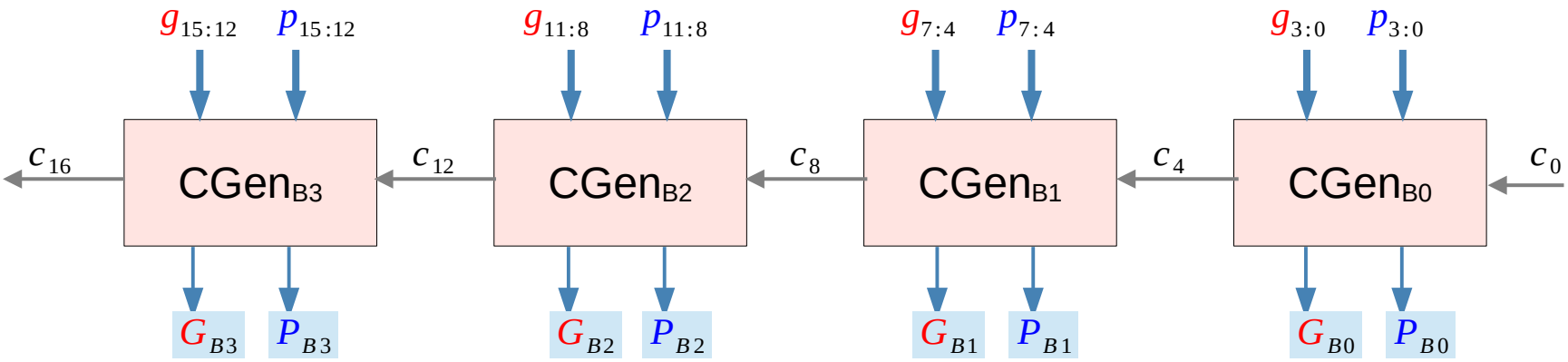


Multi-level CLA



Block Carry Equations

Let us focus on the carry circuits of each block



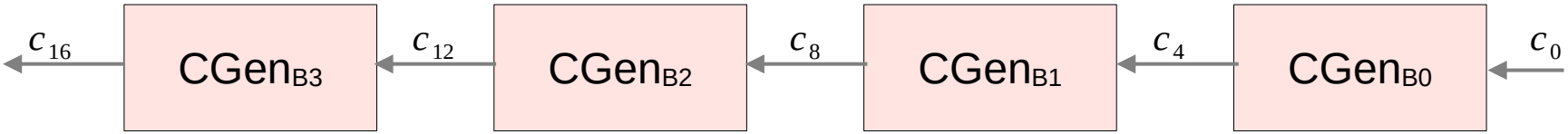
$c_{16} = G_{B3} + P_{B3} c_{12}$ $c_{12} = G_{B2} + P_{B2} c_8$ $c_8 = G_{B1} + P_{B1} c_4$ $c_4 = G_{B0} + P_{B0} c_0$

Define block G and block P signal to compute c_{16}, c_{12}, c_8, c_4 in parallel

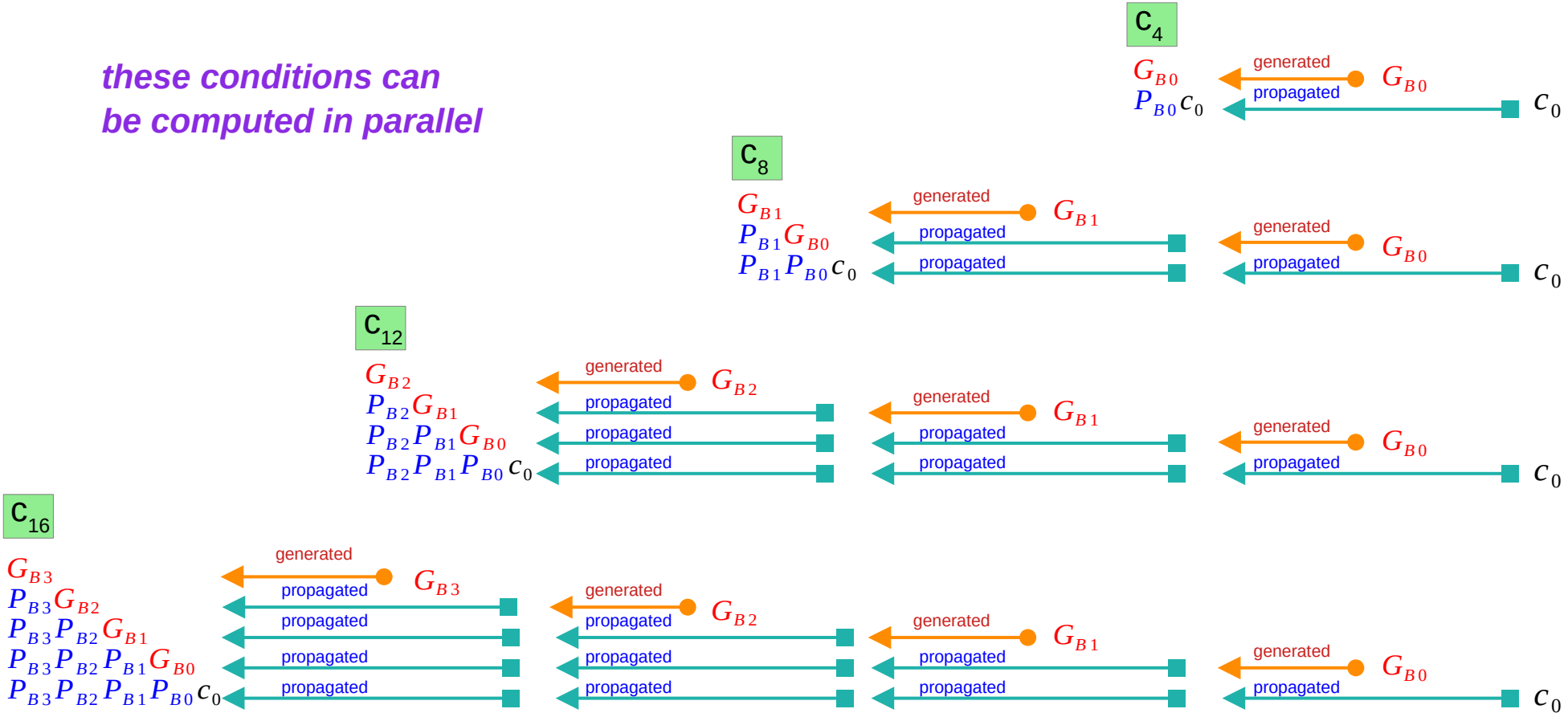
G_{Bi} *denotes the carries which are generated in the block* ***Bi***

$P_{Bi}c_{in}$ *denotes the input carry* c_{in} *which is propagated through the block* ***Bi***

Block Carry Generate and Propagate Signals



these conditions can be computed in parallel



Expanding Block Carry Equations

$$c_{4\cdot(i+1)} = G_{Bi} + P_{Bi} c_{4\cdot i}$$

$$c_4 = G_{B0} + P_{B0} c_0 \quad \Rightarrow \quad c_4 = G_{B0} + P_{B0} c_0$$

$$c_8 = G_{B1} + P_{B1} c_4 \quad \Rightarrow \quad c_8 = G_{B1} + P_{B1}\left[G_{B0} + P_{B0} c_0\right]$$

$$c_{12} = G_{B2} + P_{B2} c_8 \quad \Rightarrow \quad c_{12} = G_{B2} + P_{B2}\left[G_{B1} + P_{B1}\left[G_{B0} + P_{B0} c_0\right]\right]$$

$$c_{16} = G_{B3} + P_{B3} c_{12} \quad \Rightarrow \quad c_{16} = G_{B3} + P_{B3}\left[G_{B2} + P_{B2}\left[G_{B1} + P_{B1}\left[G_{B0} + P_{B0} c_0\right]\right]\right]$$

c_{out} c_{in}

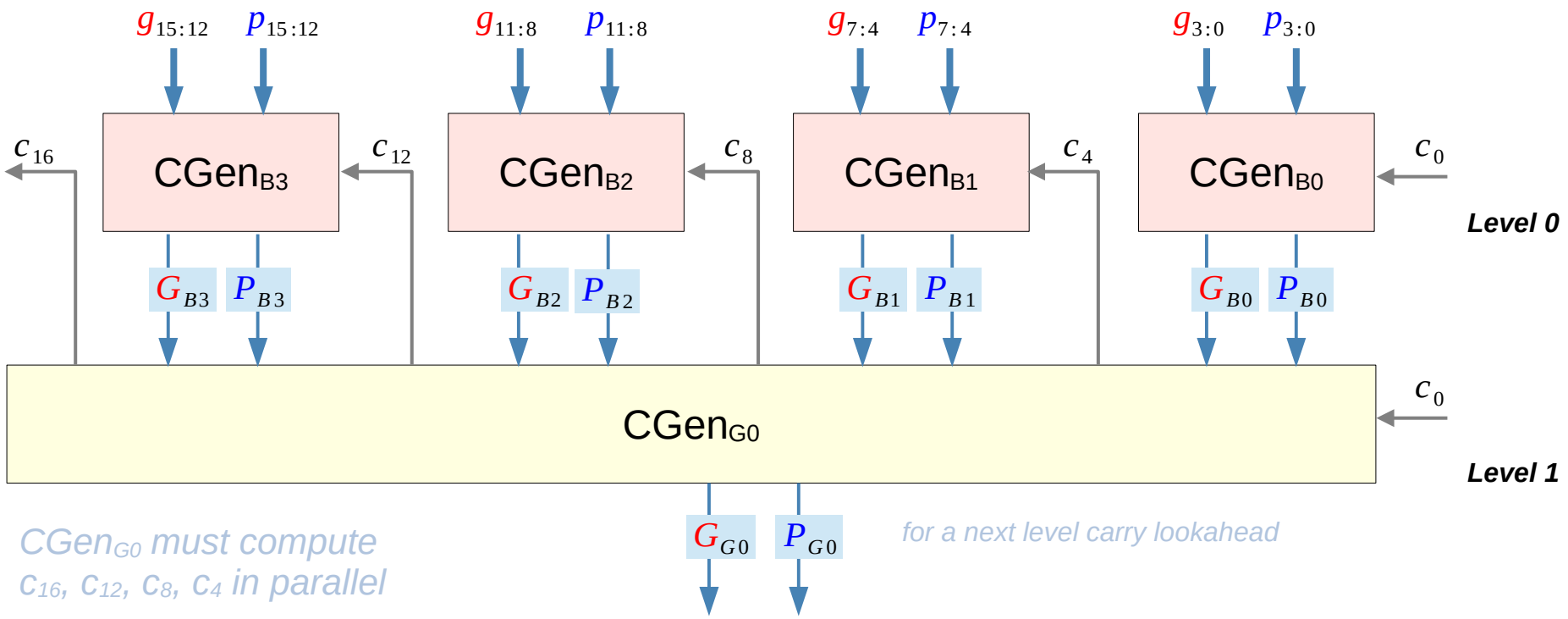
$$G_{B0} + P_{B0} c_0 = c_4$$

$$G_{B1} + P_{B1} G_{B0} + P_{B1} P_{B0} c_0 = c_8$$

$$G_{B2} + P_{B2} G_{B1} + P_{B2} P_{B1} G_{B0} + P_{B2} P_{B1} P_{B0} c_0 = c_{12}$$

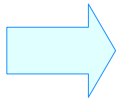
$$G_{B3} + P_{B3} G_{B2} + P_{B3} P_{B2} G_{B1} + P_{B3} P_{B2} P_{B1} G_{B0} + P_{B3} P_{B2} P_{B1} P_{B0} c_0 = c_{16}$$

Block Carry Generating Circuits



Carry Lookahead Logic Comparison

$$c_{i+1} = g_i + p_i c_i$$



Block B0 Carries

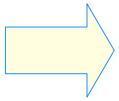
$$g_0 + p_0 c_0 = c_1$$

$$g_1 + p_1 g_0 + p_1 p_0 c_0 = c_2$$

$$g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 = c_3$$

$$g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0 = c_4$$

$$c_{4\cdot(i+1)} = G_{Bi} + P_{Bi} c_{4\cdot i}$$



Group G0 Carries

$$G_{B0} + P_{B0} c_0 = c_4$$

$$G_{B1} + P_{B1} G_{B0} + P_{B1} P_{B0} c_0 = c_8$$

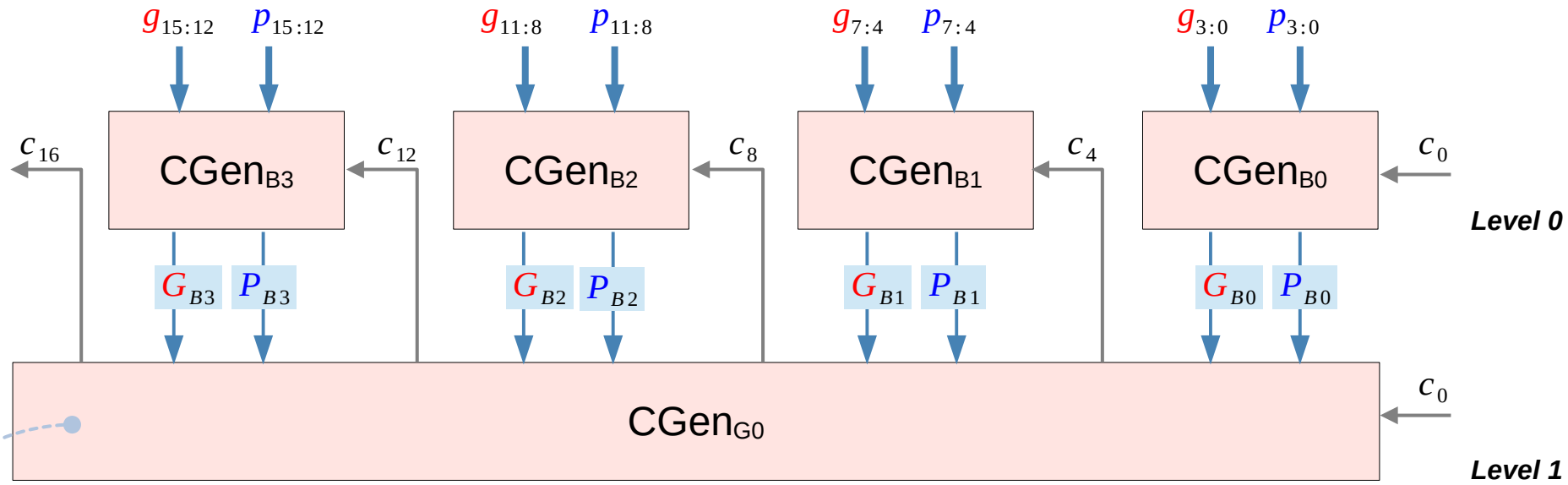
$$G_{B2} + P_{B2} G_{B1} + P_{B2} P_{B1} G_{B0} + P_{B2} P_{B1} P_{B0} c_0 = c_{12}$$

$$G_{B3} + P_{B3} G_{B2} + P_{B3} P_{B2} G_{B1} + P_{B3} P_{B2} P_{B1} G_{B0} + P_{B3} P_{B2} P_{B1} P_{B0} c_0 = c_{16}$$

the same structure

Block Carry Generating Circuits – simplified

Let us focus on the carry circuits of each block



has the same structure as the structure of each block

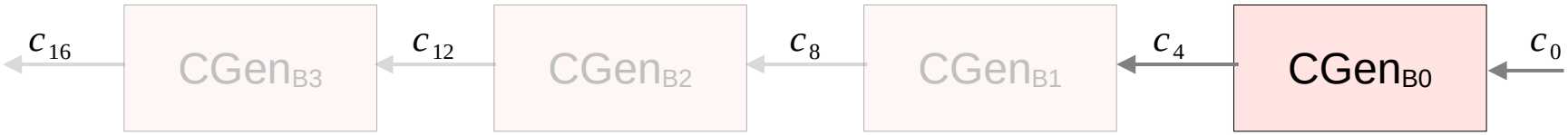
$$G_{B0} + P_{B0}c_0 = c_4$$

$$G_{B1} + P_{B1}G_{B0} + P_{B1}P_{B0}c_0 = c_8$$

$$G_{B2} + P_{B2}G_{B1} + P_{B2}P_{B1}G_{B0} + P_{B2}P_{B1}P_{B0}c_0 = c_{12}$$

$$G_{B3} + P_{B3}G_{B2} + P_{B3}P_{B2}G_{B1} + P_{B3}P_{B2}P_{B1}G_{B0} + P_{B3}P_{B2}P_{B1}P_{B0}c_0 = c_{16}$$

G_{B0}, P_{B0} : Block Carry Generate and Propagate

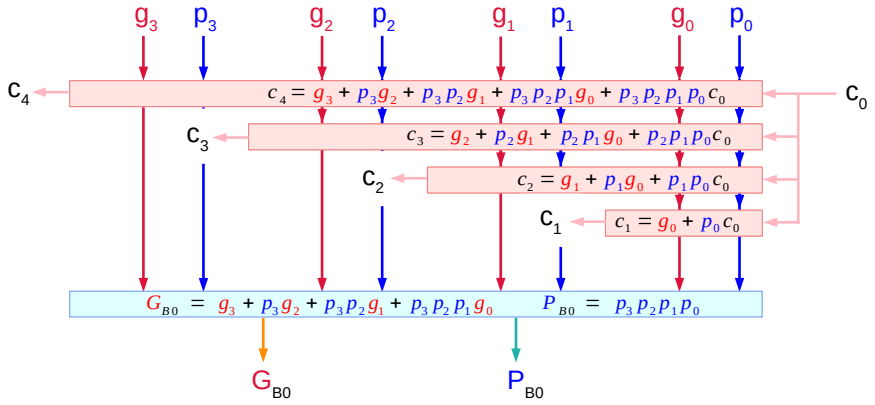


$$c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$$

$$G_{B0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$$

$$P_{B0} = p_3 p_2 p_1 p_0$$

$$c_4 = G_{B0} + P_{B0} c_0$$

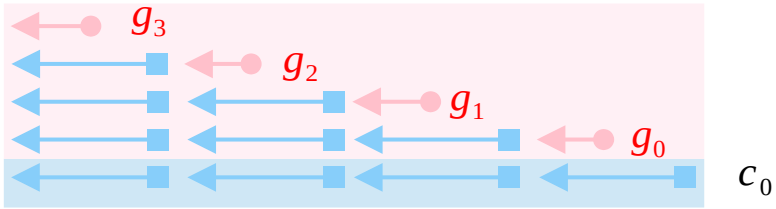


G_{B0} *denotes the generated carries in the block B0*

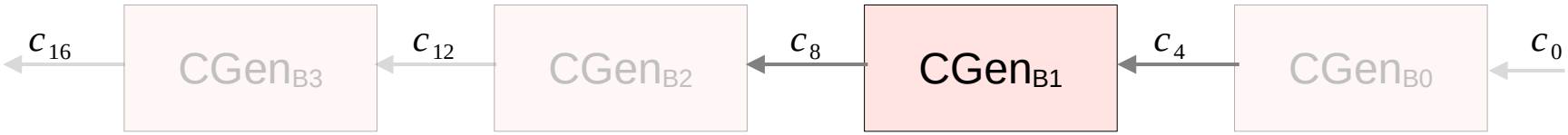
$P_{B0} c_0$ *denotes the the propagated input carry* c_0

c_4

g_3
$p_3 g_2$
$p_3 p_2 g_1$
$p_3 p_2 p_1 g_0$
$p_3 p_2 p_1 p_0 c_0$



G_{B1}, P_{B1} : Block Carry Generate and Propagate

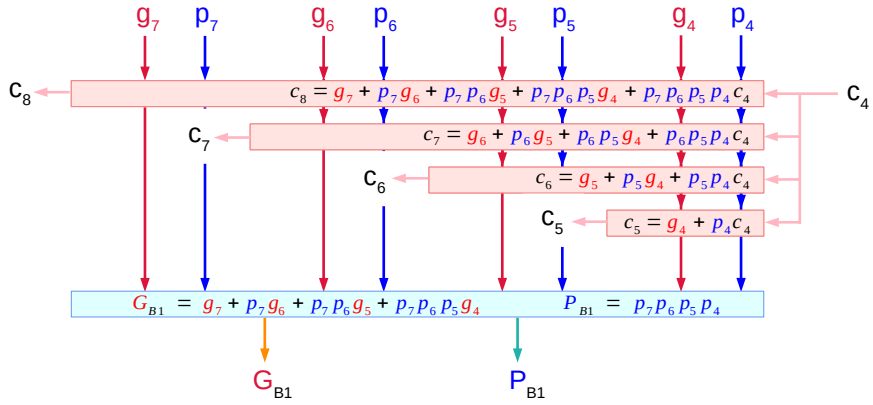


$$c_8 = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 + p_7 p_6 p_5 p_4 c_4$$

$$G_{B1} = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4$$

$$P_{B1} = p_7 p_6 p_5 p_4$$

$$c_8 = G_{B1} + P_{B1} c_4$$

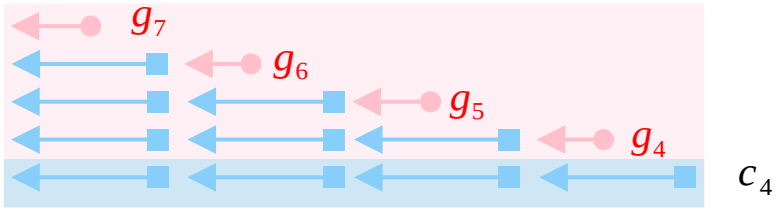


G_{B1} *denotes the generated carries in the block B1*

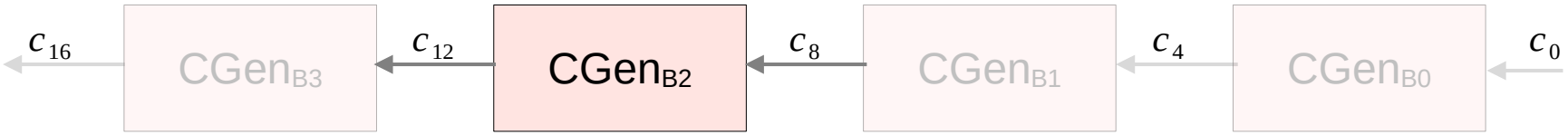
$P_{B1} c_4$ *denotes the the propagated input carry* c_4

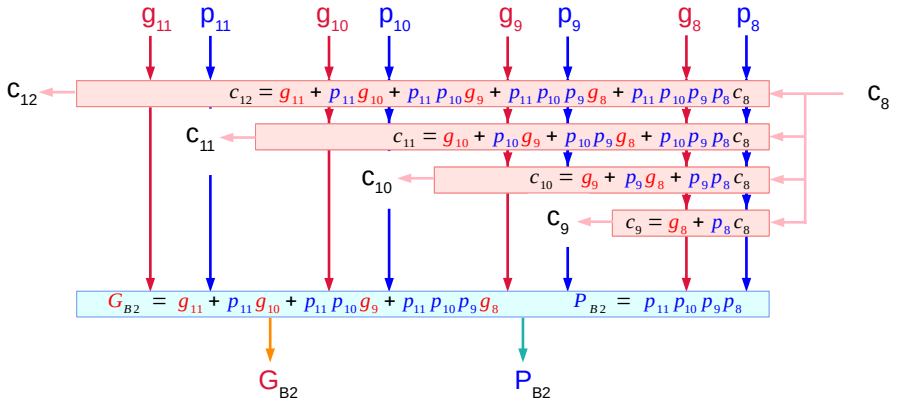
c_8

g_7
$p_7 g_6$
$p_7 p_6 g_5$
$p_7 p_6 p_5 g_4$
$p_7 p_6 p_5 p_4 c_4$



G_{B2}, P_{B2} : Block Carry Generate and Propagate



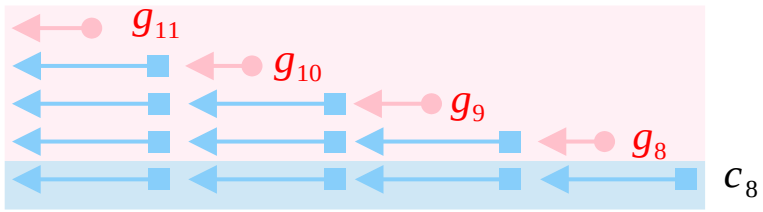


$$c_{12} = g_{11} + p_{11}g_{10} + p_{11}p_{10}g_9 + p_{11}p_{10}p_9g_8 + p_{11}p_{10}p_9p_8c_8$$

$$G_{B2} = g_{11} + p_{11}g_{10} + p_{11}p_{10}g_9 + p_{11}p_{10}p_9g_8$$

$$P_{B2} = p_{11}p_{10}p_9p_8$$

$$c_{12} = G_{B2} + P_{B2}c_8$$



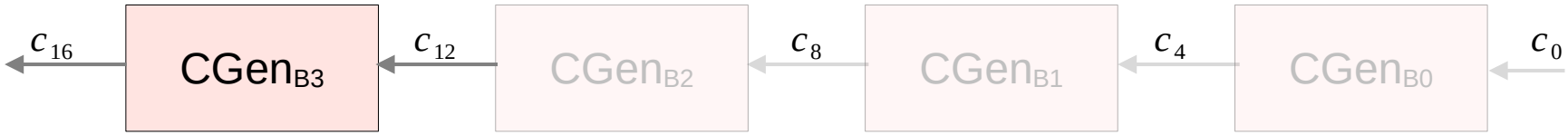
c12

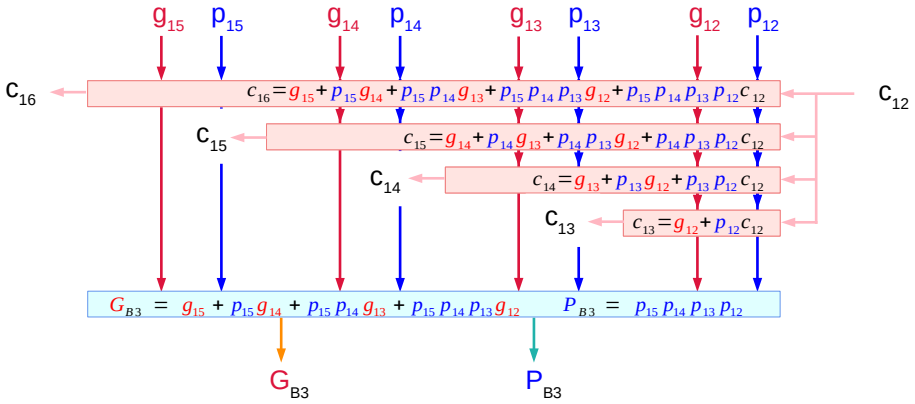
g_{11}
$p_{11}g_{10}$
$p_{11}p_{10}g_9$
$p_{11}p_{10}p_9g_8$
$p_{11}p_{10}p_9p_8c_8$

G_{B2} denotes the generated carries in the block B2

$P_{B2}c_8$ denotes the the propagated input carry c_8

G_{B3}, P_{B3} : Block Carry Generate and Propagate



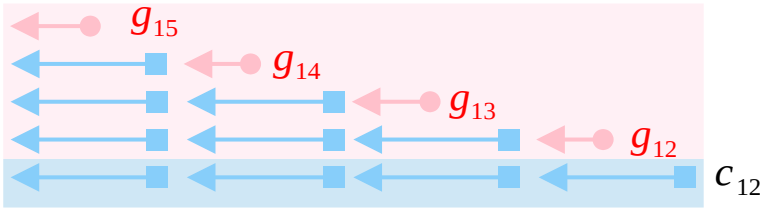


$$c_{16} = g_{15} + p_{15} g_{14} + p_{15} p_{14} g_{13} + p_{15} p_{14} p_{13} g_{12} + p_{15} p_{14} p_{13} p_{12} c_{12}$$

$$G_{B3} = g_{15} + p_{15} g_{14} + p_{15} p_{14} g_{13} + p_{15} p_{14} p_{13} g_{12}$$

$$P_{B3} = p_{15} p_{14} p_{13} p_{12}$$

$$c_{16} = G_{B3} + P_{B3} c_{12}$$



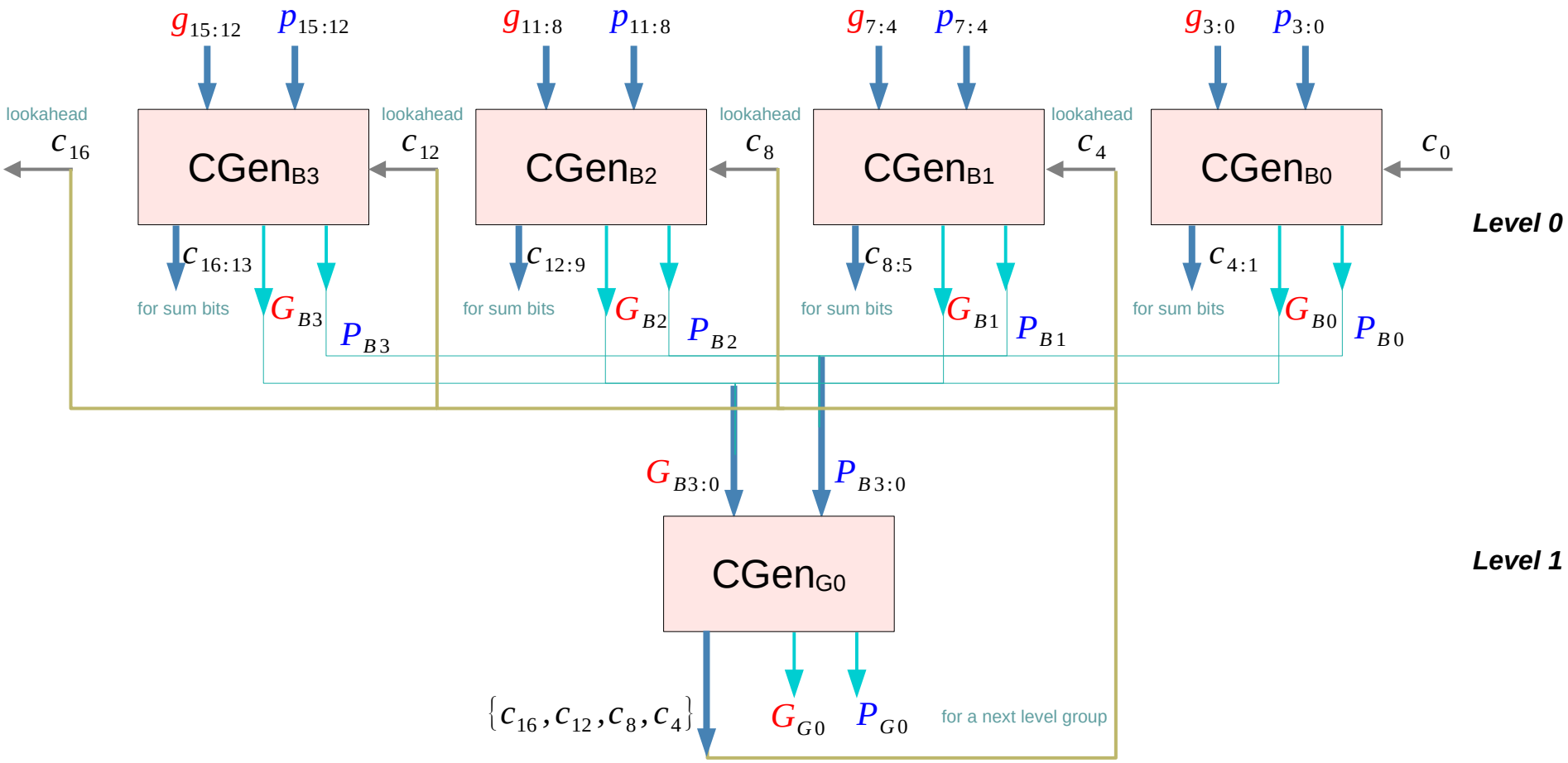
c_{16}

g_{15}
$p_{15} g_{14}$
$p_{15} p_{14} g_{13}$
$p_{15} p_{14} p_{13} g_{12}$
$p_{15} p_{14} p_{13} p_{12} c_{12}$

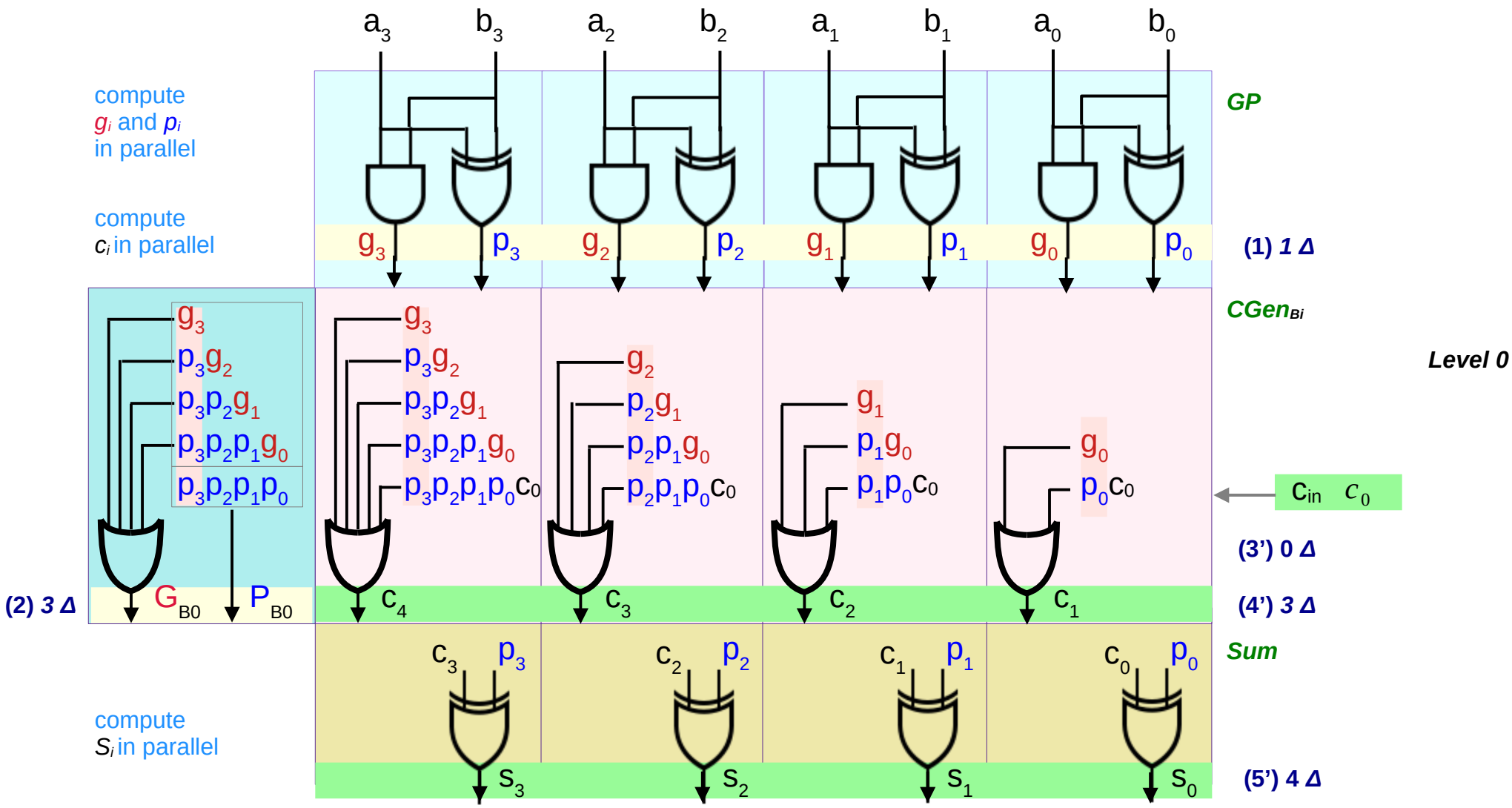
G_{B3} *denotes the generated carries in the block B3*

$P_{B2} c_8$ *denotes the the propagated input carry* c_8

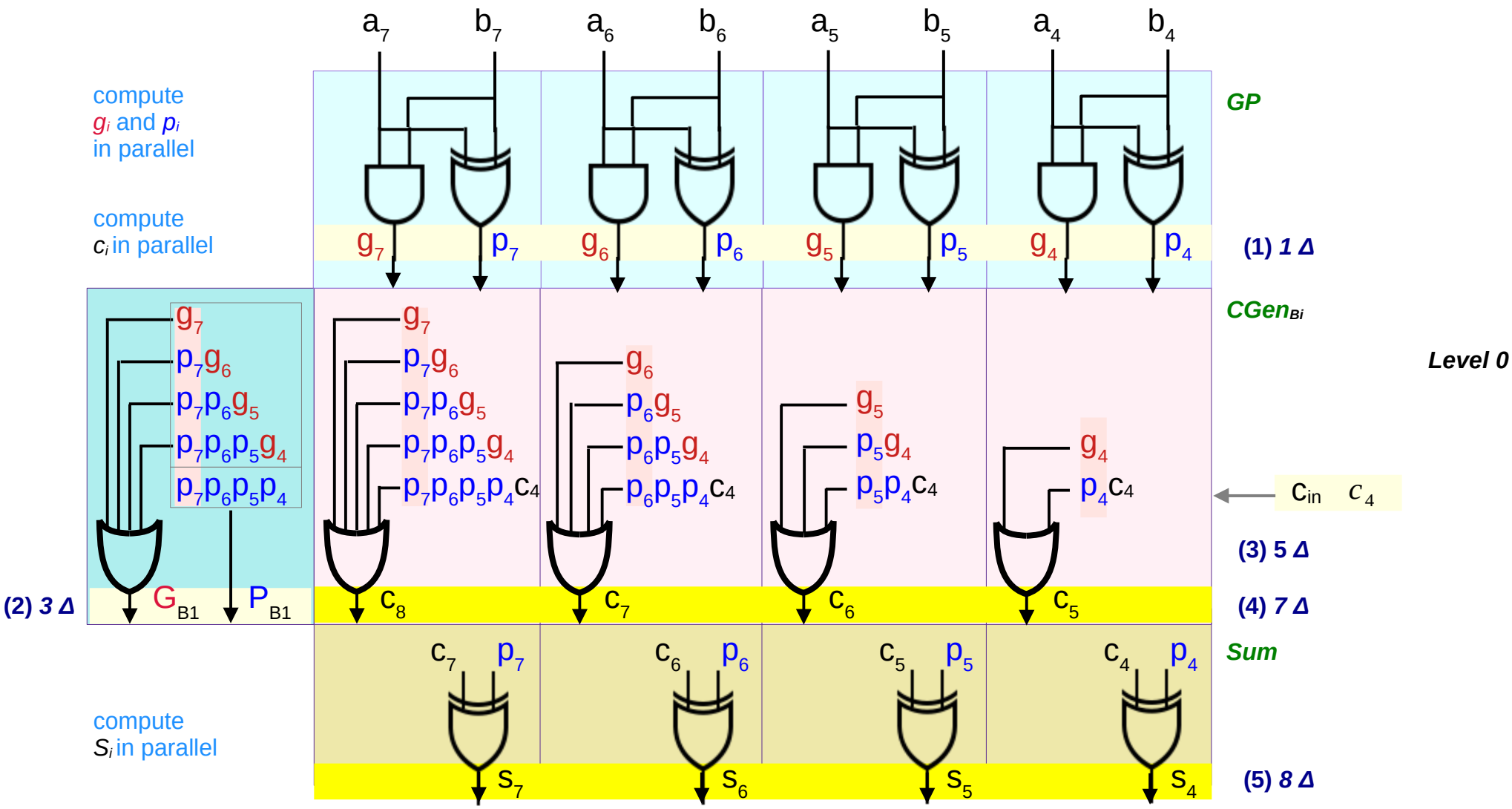
Block Carry Generating Circuits – detailed



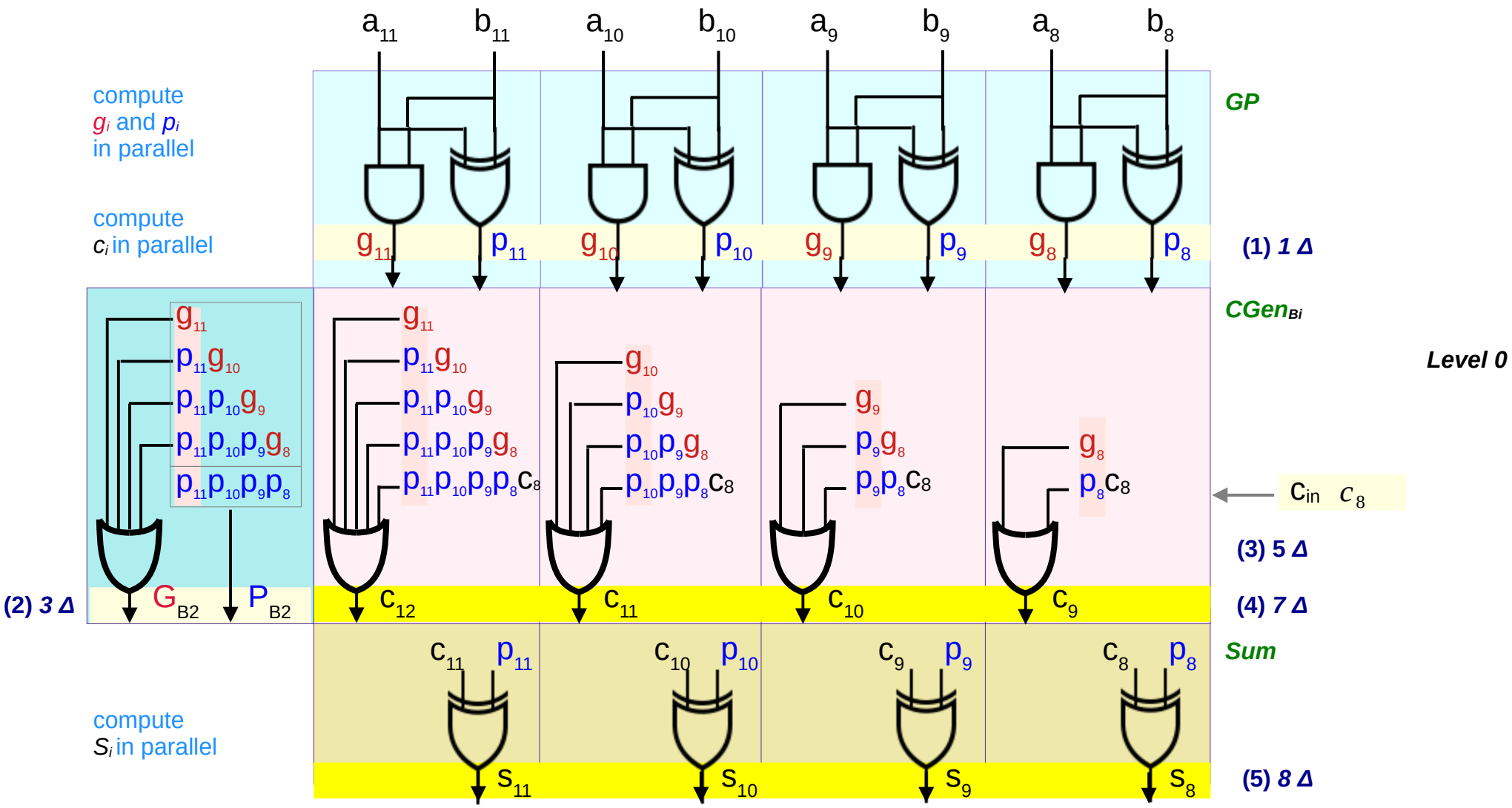
Simple Carry Lookahead Adder – B0



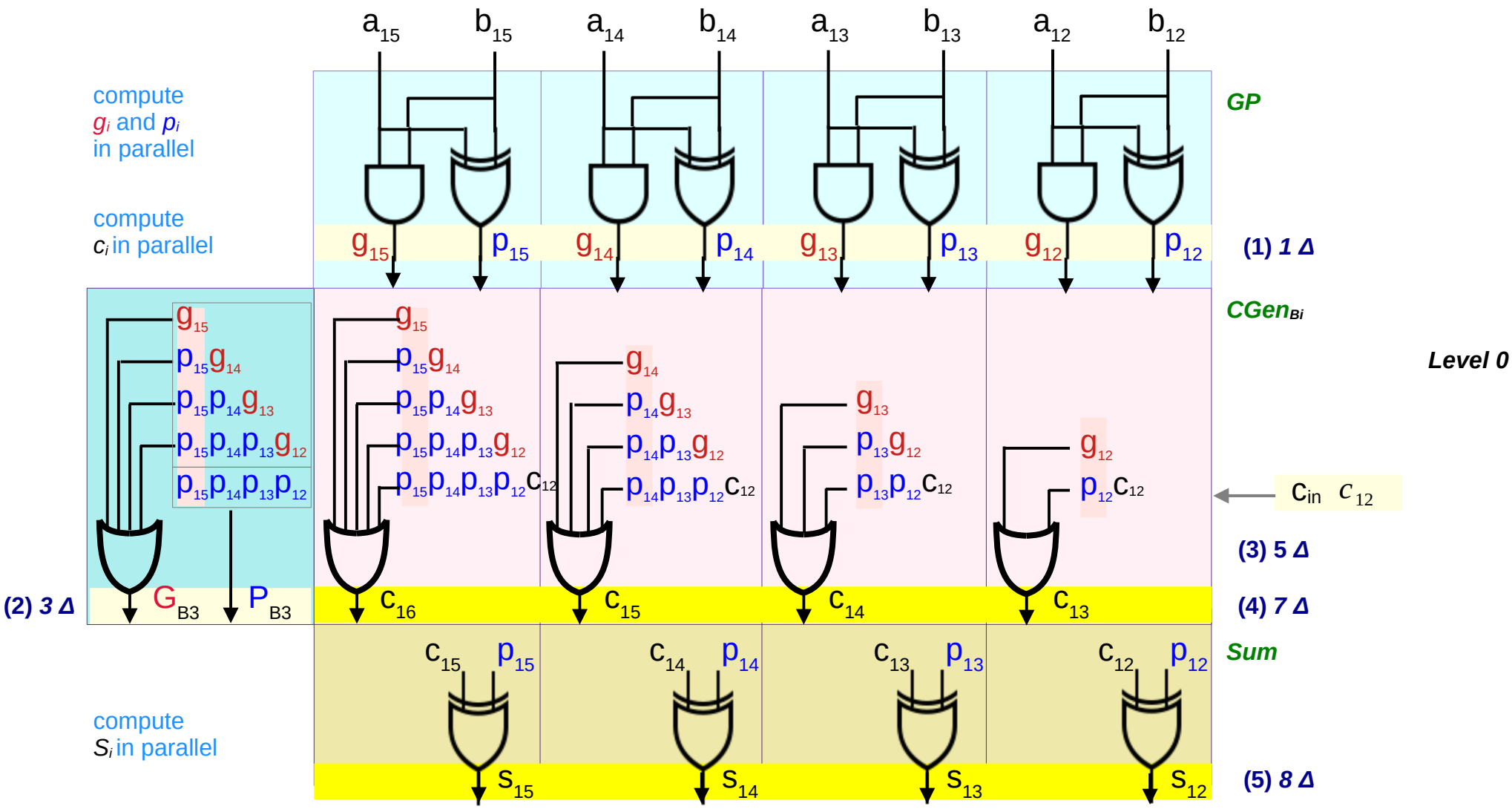
Simple Carry Lookahead Adder – B1



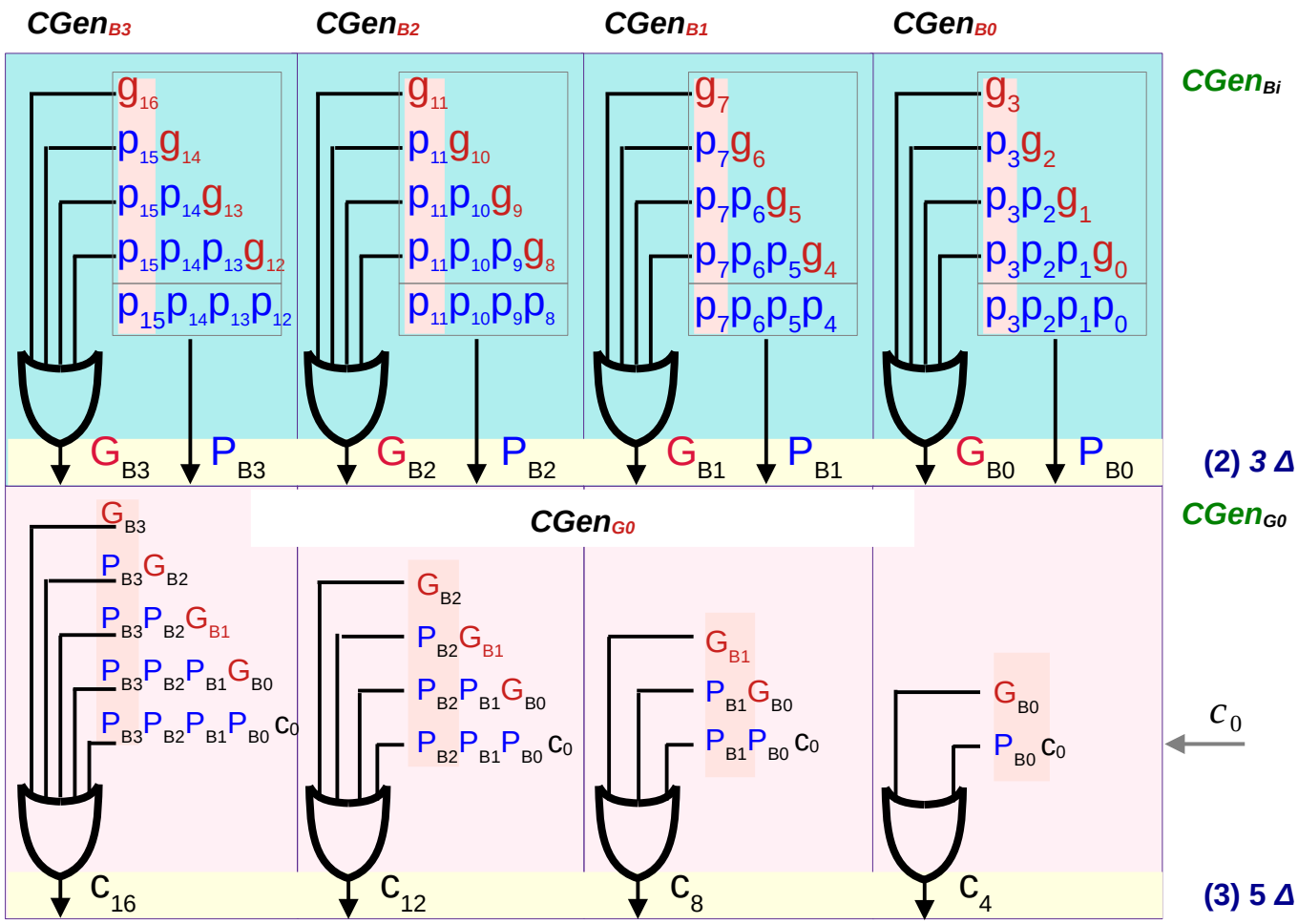
Simple Carry Lookahead Adder – B2



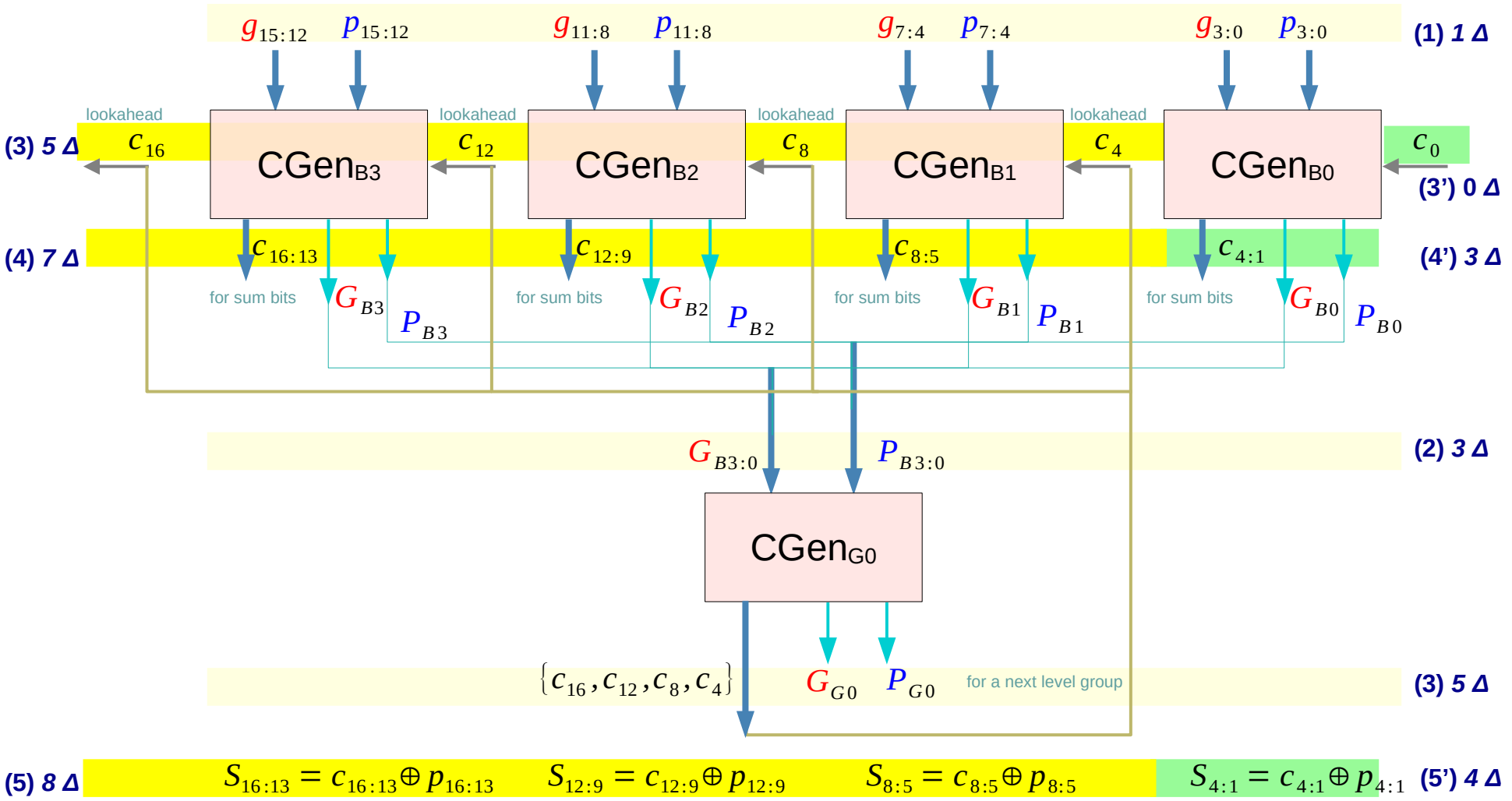
Simple Carry Lookahead Adder – B3



A 2-Level Carry Lookahead Adder – Level 1



Delay calculations of 2-Level CLA (1)



Delay calculations of 2-Level CLA (2)

Starting at time of zero:

- calculation of p_i and g_i is done at time 1,
- calculation of the P_{Bi} is done at time 2,
- calculation of the G_{Bi} is done at time 3,
- calculation of the inputs for the $CGen_{Bi}$ from the $CGen_{G0}$ are done at:
  - time 0 for $CGen_{B0}$,
  - time 5 for $CGen_{B1}$, $CGen_{B2}$, $CGen_{B3}$,
- calculation of the S_i are done at:
  - time 4 for Sum_{B0},
  - time 8 for Sum_{B1}, Sum_{B2}, Sum_{B3},
- calculation of the final carry bit (c_{16}) is done at time 5.

The maximal time is 8 gate delays (for $S_{[4:15]}$).

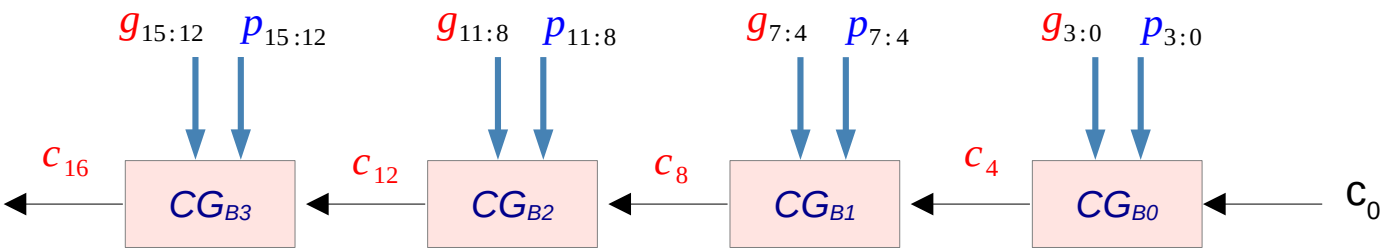
https://en.wikipedia.org/wiki/Carry-lookahead_adder

A Variation of 2 Level CLA

Ripple Carry Adder
+ Carry Lookahead

S. & D. Harris, Digital Design and Computer Architecture 2nd ed.

Block Carry Generators



block carry generators	block carry in	block carry out
	c_{out}	c_{in}
CG_{B0}	c_4	c_0
CG_{B1}	c_8	c_4
CG_{B2}	c_{12}	c_8
CG_{B3}	c_{16}	c_{12}

A simplified carry lookahead circuit

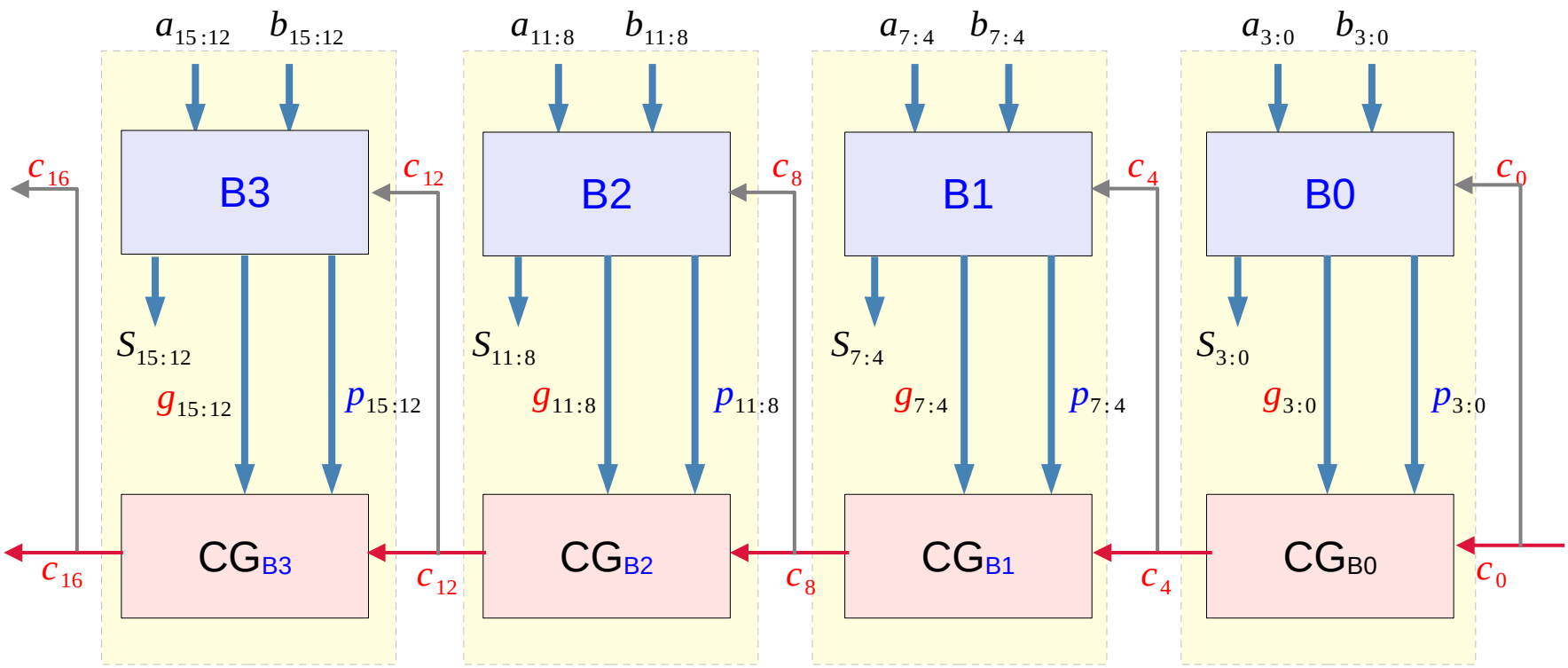
computes block carry output much faster than a ripple carry adder block does

Whereas the traditional carry lookahead adder computes all the carries,

the simplified carry lookahead circuit computes only the block carry output

https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

A 16-bit adder using four 4-bit Simplified CLA's (1)

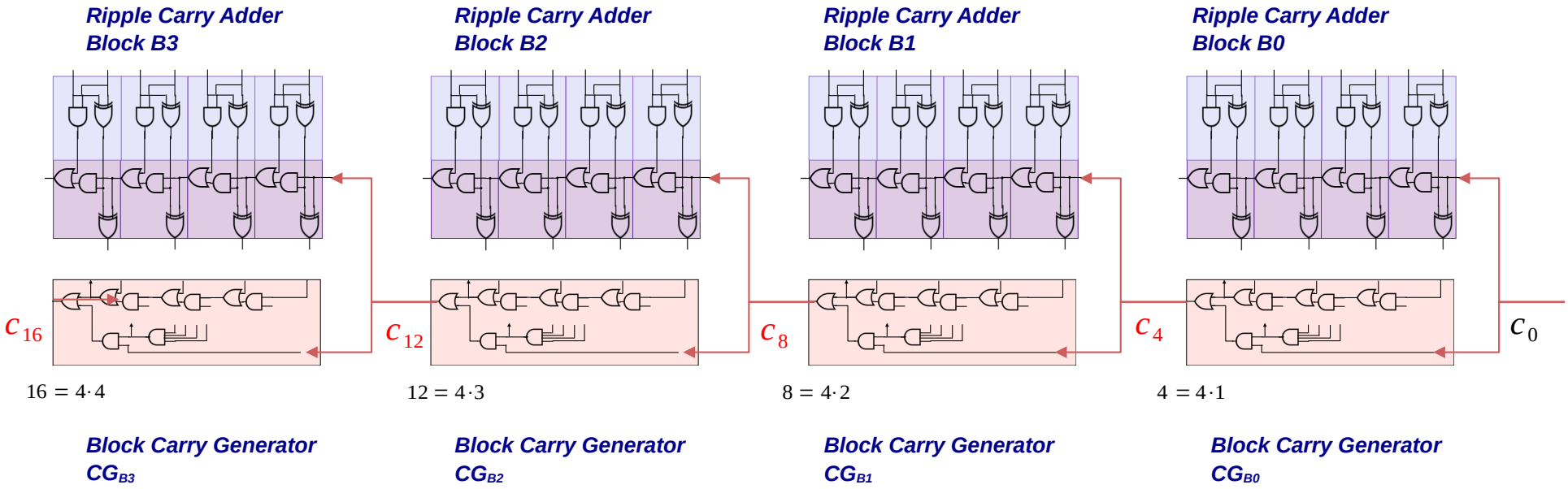


B3, B2, B1, B0 : 4-bit ripple carry adder blocks

CG_{B3}, CG_{B2}, CG_{B1}, CG_{B0} : 4-bit block carry generators

c_{16}, c_{12}, c_8, c_4 : block carries

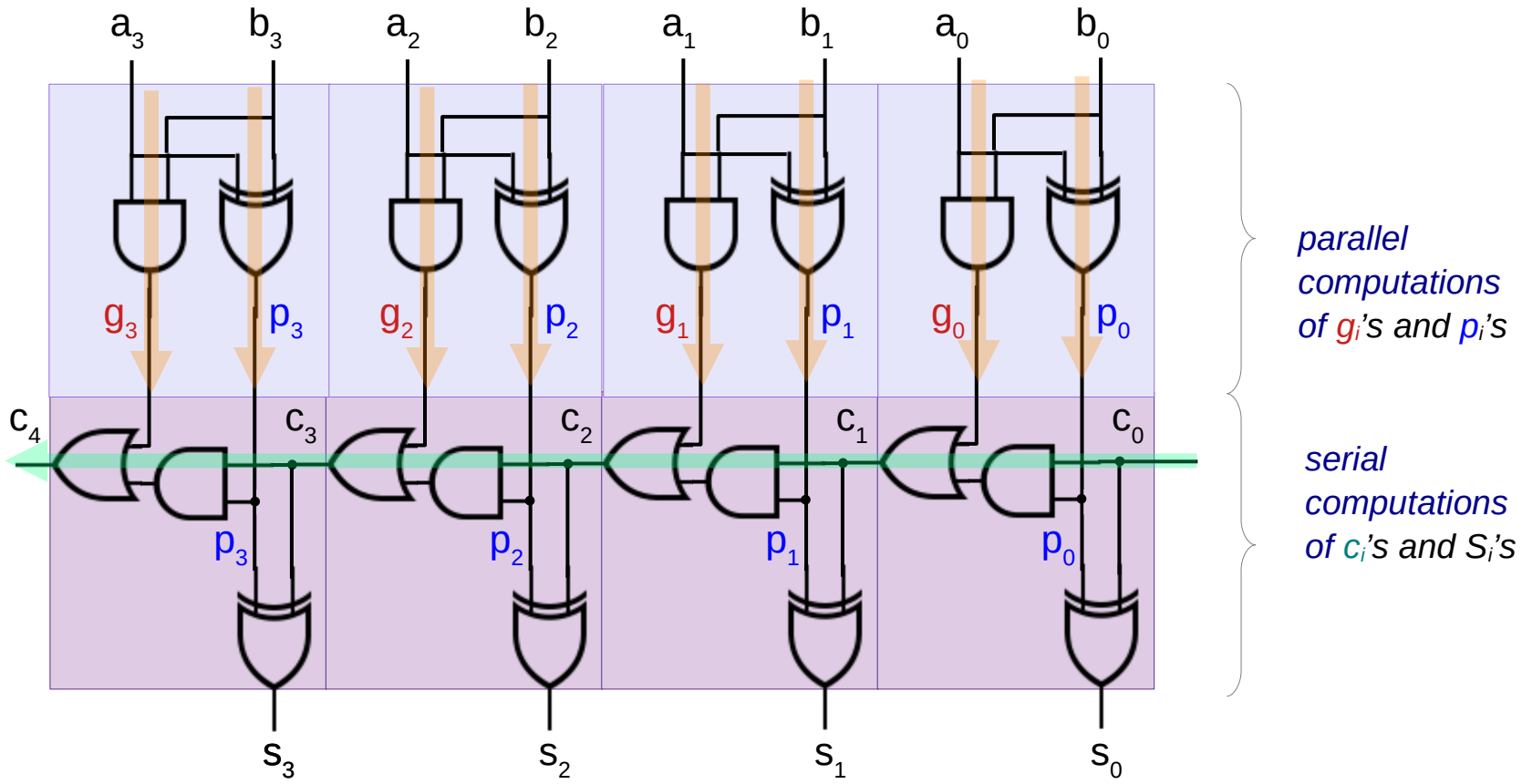
A 16-bit adder using four 4-bit Simplified CLA's (2)



As soon as the block carry input c_{in} and g_i's and p_i's are available, a carry lookahead operation can be performed in parallel with a ripple carry adder operation

https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

A 4-bit Ripple Carry Adder Block B0



when g_i's and p_i's are available,
the critical path : $c_4 \leftarrow c_3 \leftarrow c_2 \leftarrow c_1 \leftarrow c_0$: 8 gate delay

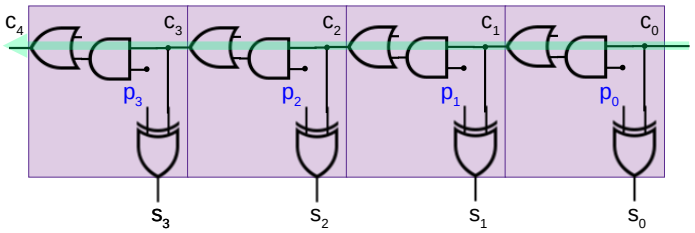
https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

Two ways of computing a block carry (1)

$$c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$$

Method 1 : using multiple 2-input gates

$$c_4 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0)))$$



In a ripple carry adder,
to compute sum bits S_3, S_2, S_1, S_0,
carry bits c_4, c_3, c_2, c_1 must be computed

Method 2 : using multi-input gates

$$c_4 = G_{B0} + P_{B0} c_0$$

$$G_{B0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$$

$$P_{B0} = p_3 p_2 p_1 p_0$$

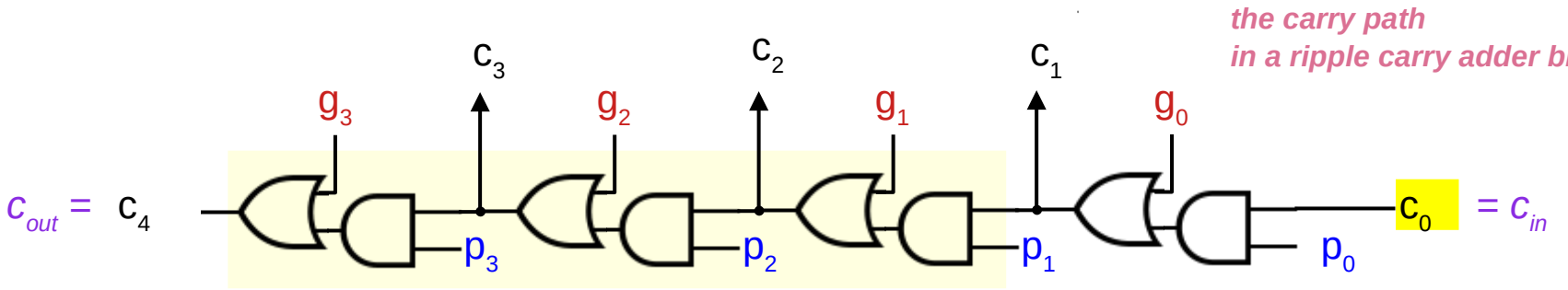
As g_i's and p_i's are computed in parallel,
G_{B0} and P_{B0} can be computed in parallel

the path from c_0 to c_4 is much smaller,
compared to the 1st method

Two ways of computing a block carry (2)

Method 1 : using multiple 2-input gates

$$c_4 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0)))$$

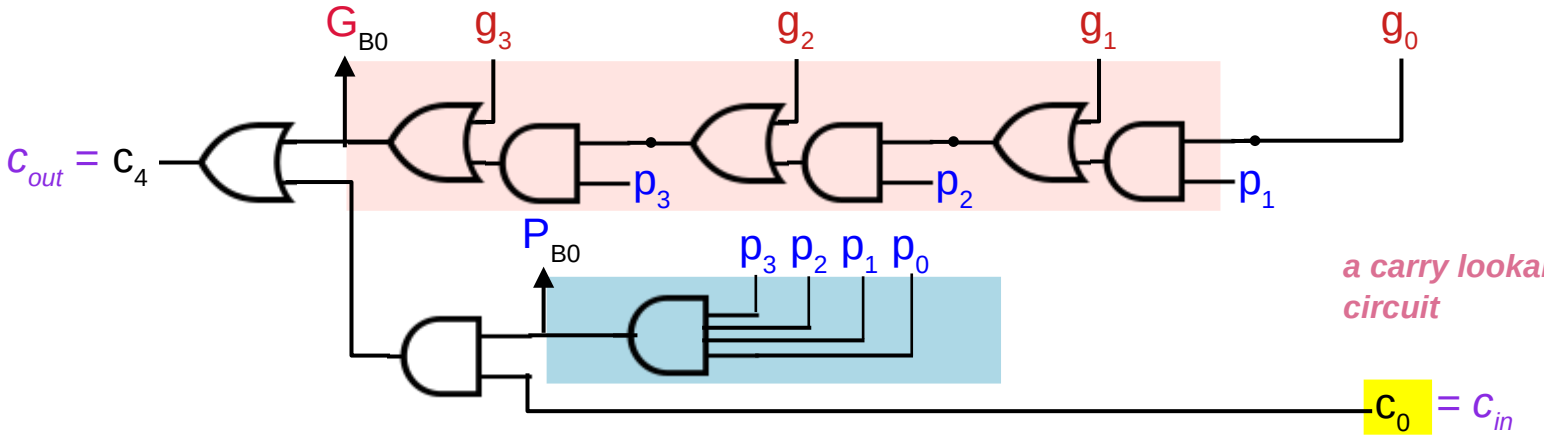


Method 2 : using multi-input gates

$$c_4 = G_{B0} + P_{B0} c_0$$

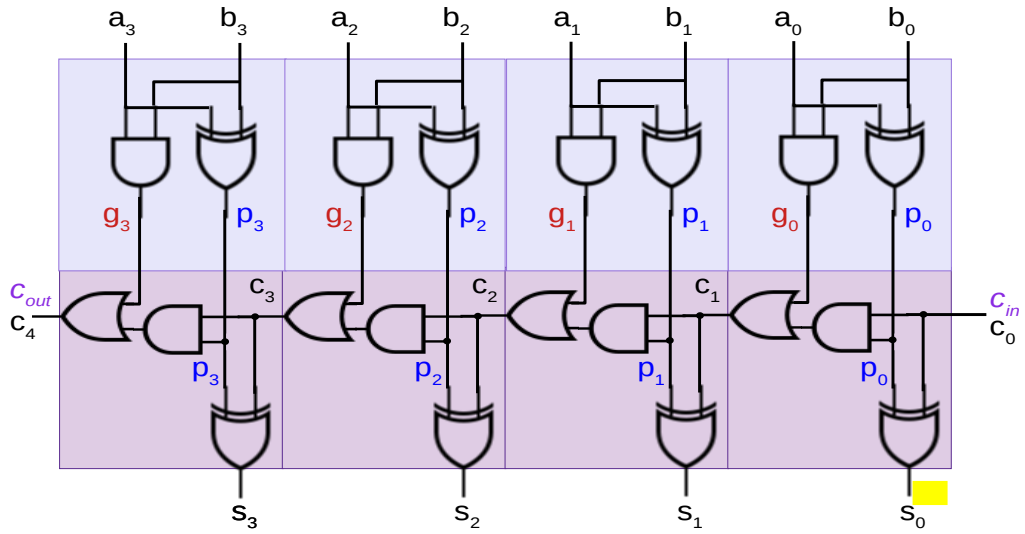
$$G_{B0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$$

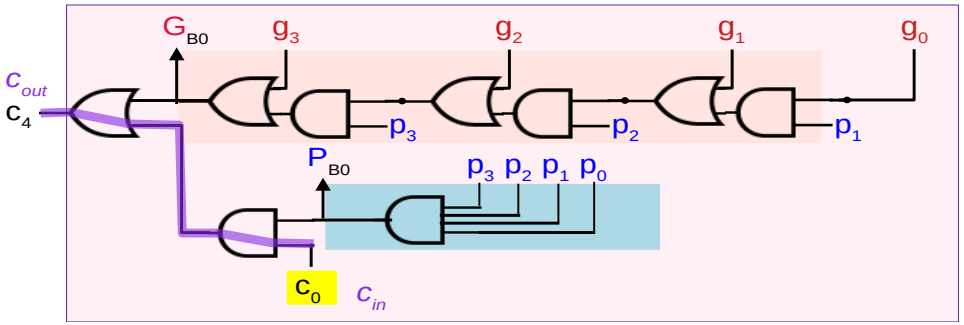
$$P_{B0} = p_3 p_2 p_1 p_0$$



A Ripple Carry Adder Block and a Block Carry Generator

a ripple carry adder block





a block carry generator

Method 1 : using multiple 2-input gates

$$c_4 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0)))$$

Method 2 : using multi-input gates

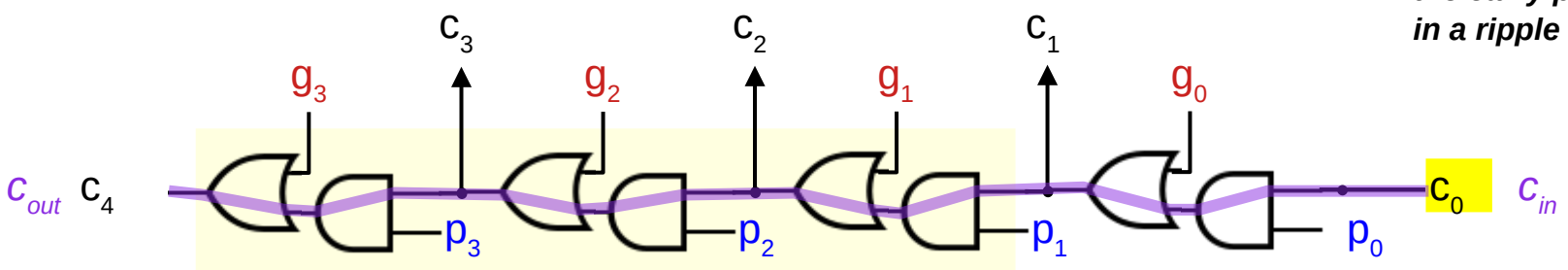
$$c_4 = G_{B0} + P_{B0} c_0$$

$$G_{B0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$$

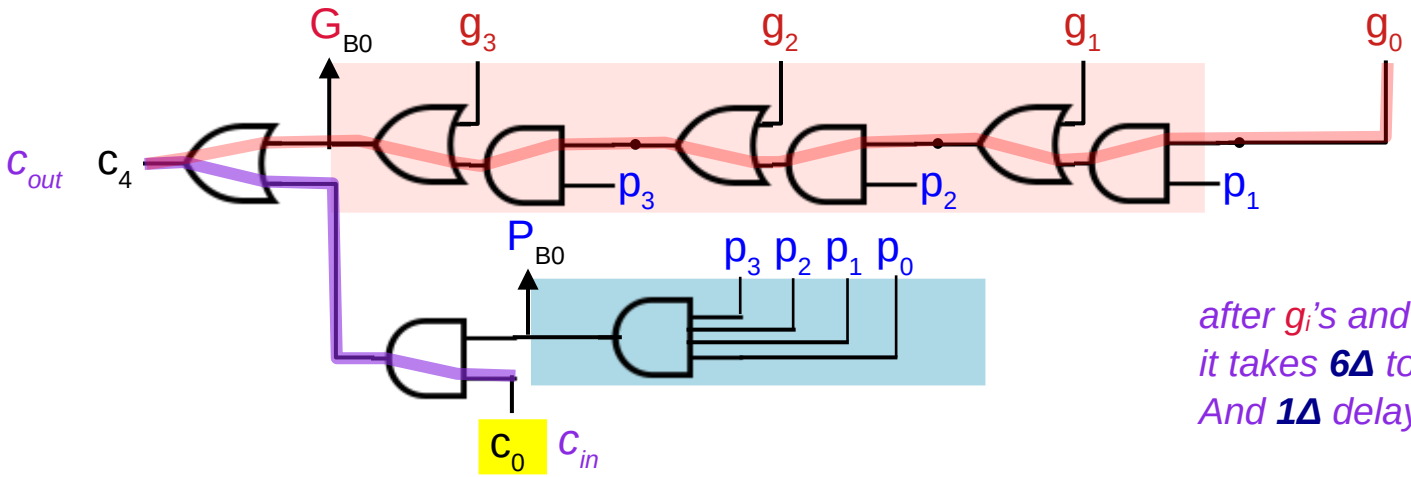
$$P_{B0} = p_3 p_2 p_1 p_0$$

After all g_i's and p_i's are available

The longest path : the path from c_{in} to c_{out} : ***8Δ***



the carry path in a ripple carry adder block



a carry lookahead circuit

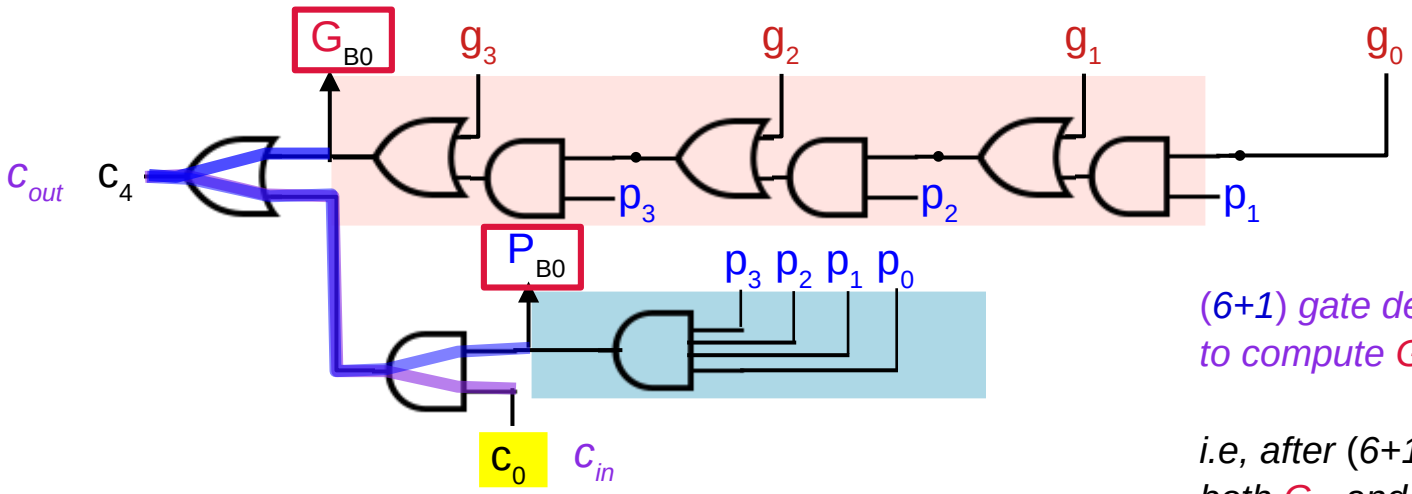
after g_i's and p_i's are available,
it takes ***6Δ*** *to compute G_{B0}*
And ***1Δ*** *delay to compute P_{B0}*

the longest path : the path from g_0 to c_{out} : ***7Δ***

the shortest path : the path from c_{in} to c_{out} : ***2Δ***

Δ = a gate delay

After all G_{Bi}'s and P_{Bi}'s are available



a carry lookahead circuit

(6+1) gate delay is sufficient to compute G_{B0} and P_{B0} from a_i and b_i

i.e, after (6+1) gate delay, both G_{B0} and P_{B0} are available

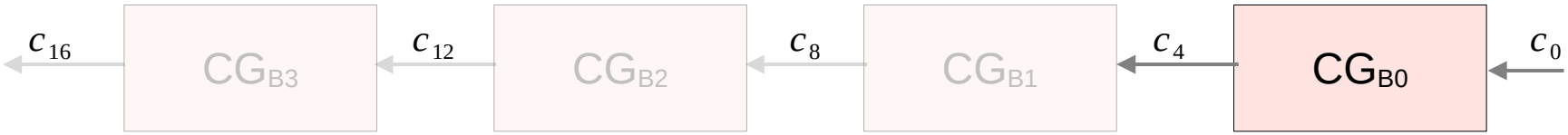
after (***6+1***) ***Δ***, G_{B0} *is available.*
after (***1+1***) ***Δ***, P_{B0} *is available.*

When G_{B0} is available and thus and P_{B0} is available,

the longest path : *the path from c_{in} (= c_0) to c_{out} (= c_4) :* ***2 Δ***
the path from P_{B0} to c_{out} (= c_4) : ***2 Δ***

the shortest path : *the path from G_{B0} to c_{out} (= c_4) :* ***1 Δ***

Block Carry Generator B0



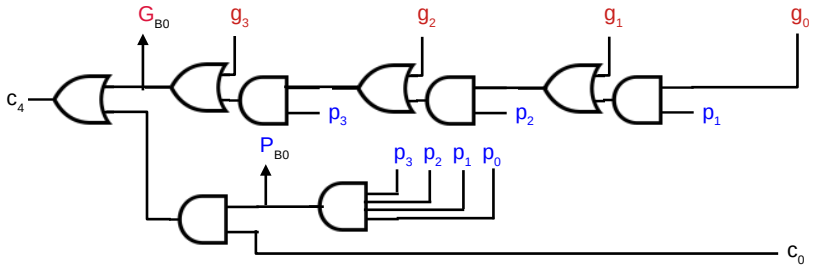
a block carry generator B0

$$c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$$

$$G_{B0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$$

$$P_{B0} = p_3 p_2 p_1 p_0$$

$$c_4 = G_{B0} + P_{B0} c_0$$



$G_{B0} \leftarrow g_0$: 6Δ
$G_{B0} \leftarrow g_1$: 5Δ
$G_{B0} \leftarrow g_2$: 3Δ
$G_{B0} \leftarrow g_3$: 1Δ

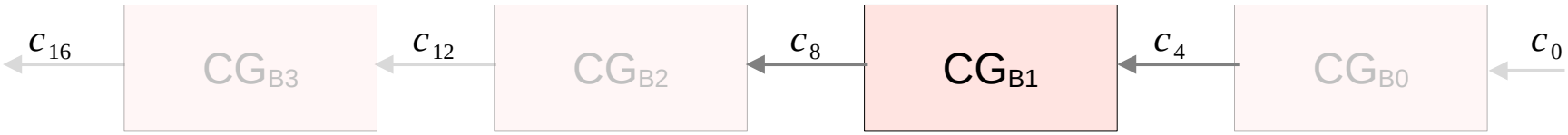
$G_{B0} \leftarrow p_1$: 6Δ
$G_{B0} \leftarrow p_2$: 4Δ
$G_{B0} \leftarrow p_3$: 2Δ

$P_{B0} \leftarrow p_0$: 1Δ
$P_{B0} \leftarrow p_1$: 1Δ
$P_{B0} \leftarrow p_2$: 1Δ
$P_{B0} \leftarrow p_3$: 1Δ

$delay(c_4 \leftarrow g_i) \le 7\Delta$
$delay(c_4 \leftarrow p_i) \le 7\Delta$
$delay(c_4 \leftarrow c_0) = 2\Delta$

$delay(G_{B0} \leftarrow g_i) \le 6\Delta$
$delay(G_{B0} \leftarrow p_i) \le 6\Delta$
$delay(P_{B0} \leftarrow p_i) = 1\Delta$

Block Carry Generator B1



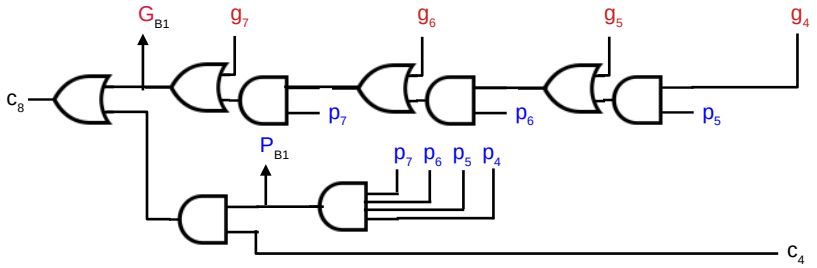
a block carry generator B1

$$c_8 = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 + p_7 p_6 p_5 p_4 c_4$$

$$G_{B1} = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4$$

$$P_{B1} = p_7 p_6 p_5 p_4$$

$$c_8 = G_{B1} + P_{B1} c_4$$



$G_{B1} \leftarrow g_7 : 6\Delta$
$G_{B1} \leftarrow g_6 : 5\Delta$
$G_{B1} \leftarrow g_5 : 3\Delta$
$G_{B1} \leftarrow g_4 : 1\Delta$

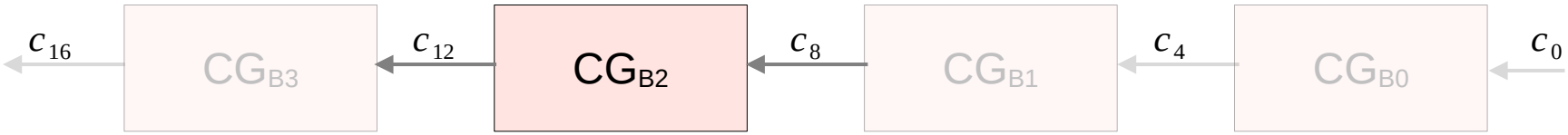
$G_{B1} \leftarrow p_5 : 6\Delta$
$G_{B1} \leftarrow p_6 : 4\Delta$
$G_{B1} \leftarrow p_7 : 2\Delta$

$P_{B1} \leftarrow p_4 : 1\Delta$
$P_{B1} \leftarrow p_5 : 1\Delta$
$P_{B1} \leftarrow p_6 : 1\Delta$
$P_{B1} \leftarrow p_7 : 1\Delta$

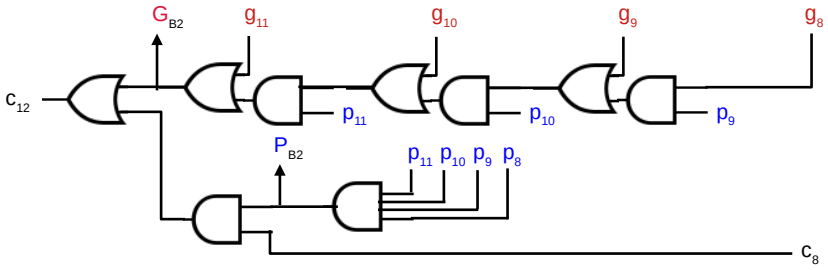
$delay(c_8 \leftarrow g_i) \le 7\Delta$
$delay(c_8 \leftarrow p_i) \le 7\Delta$
$delay(c_8 \leftarrow c_4) = 2\Delta$

$delay(G_{B1} \leftarrow g_i) \le 6\Delta$
$delay(G_{B1} \leftarrow p_i) \le 6\Delta$
$delay(P_{B1} \leftarrow p_i) = 1\Delta$

Block Carry Generator B2



a block carry generator B2



$$c_{12} = g_{11} + p_{11}g_{10} + p_{11}p_{10}g_9 + p_{11}p_{10}p_9g_8 + p_{11}p_{10}p_9p_8c_8$$

$$G_{B2} = g_{11} + p_{11}g_{10} + p_{11}p_{10}g_9 + p_{11}p_{10}p_9g_8$$
$$P_{B2} = p_{11}p_{10}p_9p_8$$

$$c_{12} = G_{B2} + P_{B2}c_8$$

$G_{B2} \leftarrow g_{11}$: 6Δ
$G_{B2} \leftarrow g_{10}$: 5Δ
$G_{B2} \leftarrow g_9$: 3Δ
$G_{B2} \leftarrow g_8$: 1Δ

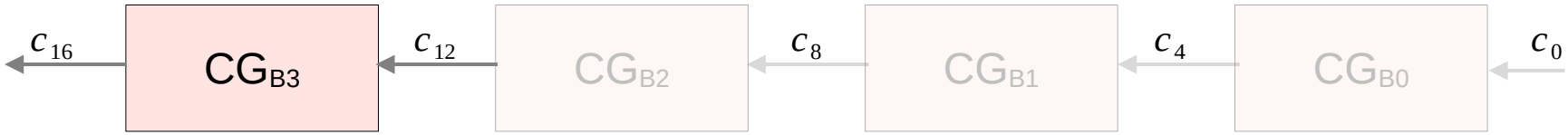
$G_{B2} \leftarrow p_9$: 6Δ
$G_{B2} \leftarrow p_{10}$: 4Δ
$G_{B2} \leftarrow p_{11}$: 2Δ

$P_{B2} \leftarrow p_8$: 1Δ
$P_{B2} \leftarrow p_9$: 1Δ
$P_{B2} \leftarrow p_{10}$: 1Δ
$P_{B2} \leftarrow p_{11}$: 1Δ

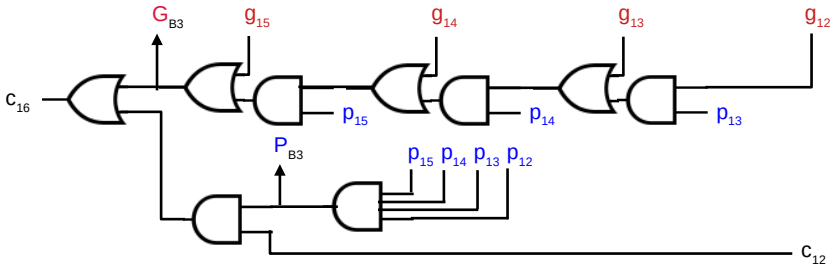
$delay(c_{12} \leftarrow g_i) \leq 7\Delta$
$delay(c_{12} \leftarrow p_i) \leq 7\Delta$
$delay(c_{12} \leftarrow c_8) = 2\Delta$

$delay(G_{B2} \leftarrow g_i) \leq 6\Delta$
$delay(G_{B2} \leftarrow p_i) \leq 6\Delta$
$delay(P_{B2} \leftarrow p_i) = 1\Delta$

Block Carry Generator B3



a block carry generator B3



$$c_{16} = g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12} + p_{15}p_{14}p_{13}p_{12}c_{12}$$

$$G_{B3} = g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12}$$
$$P_{B3} = p_{15}p_{14}p_{13}p_{12}$$

$$c_{16} = G_{B3} + P_{B3}c_{12}$$

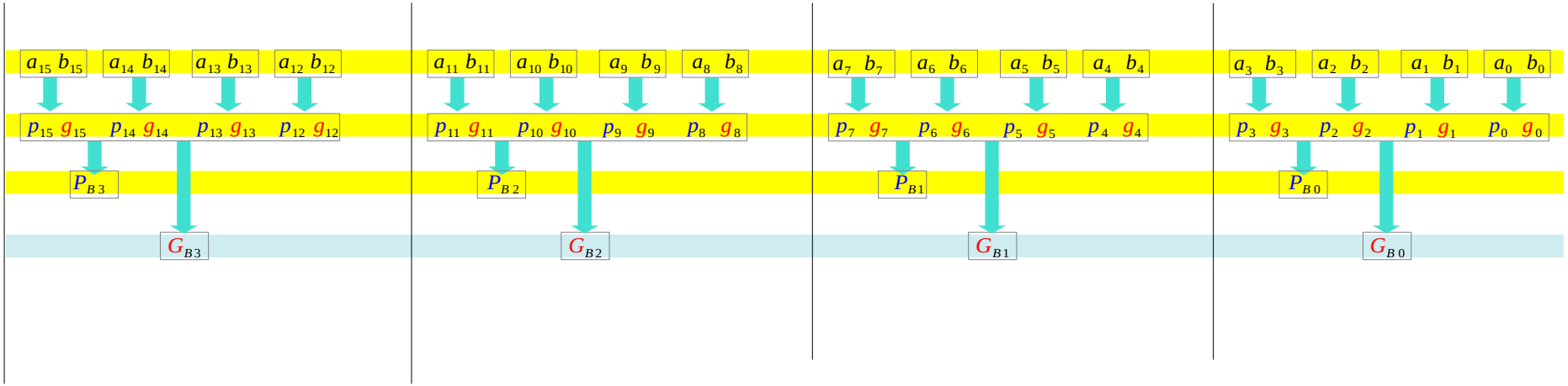
$G_{B3} \leftarrow g_{15} : 6\Delta$
$G_{B3} \leftarrow g_{14} : 5\Delta$
$G_{B3} \leftarrow g_{13} : 3\Delta$
$G_{B3} \leftarrow g_{12} : 1\Delta$

$G_{B3} \leftarrow p_{13} : 6\Delta$
$G_{B3} \leftarrow p_{14} : 4\Delta$
$G_{B3} \leftarrow p_{15} : 2\Delta$

$P_{B3} \leftarrow p_{12} : 1\Delta$
$P_{B3} \leftarrow p_{13} : 1\Delta$
$P_{B3} \leftarrow p_{14} : 1\Delta$
$P_{B3} \leftarrow p_{15} : 1\Delta$

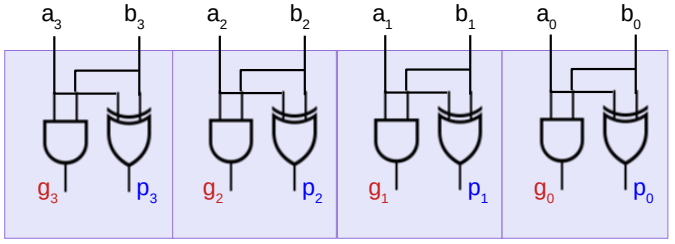
$$delay(c_{16} \leftarrow g_i) \leq 7\Delta \qquad delay(G_{B3} \leftarrow g_i) \leq 6\Delta$$
$$delay(c_{16} \leftarrow p_i) \leq 7\Delta \qquad delay(G_{B3} \leftarrow p_i) \leq 6\Delta$$
$$delay(c_{16} \leftarrow c_{12}) = 2\Delta \qquad delay(P_{B3} \leftarrow p_i) = 1\Delta$$

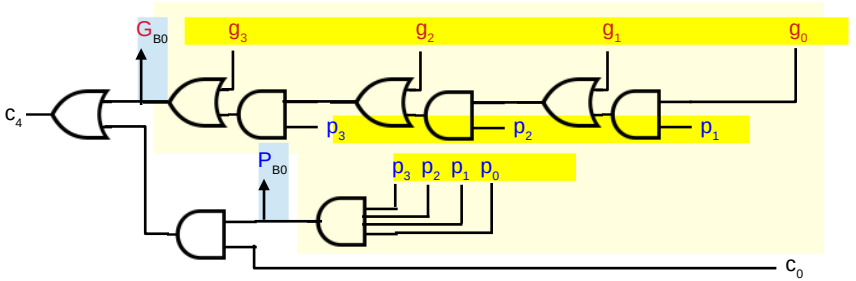
All P_{Bi}'s and all G_{Bi}'s are computed in parallel



$$
\begin{aligned}
&delay(p_i, g_i \leftarrow a_i, b_i) = 1\Delta \\
&delay(G_{Bi} \leftarrow p_i, g_i) \le 6\Delta \\
&delay(P_{Bi} \leftarrow p_i, g_i) = 1\Delta
\end{aligned}
$$

$$
\begin{aligned}
&delay(G_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) \le 7\Delta \\
&delay(P_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) = 2\Delta
\end{aligned}
$$

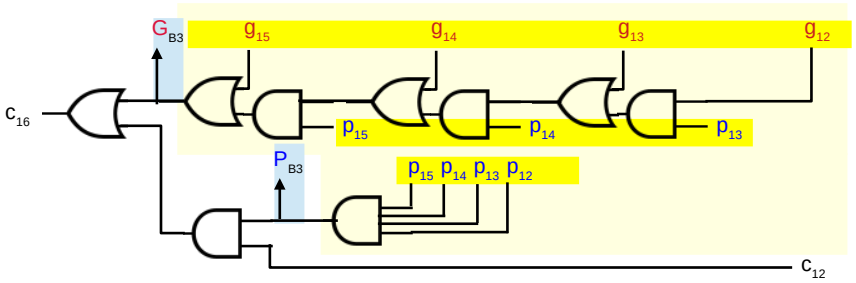




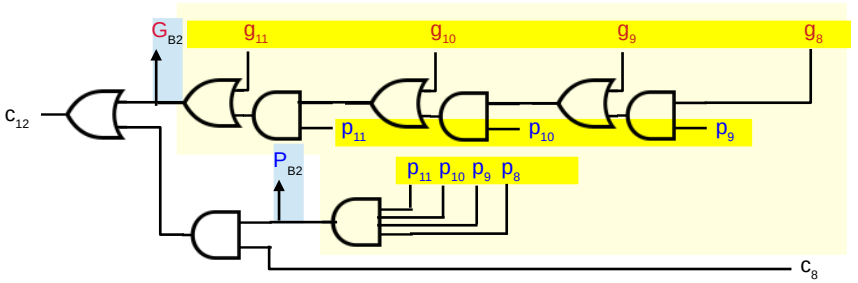
from a_i, b_i to G_{Bi}, P_{Bi} : all in (1+6) Δ

Delay to generate all G_{Bi} and P_{Bi} : (1+6)Δ

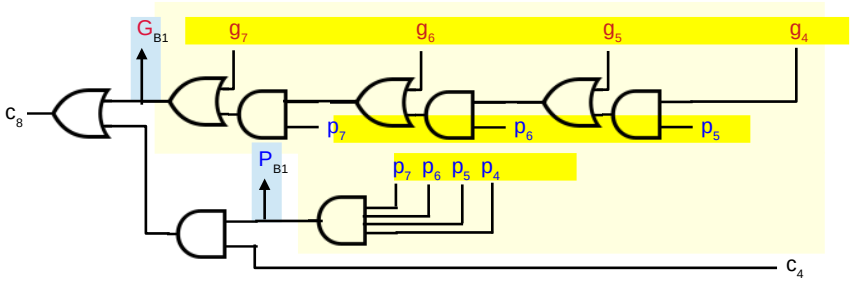
from a_i, b_i to G_{Bi}, P_{Bi} : all in (1+6) Δ



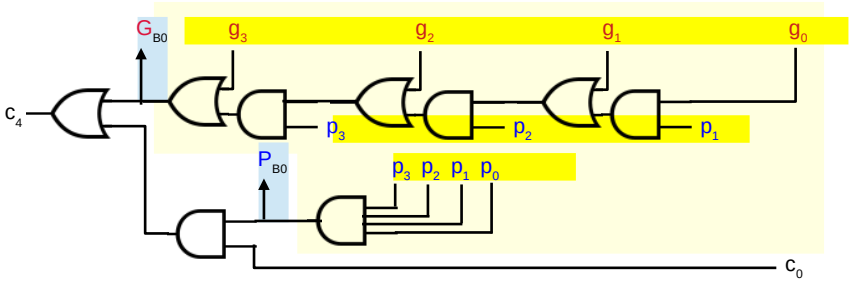
from a_i, b_i to G_{Bi}, P_{Bi} : all in (1+6) Δ



from a_i, b_i to G_{Bi}, P_{Bi} : all in (1+6) Δ



from a_i, b_i to G_{Bi}, P_{Bi} : all in (1+6) Δ



$$\begin{aligned} delay(p_i, g_i \leftarrow a_i, b_i) &= 1\Delta \\ delay(G_{Bi} \leftarrow p_i, g_i) &\le 6\Delta \\ delay(P_{Bi} \leftarrow p_i, g_i) &= 1\Delta \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} delay(G_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) &\le 7\Delta \\ delay(P_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) &= 2\Delta \end{aligned}$$

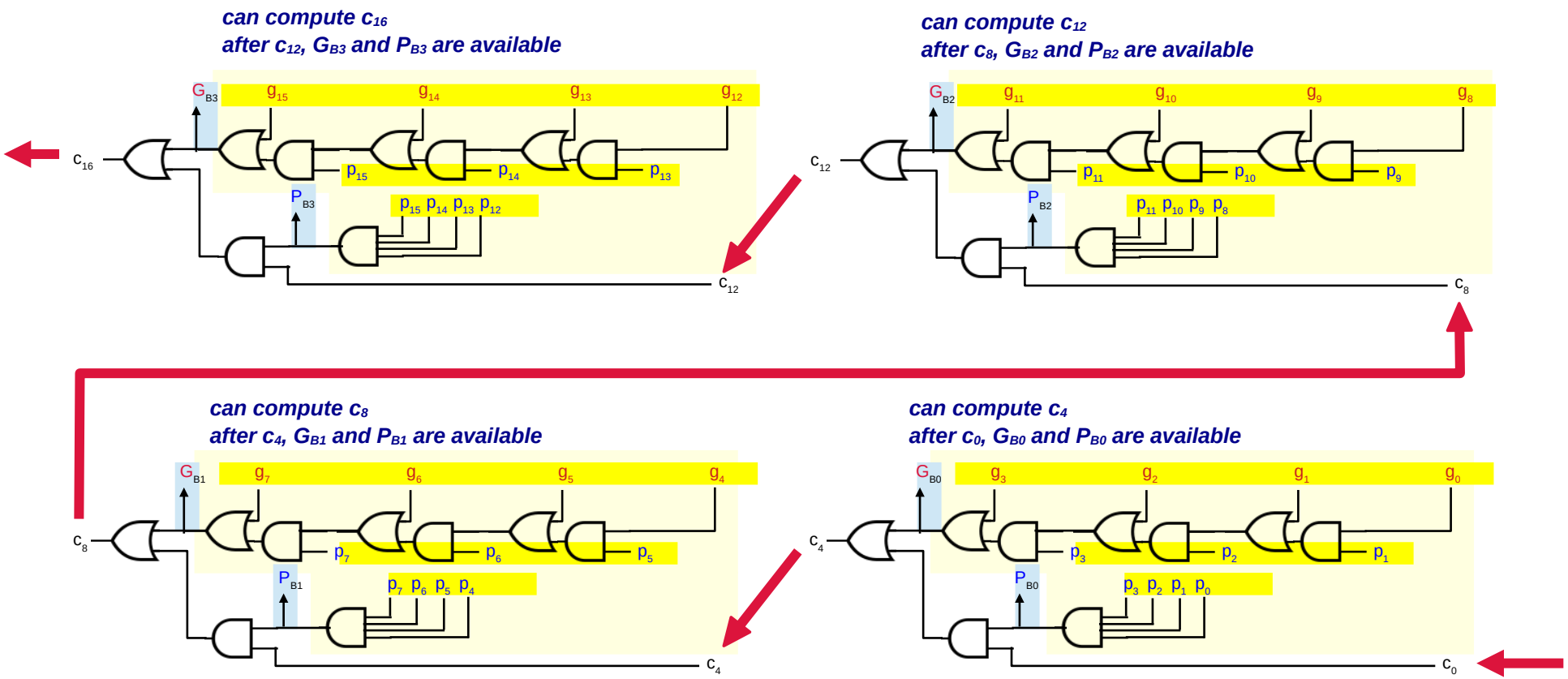
$$\begin{aligned} delay(c_o \leftarrow G_{Bi}) &= 1\Delta \\ delay(c_o \leftarrow P_{Bi}) &= 2\Delta \\ delay(c_o \leftarrow c_i) &= 2\Delta \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} delay(c_o \leftarrow G_{Bi} \leftarrow a_i, b_i) &\le 8\Delta \\ delay(c_o \leftarrow P_{Bi} \leftarrow a_i, b_i) &= 4\Delta \\ delay(c_o \leftarrow c_i) &= 2\Delta \end{aligned}$$

p_i, g_i : all computed in parallel

Block carries are computed sequentially



to compute block carries,
G_{Bi} and P_{Bi} must be available

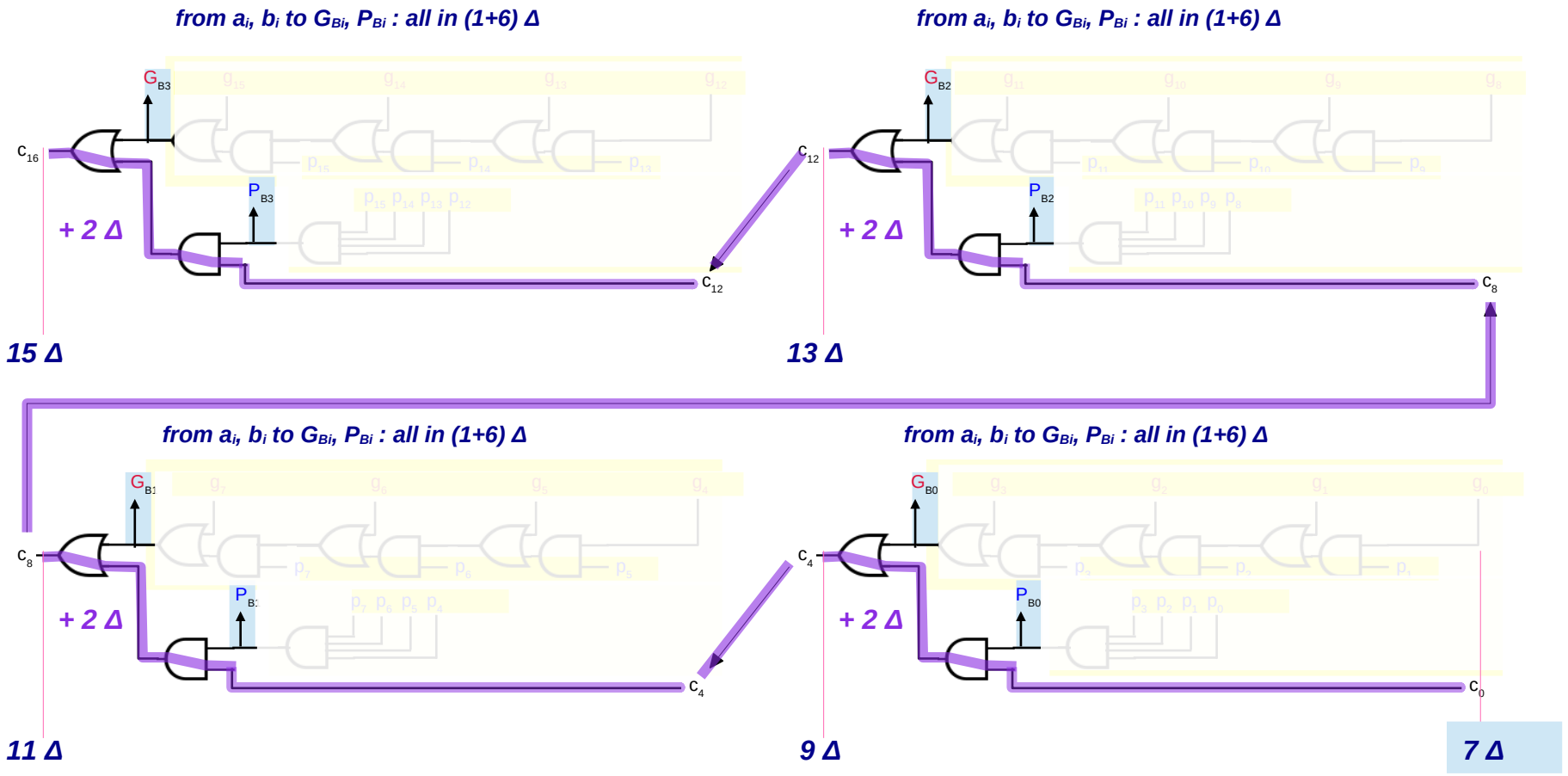
after 7Δ,
G_{Bi} and P_{Bi} must be available

Block Carry Dependencies

$c_0 \rightarrow c_4 \rightarrow c_8 \rightarrow c_{12} \rightarrow c_{16}$

$$delay\left(G_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i\right) \leq 7\Delta$$
$$delay\left(P_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i\right) = 2\Delta$$

The critical path for block carries after 7Δ



$$
\begin{aligned}
delay(c_4 \leftarrow c_0) &= 2\Delta \\
delay(c_8 \leftarrow c_4) &= 2\Delta \\
delay(c_{12} \leftarrow c_8) &= 2\Delta \\
delay(c_{16} \leftarrow c_{12}) &= 2\Delta
\end{aligned}
$$

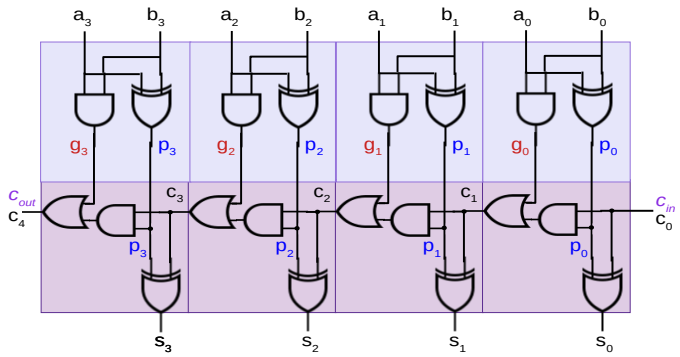
$$
\begin{aligned}
delay(c_4 \leftarrow c_0) &= 2\Delta \rightarrow 9\Delta \\
delay(c_8 \leftarrow c_0) &= 4\Delta \rightarrow 11\Delta \\
delay(c_{12} \leftarrow c_0) &= 6\Delta \rightarrow 13\Delta \\
delay(c_{16} \leftarrow c_0) &= 8\Delta \rightarrow 15\Delta
\end{aligned}
$$

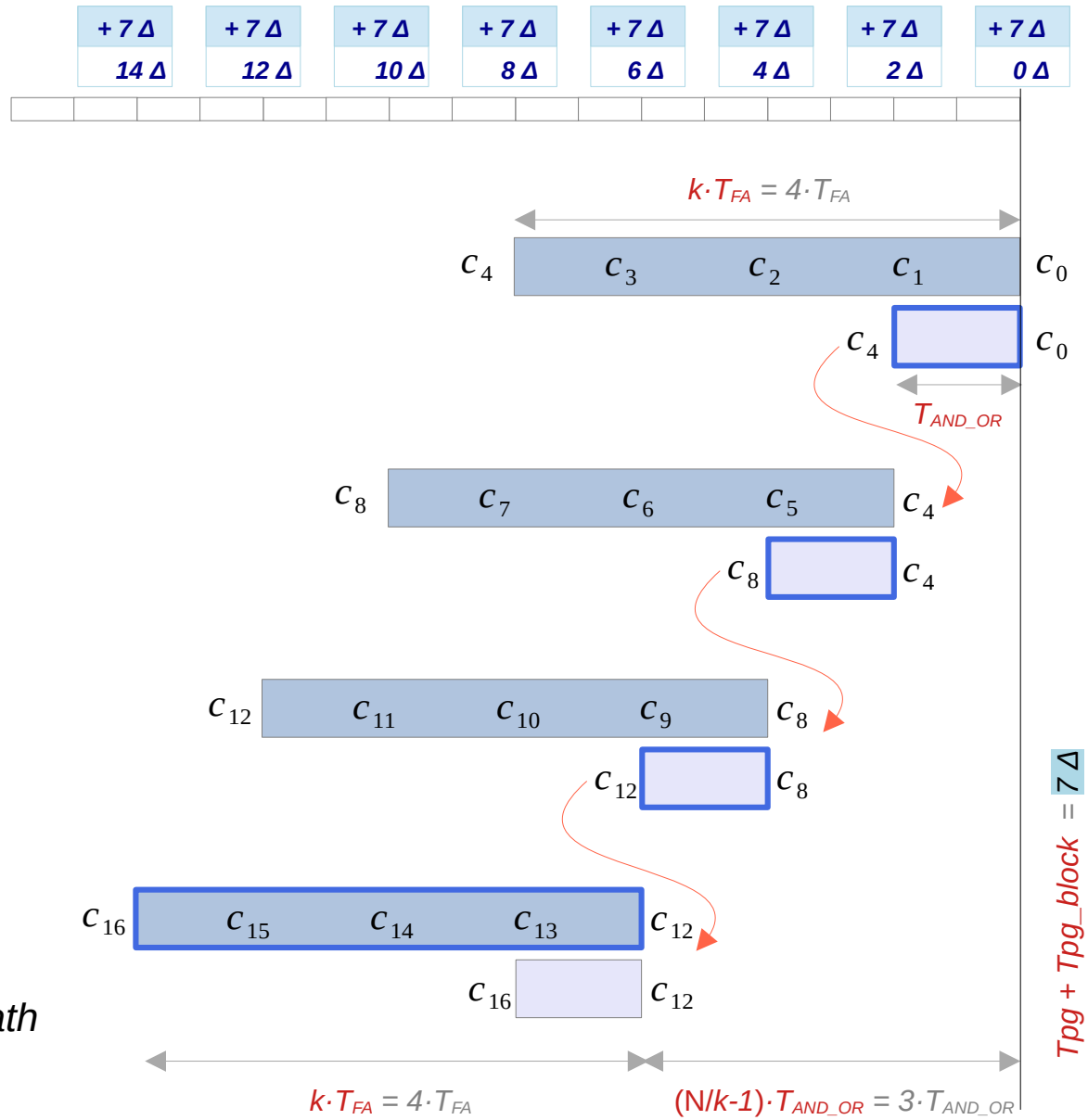
+ 7 Δ

after 7Δ, G_{Bi} and P_{Bi} are available

$$
\begin{aligned}
delay(G_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) &\le 7\Delta \\
delay(P_{Bi} \leftarrow p_i, g_i \leftarrow a_i, b_i) &= 2\Delta
\end{aligned}
$$

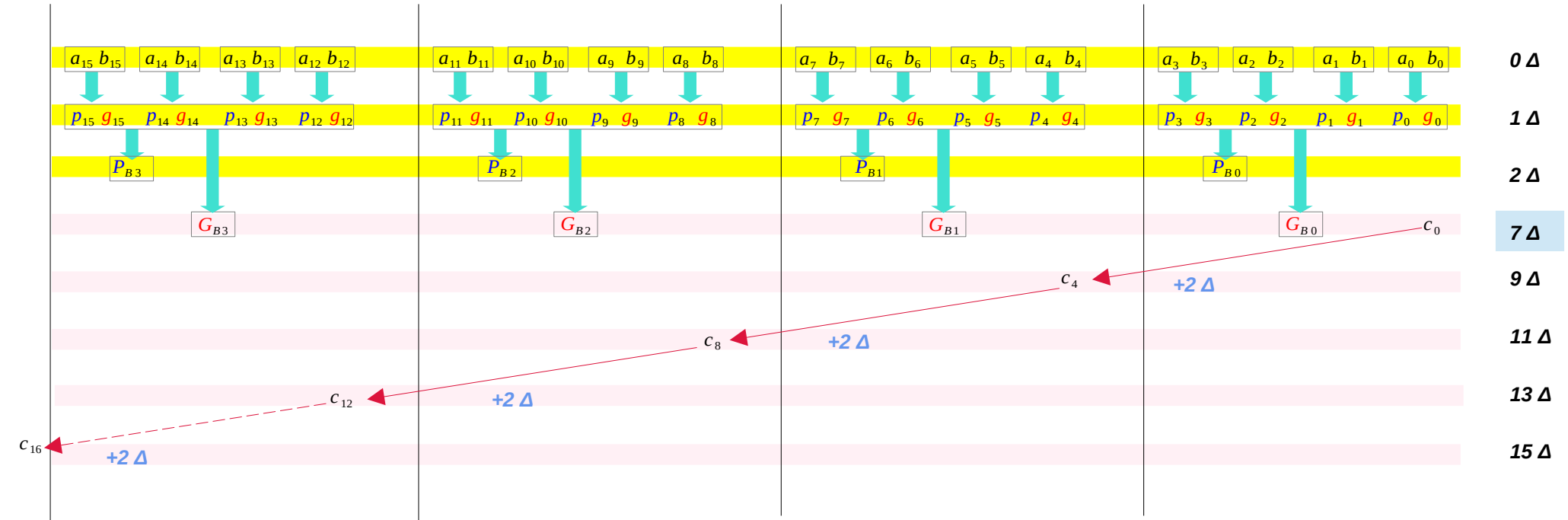
Parallel computations for each block carry input





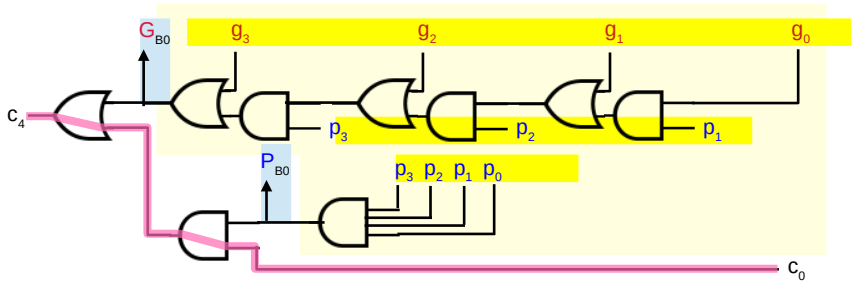
the critical path =
three carry lookahead paths +
the last ripple carry adder block path

Delays to generate all block carries c_4, c_8, c_{12}, c_{16}

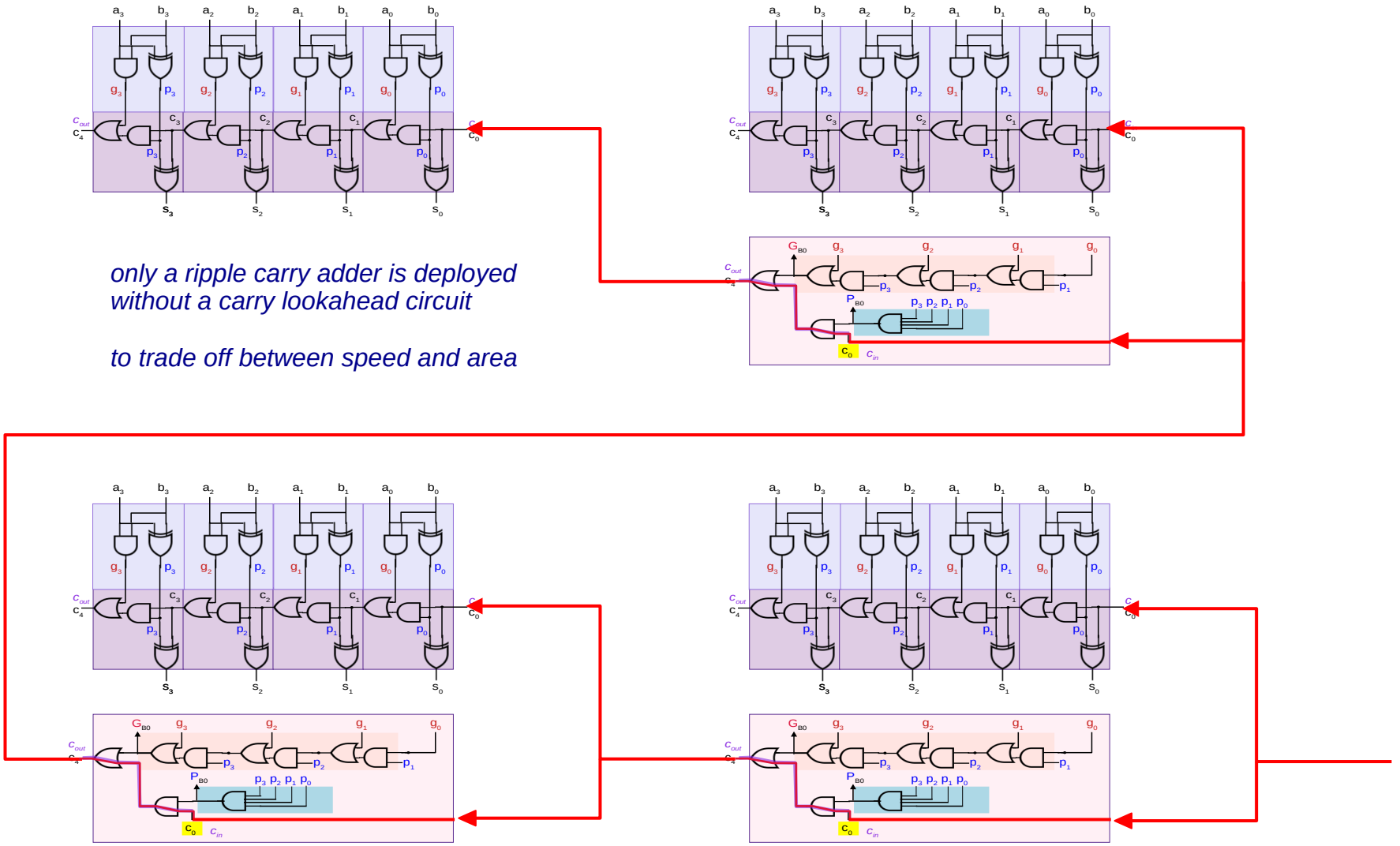


last block carry generator is not be necessary

Delay from $c_{4\cdot(i-1)}$ to $c_{4\cdot i}$: 2Δ, after 7Δ



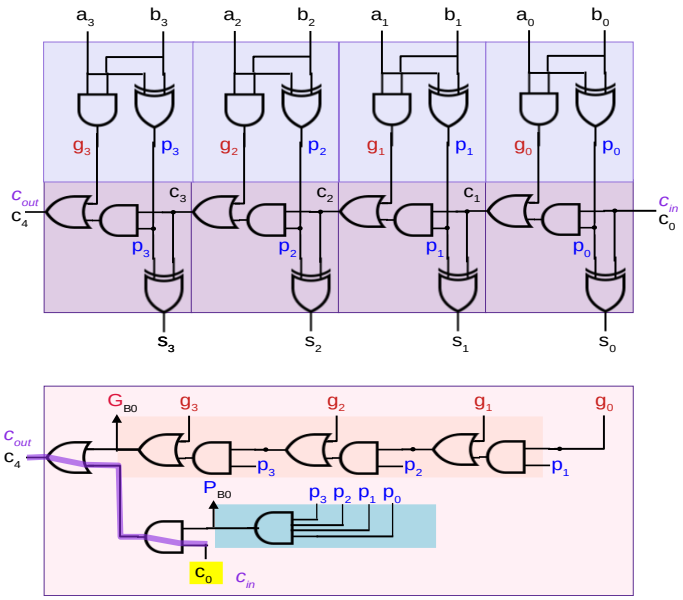
The last block carry generator is not needed

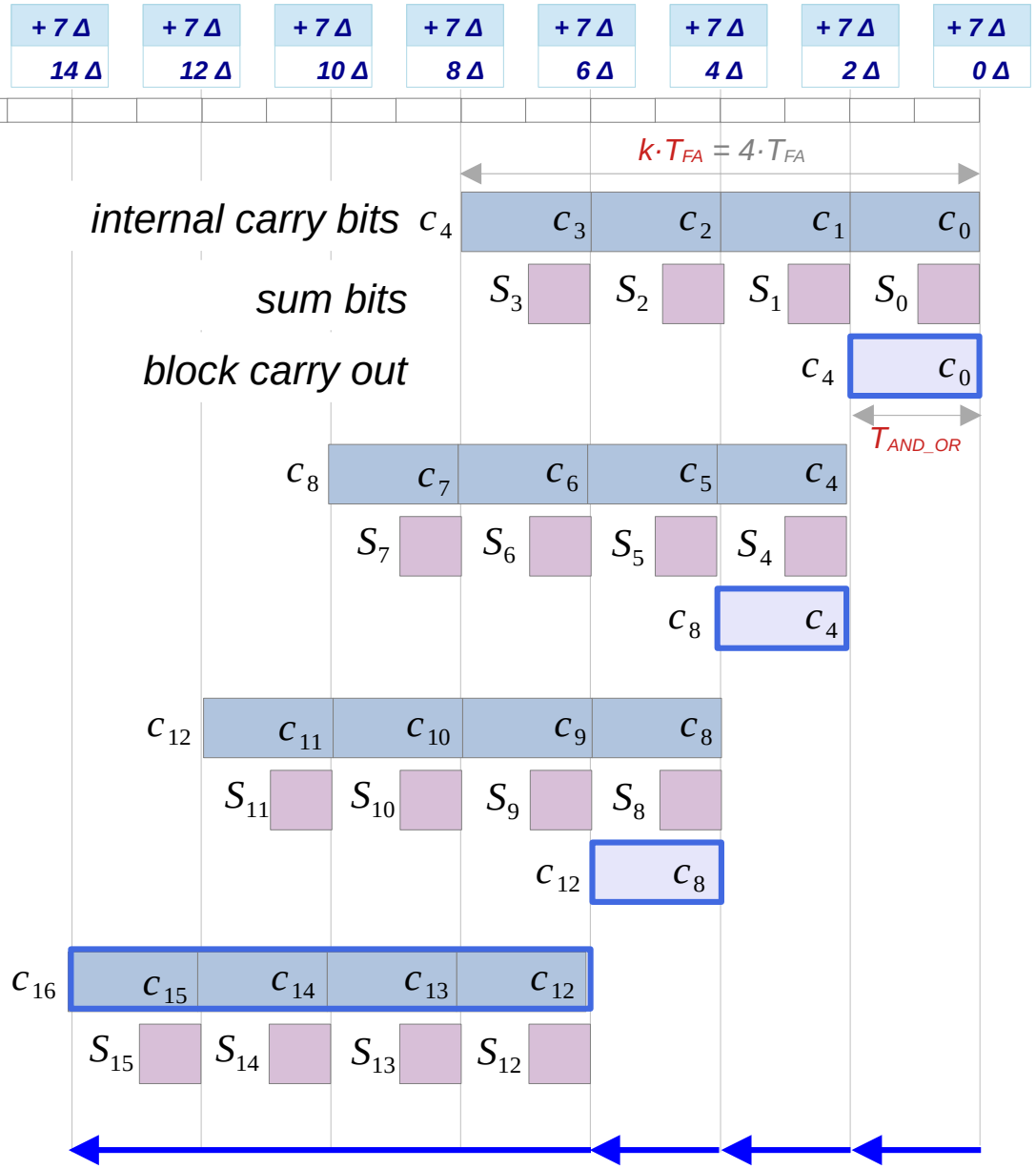


only a ripple carry adder is deployed without a carry lookahead circuit

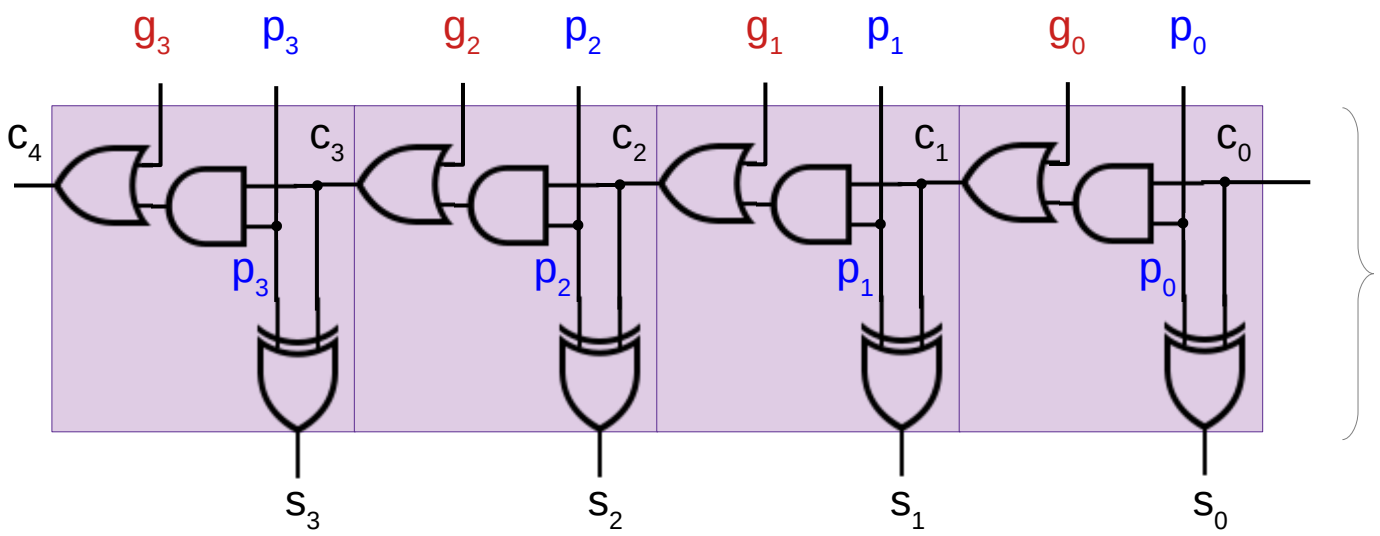
to trade off between speed and area

Computing S_{i+3}, S_{i+2}, S_{i+1}, S_i for a block carry input c_i





Delays to sum bits S_0, S_1, S_2, S_3 from block carry input c_0



serial computations of c_i's and S_i's

$$\begin{aligned} delay(S_0 \leftarrow c_0) &= 1\Delta \\ delay(S_1 \leftarrow c_1) &= 1\Delta \\ delay(S_2 \leftarrow c_2) &= 1\Delta \\ delay(S_3 \leftarrow c_3) &= 1\Delta \end{aligned}$$

from internal carries to sum bits

$+$

$$\begin{aligned} delay(c_1 \leftarrow c_0) &= 2\Delta \\ delay(c_2 \leftarrow c_1) &= 2\Delta \\ delay(c_3 \leftarrow c_2) &= 2\Delta \end{aligned}$$

from block carry inputs to internal carries

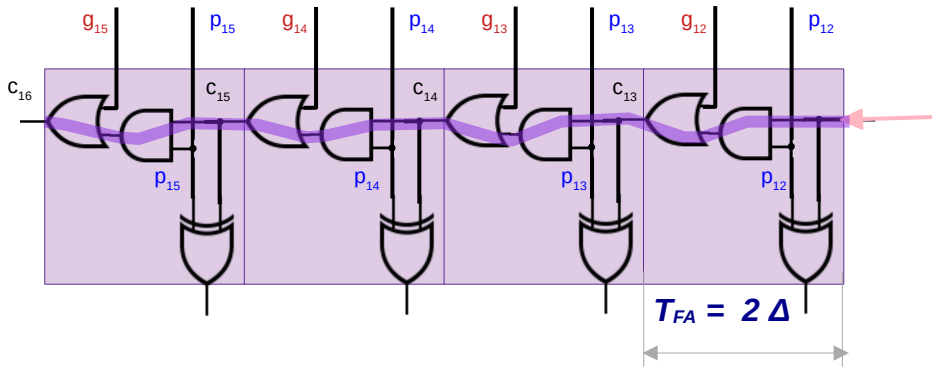
$\rightarrow$

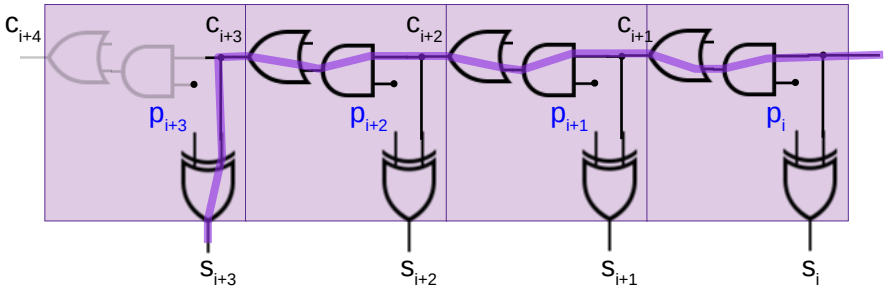
$$\begin{aligned} delay(S_0 \leftarrow c_0) &= 1\Delta \\ delay(S_1 \leftarrow c_0) &= 3\Delta \\ delay(S_2 \leftarrow c_0) &= 5\Delta \\ delay(S_3 \leftarrow c_0) &= 7\Delta \end{aligned}$$

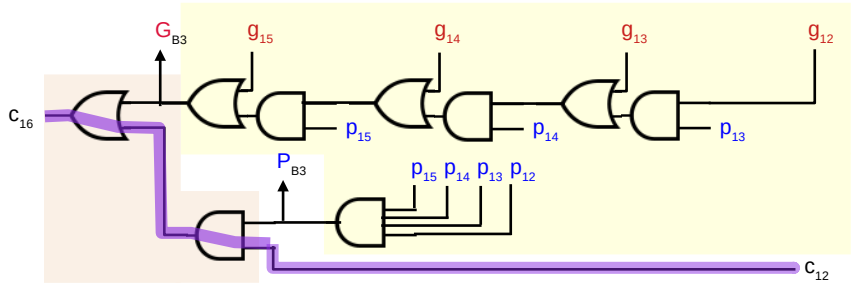
from block carry inputs to sum bits

https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

Delays for the last block







$$delay\left(c_{16} \leftarrow c_{12}\right) = kT_{FA} = 4 \cdot 2 \cdot \Delta$$

kT_{FA} = the carry path delay
in a k-bit ripple carry adder block

$$delay\left(S_{15} \leftarrow c_{12}\right) = kT_{FA} - 1 = 7 \cdot \Delta$$

Unless multi-level carry lookahead methods are used,
in the last block, only a ripple carry adder is deployed
without a carry lookahead circuit

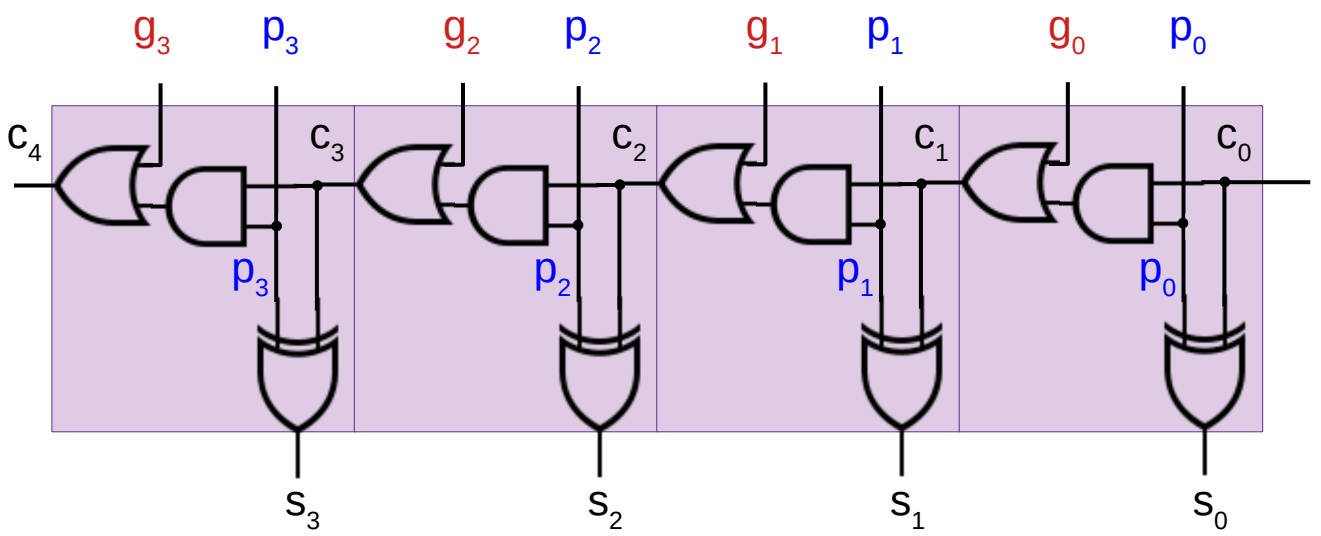
to trade off between speed and area

carries c_{15}, c_{14}, c_{13} needs to be computed
for sum bits S_{15}, S_{14}, S_{13}, S_{12}

the carry lookahed circuit does not help much
in the last block

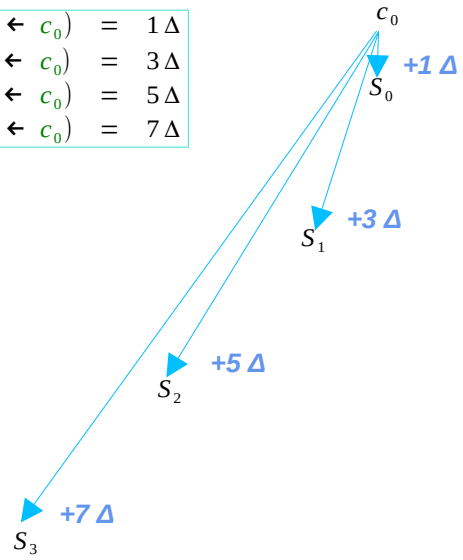
$$delay\left(c_{16} \leftarrow c_{12}\right) = T_{ANDOR} = 2 \cdot \Delta$$

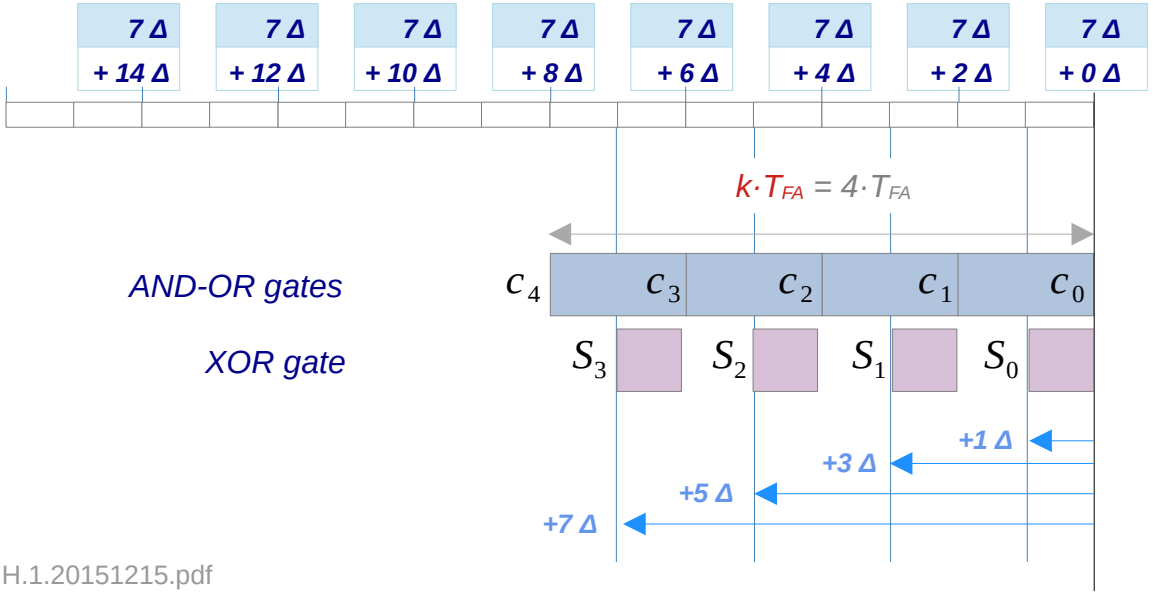
Delays to sum bits and internal carries



serial computations of c_i's and S_i's

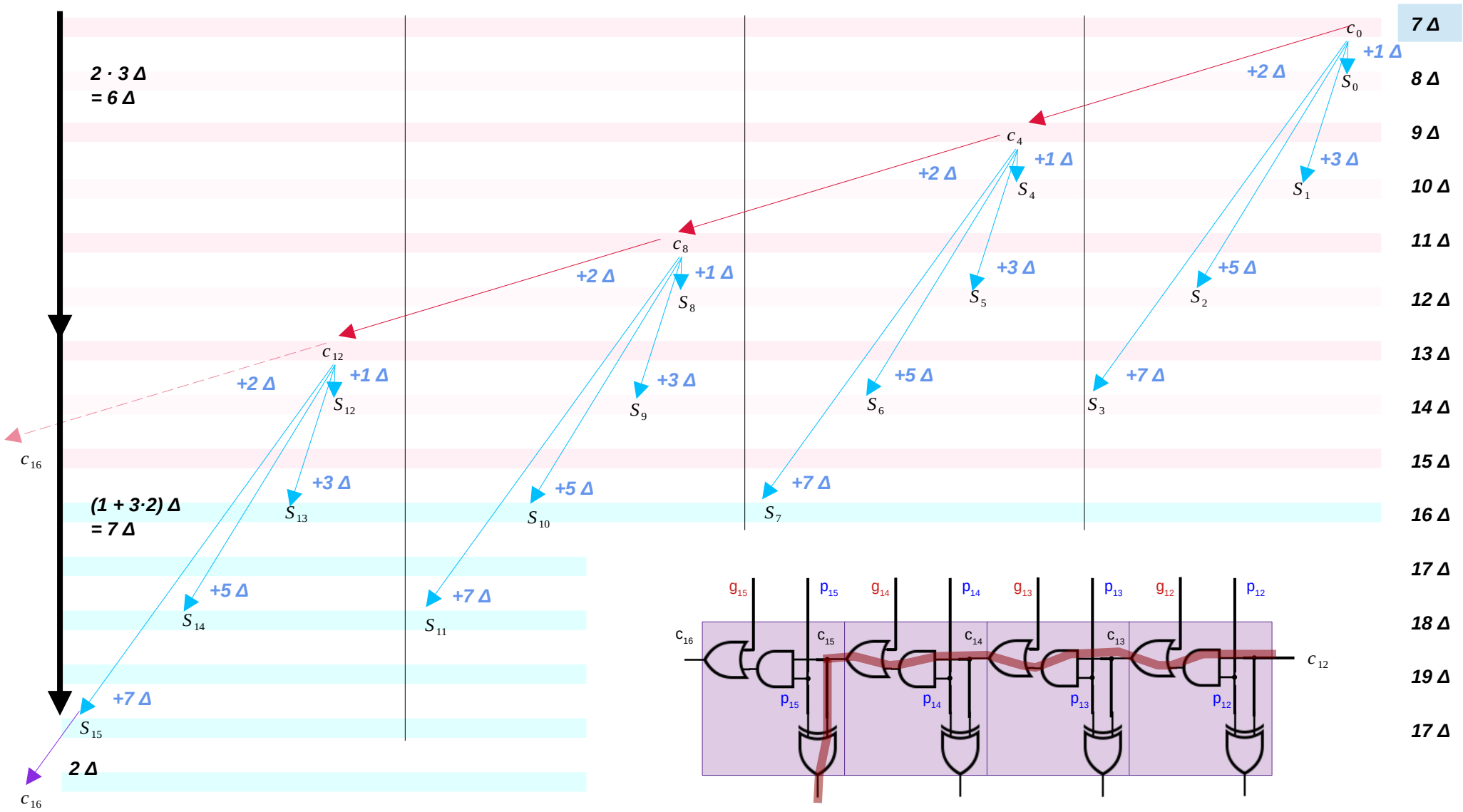
$delay(S_0 \leftarrow c_0)$	=	1 Δ
$delay(S_1 \leftarrow c_0)$	=	3 Δ
$delay(S_2 \leftarrow c_0)$	=	5 Δ
$delay(S_3 \leftarrow c_0)$	=	7 Δ





https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf

Delays to sum bits from block carry inputs



T_{pg}, T_{pg_block}

T_{pg} : path delay from a_i, b_i to p_i, g_i 1Δ

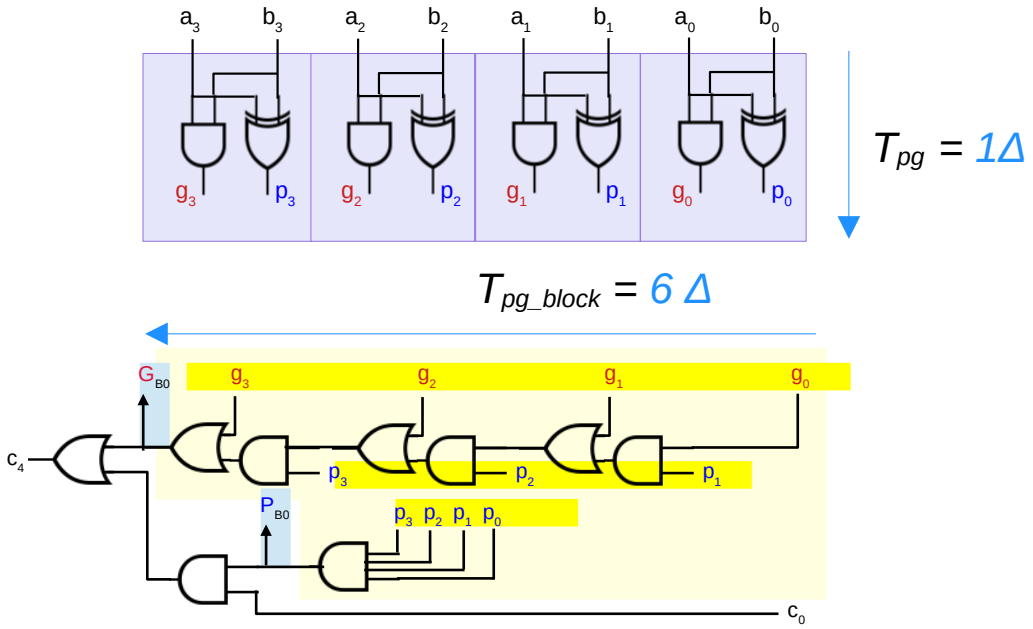
T_{pg_block} : path delay from p_i, g_i to G_{Bi}, P_{Bi} 6Δ

$T_{pg} + T_{pg\_block}$: path delay from a_i, b_i to G_{Bi}, P_{Bi} 7Δ

after $T_{pg} + T_{pg\_block}$, all G_{bi}'s , P_{bi}'s are available.

But each carry lookahead circuit can compute its own c_{out}, only after c_{in} is ready.

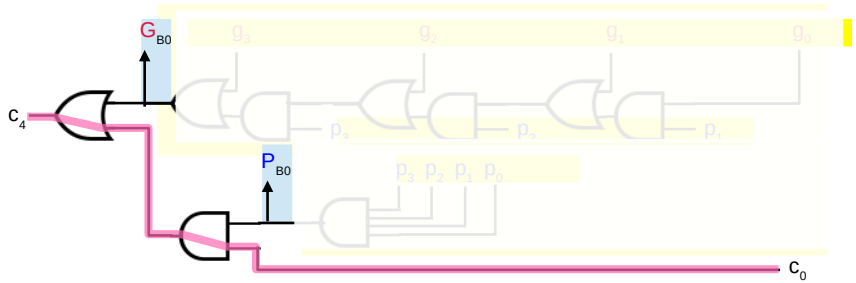
Thus, $c_0 \rightarrow c_4 \rightarrow c_8 \rightarrow c_{12}$ forms a critical path, after all G_{Bi} and P_{bi} are computed (7Δ)



	$c_{in} \rightarrow c_{out}$	Lookahead circuits		Ripple Carry Adder
in the 1st block,	$c_0 \rightarrow c_4$	$T_{AND\_OR} = 2\Delta$	$c_0 \rightarrow c_4$	$k \cdot T_{FA} = 4 \cdot 2\,\Delta = 8\Delta$
in the 2nd block,	$c_4 \rightarrow c_8$	$T_{AND\_OR} = 2\Delta$	$c_4 \rightarrow c_8$	$k \cdot T_{FA} = 4 \cdot 2\,\Delta = 8\Delta$
in the 3rd block,	$c_8 \rightarrow c_{12}$	$T_{AND\_OR} = 2\Delta$	$c_8 \rightarrow c_{12}$	$k \cdot T_{FA} = 4 \cdot 2\,\Delta = 8\Delta$
in the 4th block,	$c_{12} \rightarrow c_{16}$	$T_{AND\_OR} = 2\Delta$	$c_{12} \rightarrow c_{15}$	$k \cdot T_{FA} = 4 \cdot 2\,\Delta = 8\Delta$

	$c_o \rightarrow c_{out}$	Lookahead circuits	$c_o \rightarrow c_{out}$	Ripple Carry Adder
in the 1st block,	$c_0 \rightarrow c_4$	$1 \cdot T_{AND\_OR} = 2\Delta$	$c_0 \rightarrow c_4$	$1 \cdot k \cdot T_{FA} = 1 \cdot 4 \cdot 2\,\Delta = 8\Delta$
in the 2nd block,	$c_0 \rightarrow c_8$	$2 \cdot T_{AND\_OR} = 4\Delta$	$c_0 \rightarrow c_8$	$2 \cdot k \cdot T_{FA} = 2 \cdot 4 \cdot 2\,\Delta = 16\Delta$
in the 3rd block,	$c_0 \rightarrow c_{12}$	$3 \cdot T_{AND\_OR} = 6\Delta$	$c_0 \rightarrow c_{12}$	$3 \cdot k \cdot T_{FA} = 3 \cdot 4 \cdot 2\,\Delta = 24\Delta$
in the 4th block,	$c_0 \rightarrow c_{16}$	$4 \cdot T_{AND\_OR} = 8\Delta$	$c_0 \rightarrow c_{15}$	$4 \cdot k \cdot T_{FA} = 4 \cdot 4 \cdot 2\,\Delta = 32\Delta$

from $c_{4\cdot(i-1)}$ to $c_{4\cdot i}$: 2 Δ



$T_{AND\_OR} = 2\Delta$

Delays for each block carry input c_0, c_4, c_8, c_{12}

While carry lookahead operation is performed,
the sum bits are computed by a ripple carry adder

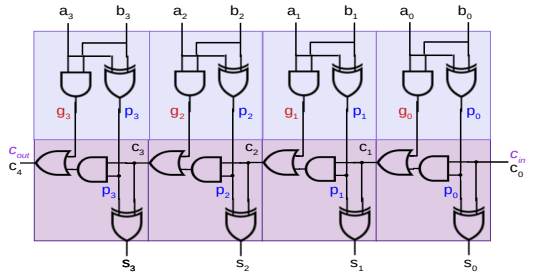
when c_0 *is available after* $T_{pg} + T_{pg\_block} + 0 \cdot T_{AND\_OR}$
the 1st block's ripple carry adder and
the 1st blocks carry lookahead circuits can compute in parallel

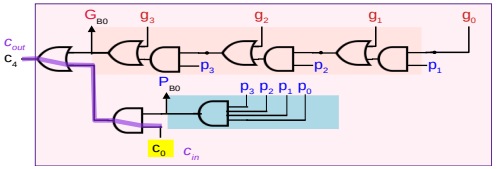
when c_4 *is available after* $T_{pg} + T_{pg\_block} + 1 \cdot T_{AND\_OR}$
the 2nd block's ripple carry adder and
the 2nd blocks carry lookahead circuits can compute in parallel

when c_8 *is available after* $T_{pg} + T_{pg\_block} + 2 \cdot T_{AND\_OR}$
the 3rd block's ripple carry adder and
the 3rd blocks carry lookahead circuits can compute in parallel

when c_{12} *is available after* $T_{pg} + T_{pg\_block} + 3 \cdot T_{AND\_OR}$
the 4th block's ripple carry adder and
the 4th blocks carry lookahead circuits can compute in parallel

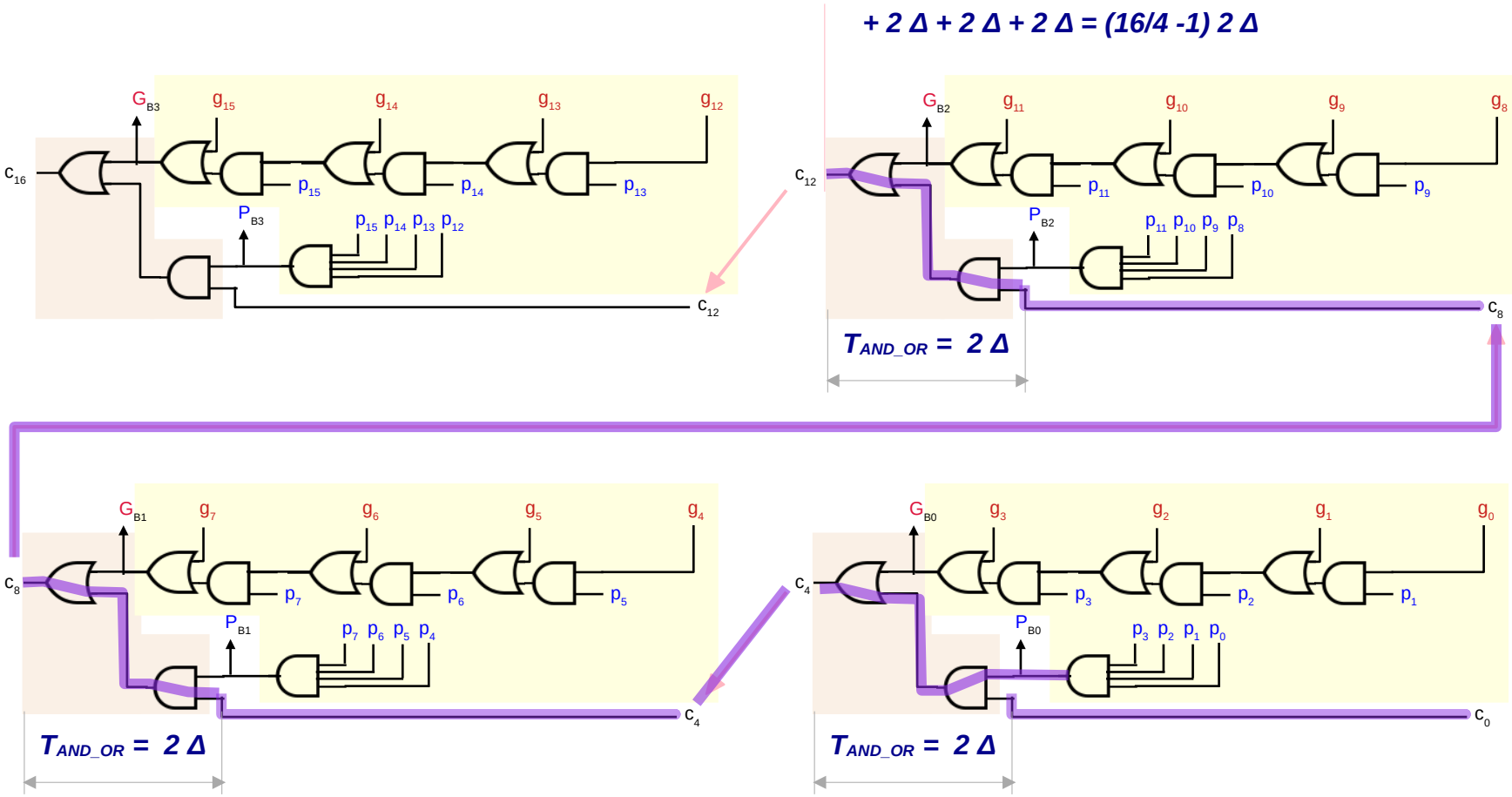
a ripple carry adder





a carry lookahead circuit

$T_{AND\_OR}*(N/k-1)$



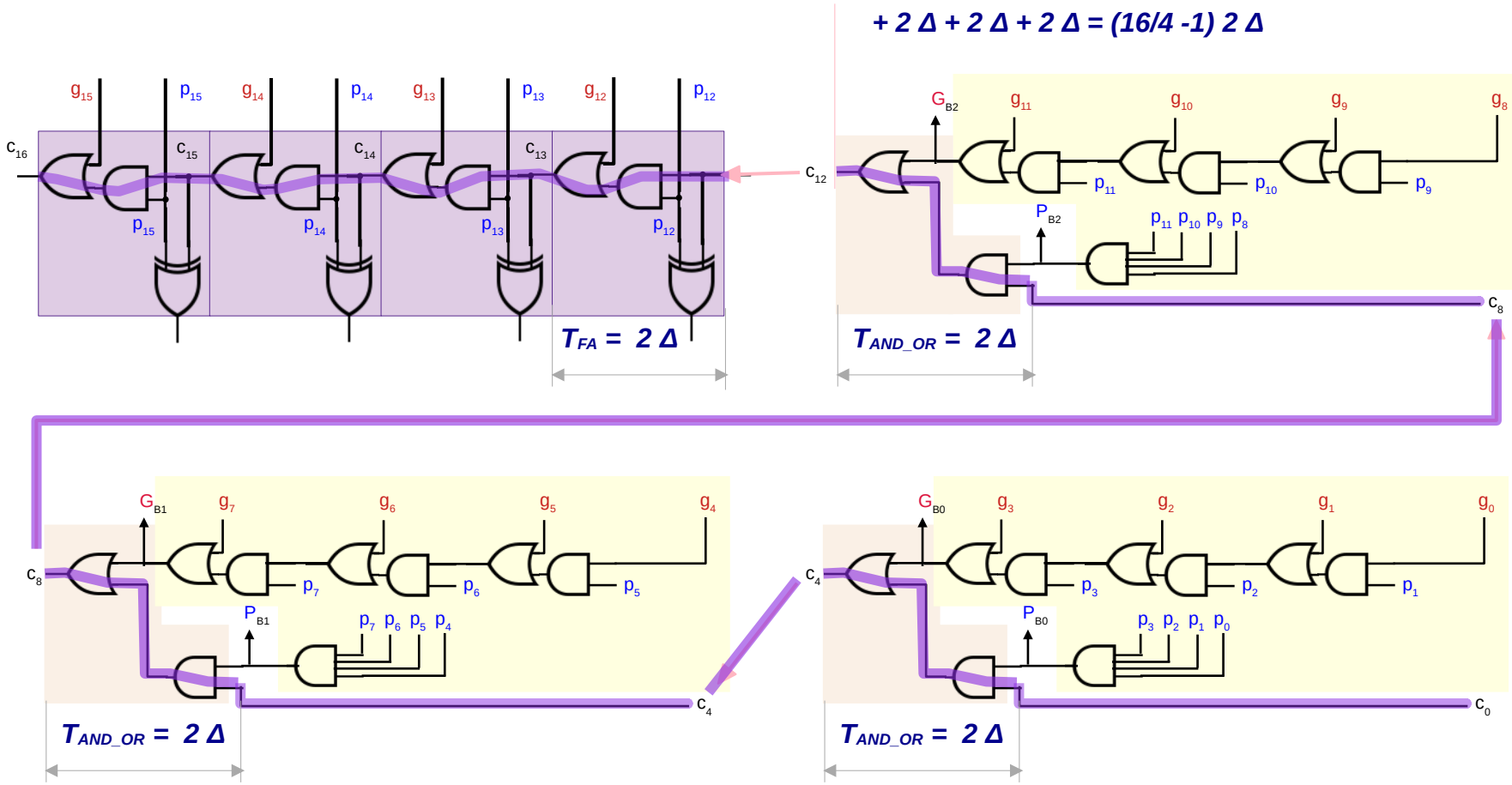
$$delay(c_{4(i+1)} \leftarrow G_{Bi}) = 1\Delta$$
$$delay(c_{4(i+1)} \leftarrow B_{Bi}) = 2\Delta$$
$$delay(c_{4(i+1)} \leftarrow c_{4i}) = 2\Delta$$

$$delay(G_{Bi} \leftarrow a_i, b_i) \leq 7\Delta$$
$$delay(P_{Bi} \leftarrow a_i, b_i) = 2\Delta$$

$$delay(c_{4(i+1)} \leftarrow a_i, b_i) \leq 8\Delta$$
$$delay(c_{4(i+1)} \leftarrow a_i, b_i) = 4\Delta$$
$$delay(c_{4(i+1)} \leftarrow c_{4i}) = 2\Delta$$

computed in serial

$kT_{FA} + T_{AND\_OR}*(N/k-1)$



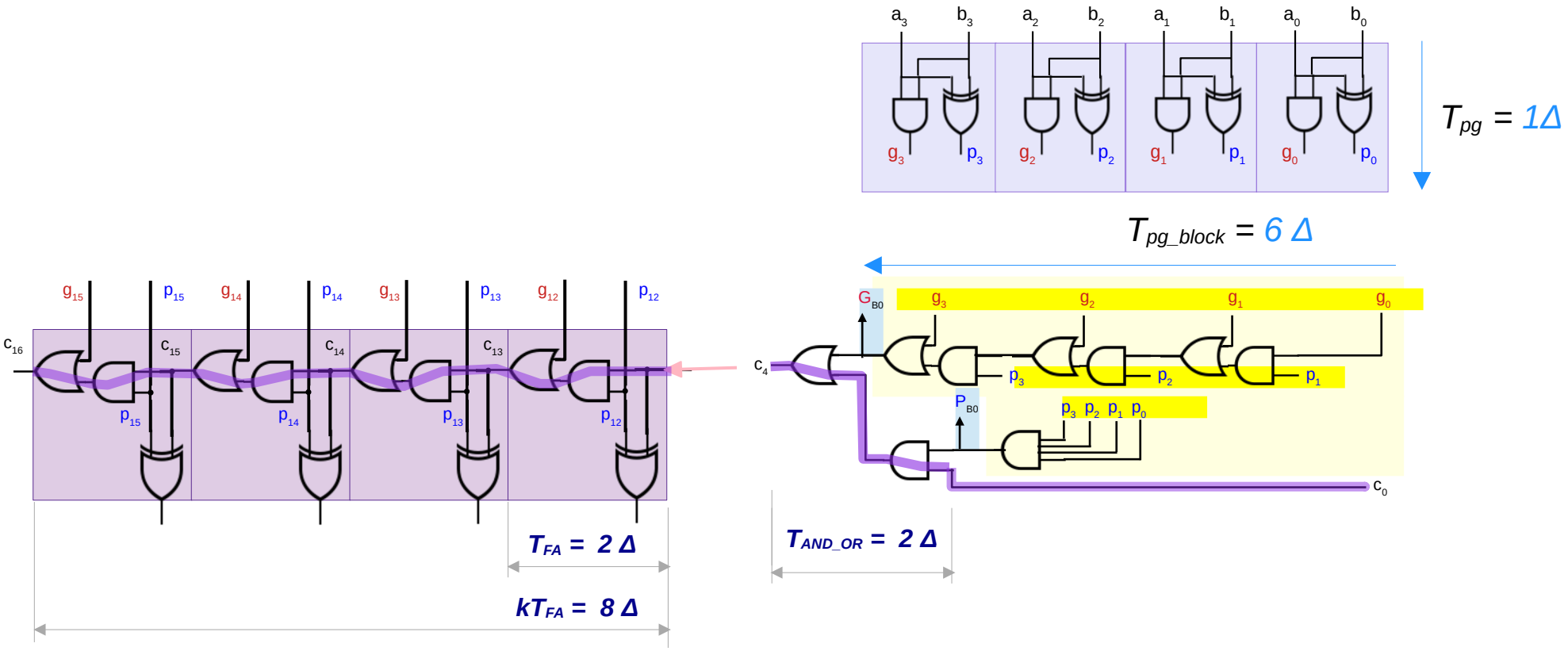
$$delay(c_{4(i+1)} \leftarrow G_{Bi}) = 1\Delta$$
$$delay(c_{4(i+1)} \leftarrow B_{Bi}) = 2\Delta$$
$$delay(c_{4(i+1)} \leftarrow c_{4i}) = 2\Delta$$

$$delay(G_{Bi} \leftarrow a_i, b_i) \leq 7\Delta$$
$$delay(P_{Bi} \leftarrow a_i, b_i) = 2\Delta$$

$$delay(c_{4(i+1)} \leftarrow a_i, b_i) \leq 8\Delta$$
$$delay(c_{4(i+1)} \leftarrow a_i, b_i) = 4\Delta$$
$$delay(c_{4(i+1)} \leftarrow c_{4i}) = 2\Delta$$

computed in serial

$T_{pg} + T_{pg\_block} + T_{AND\_OR}*(N/k-1) + kT_{FA}$



Multi-input vs. multiple 2-input gates (1)

multi-input NAND gates vs. multiple 2-input NAND

a trade-off between transistor count, area, and speed,
as well as design library availability.

While a multi-input NAND gate uses

- *+ fewer transistors*
  (N PMOS and N NMOS for an N-input gate)
- *+ a simpler structure,*
- *– slower fall times and*
- *– larger overall transistor sizes.*

Conversely, using multiple 2-input NAND gates

- *– slower overall speed and*
- *– increased area.*

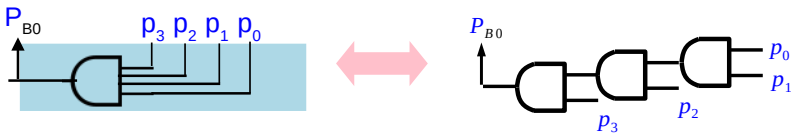
Google AI Overview : vlsi 2-input nand vs multi-input nand

Multi-input vs. multiple 2-input gates (2)

***Structure**:*
An N-input NAND gate requires
N parallel PMOS transistors
for the pull-up network and
N series NMOS transistors
for the pull-down network.

Benefits:

Reduced Transistors: Uses fewer transistors compared to cascading 2-input NANDs for the same logic function.



Drawbacks:

Slower Fall Time:
increasing pull-down resistance with more inputs, slowing down the output's transition to a LOW state.

Larger Transistor Sizes:
wider PMOS transistors to maintain drive strength, which can increase the physical area of the gate.

Library Limitations: Standard VLSI cell libraries may not provide multi-input NAND gates with fan-ins greater than 4 or 5, limiting their practical use

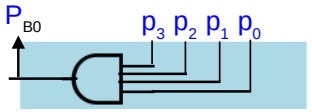
when compared to 2-input NAND gate

Google AI Overview : vlsi 2-input nand vs multi-input nand

Multi-input vs. multiple 2-input gates (3)

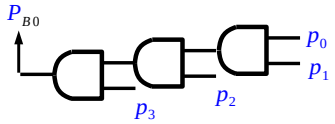
***Multi-input NAND**:*
where area and transistor count are critical, and where the performance impact of slower fall times is acceptable.

One 4-input NAND:
4 p-type transisters
4 n-type transistors
Total 8 transistors



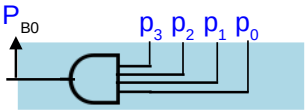
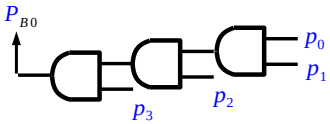
***Multiple 2-input NANDs**:*
when speed (fast fall time) is a priority, or when designing with standard cell libraries that lack large fan-in gates.

Three 2-input NAND
*3 * 2 p-type transistors*
*3 * 2 n-type transistors*
Total 12 transistors



Google AI Overview : vlsi 2-input nand vs multi-input nand

Multi-input vs. multiple 2-input gates (4)

	multi-input		***2-input***
		⟺	
area	A_m	>	A_2
fall time	Δ_m	>	Δ_2
		⟺	
area	A_m	<	$3 \cdot A_2$
propagation delay	Δ_m	<	$3 \cdot \Delta_2$

larger fall time per gate,
but smaller propagation delay in total
similarly, larger area per gate,
but smaller area in total

Google AI Overview : vlsi 2-input nand vs multi-input nand

large size *small size*

Unequal rise and fall times (1)

interconnect mismatches and unbalanced drivers
→ unequal rise and fall times,
→ signal integrity issues

- *glitches*
- *delay faults*
- *increased power dissipation*
- *logic malfunctions*
- *reduced circuit reliability.*

These asymmetries create
unbalanced parasitic effects,
causing signals to propagate at different speeds
and potentially corrupting the intended logic,
especially in high-frequency, high-density chips.

Google AI Overview : vlsi unequal fall and rise times cause serious problems or not

Unequal rise and fall times (2)

Crosstalk and Noise:
unbalanced drivers and
different interconnect lengths
→ unequal parasitic capacitance and inductance
→ increased crosstalk.
→ unintended voltage pulses (glitches)
or signal delays.

Signal Delay Faults:
unequal transition times
→ crosstalk and signal skew
→ delay faults,
signals arrive at their destination
later than expected
→ particularly detrimental
in high-speed circuits,

Logic Malfunctions:
The introduced noise pulses or delays
→ can flip the logic value
→ incorrect outputs
→ logic errors.

Increased Power Consumption:
mismatched transition times,
during parallel or in-phase switching
→ affect how parasitics are charged / discharged
→ increased dynamic power dissipation

Google AI Overview : vlsi unequal fall and rise times cause serious problems or not

Unequal rise and fall times (3)

Why it's a growing problem:

***High-Density and High-Frequency Designs**:*
as VLSI technology scales down,
chip density increases,
operating frequencies rise.
→ exacerbates parasitic coupling

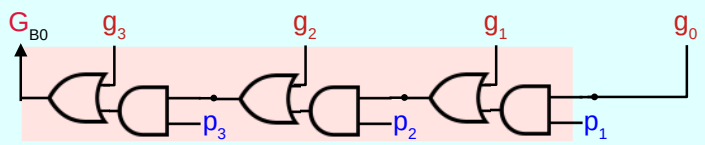
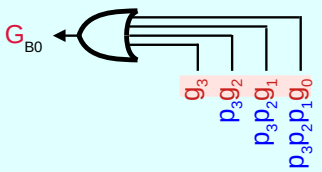
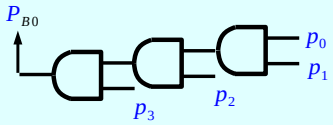
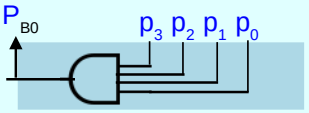
***Process Variations**:*
Random variations in semiconductor fabrication
→ differences in interconnect lengths
and driver strengths
→ unequal rise and fall times.

To mitigate these issues, designers employ techniques to balance transition times, control impedance, and reduce crosstalk, ensuring reliable operation in modern VLSI circuits.

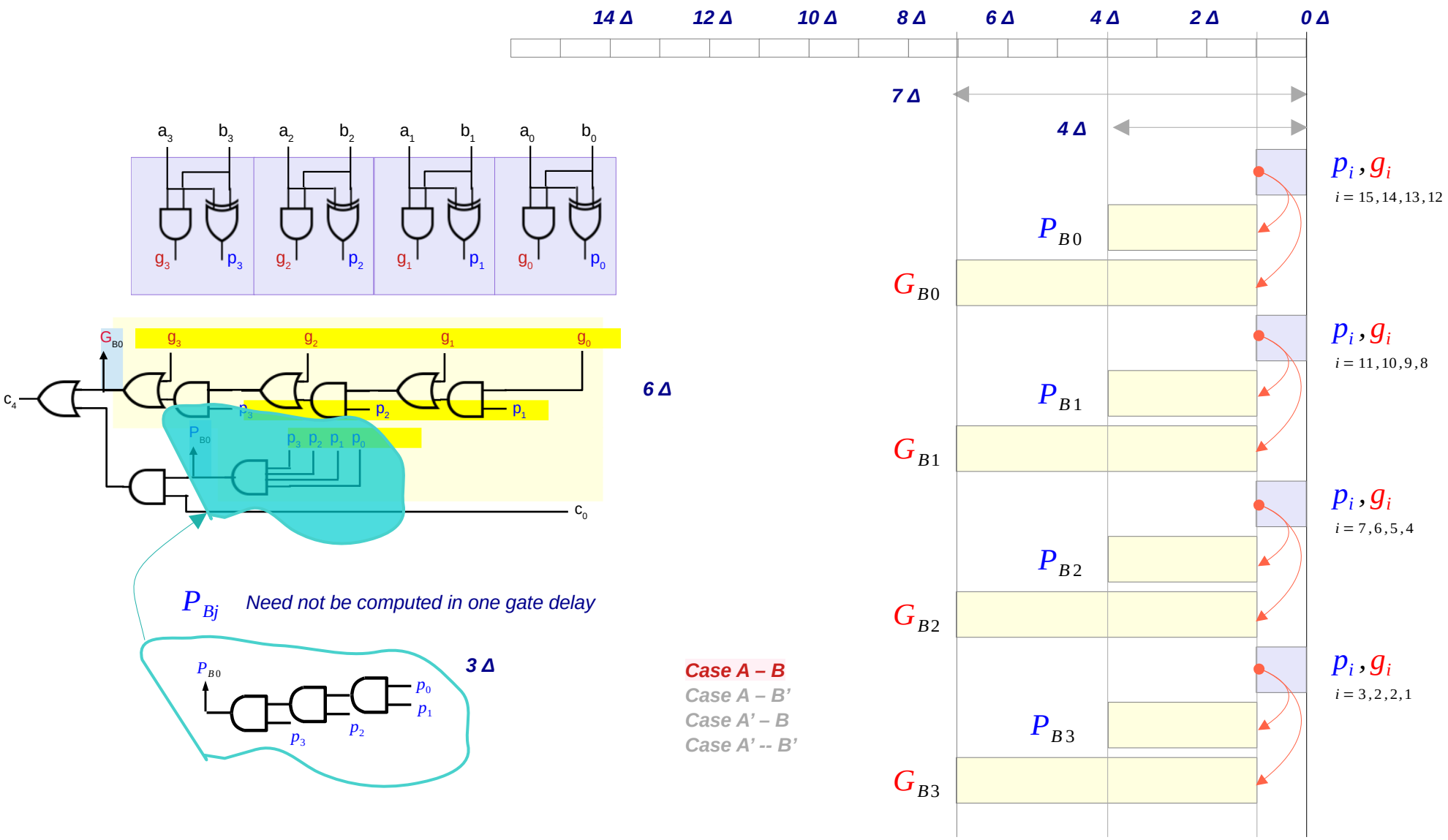
Google AI Overview : vlsi unequal fall and rise times cause serious problems or not

Four cases of Method 2

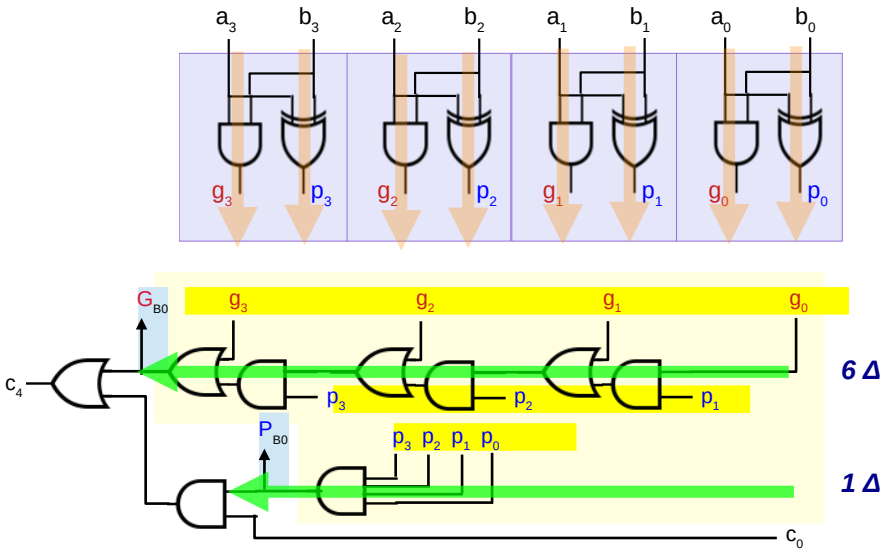
Case A – B	***(2-input, 2-input)***	***6 Δ***	***3 Δ***	***6 Δ***	***+ 1 Δ***	***= 7 Δ***
Case A – B'	***(2-input, m-input)***	***6 Δ***	***1 Δ***	***6 Δ***	***+ 1 Δ***	***= 7 Δ***
Case A' – B	***(m-input, 2-input)***	***2 Δ***	***3 Δ***	***3 Δ***	***+ 1 Δ***	***= 4 Δ***
Case A' – B'	***(m-input, m-input)***	***2 Δ***	***1 Δ***	***2 Δ***	***+ 1 Δ***	***= 3 Δ***

using multiple 2-input gates	***using multi-input gates***
large propagation delay *small power consumption*	*small propagation delay* *large power consumption*
A  ***6 Δ***	***A'***  ***2 Δ***
In the book of S. & D. Harris's book	
B  ***3 Δ***	***B'***  ***1 Δ***
	In the book of S. & D. Harris's book

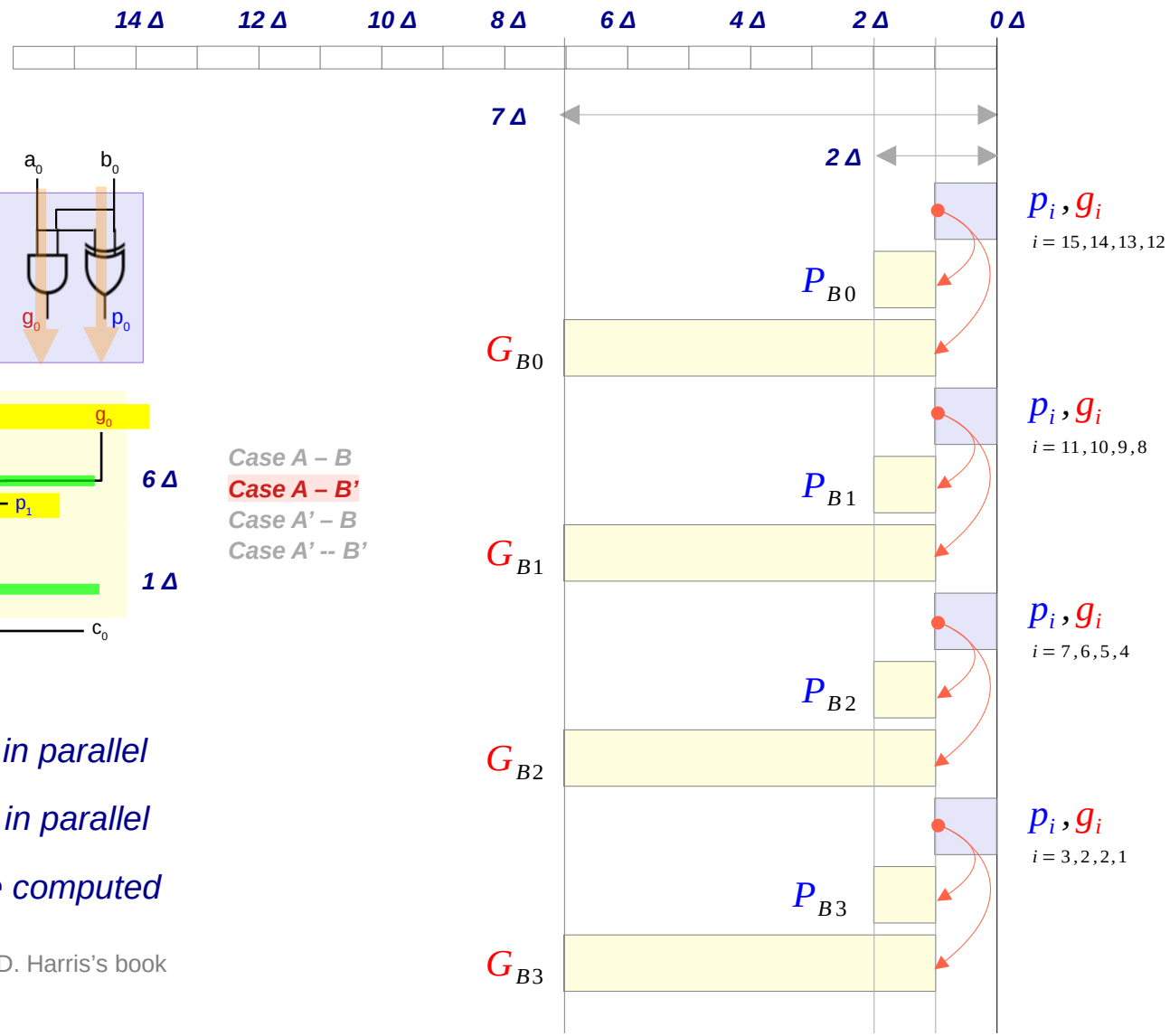
Case 1 (A - B)


14 Δ
12 Δ
10 Δ
8 Δ
6 Δ
4 Δ
2 Δ
0 Δ
7 Δ
4 Δ
a3 b3 a2 b2 a1 b1 a0 b0
g3 p3 g2 p2 g1 p1 g0 p0
GB0
g3 g2 g1 g0
c4
p3 p2 p1
PB0
p3 p2 p1 p0
c0
6 Δ
PBj Need not be computed in one gate delay
PB0
p0 p1 p2 p3
3 Δ
Case A – B
Case A – B'
Case A' – B
Case A' -- B'
pi , gi
i = 15,14,13,12
PB0
GB0
pi , gi
i = 11,10,9,8
PB1
GB1
pi , gi
i = 7,6,5,4
PB2
GB2
pi , gi
i = 3,2,2,1
PB3
GB3

Case 2 (A - B')



Case A – B
Case A – B'
Case A' – B
Case A' -- B'



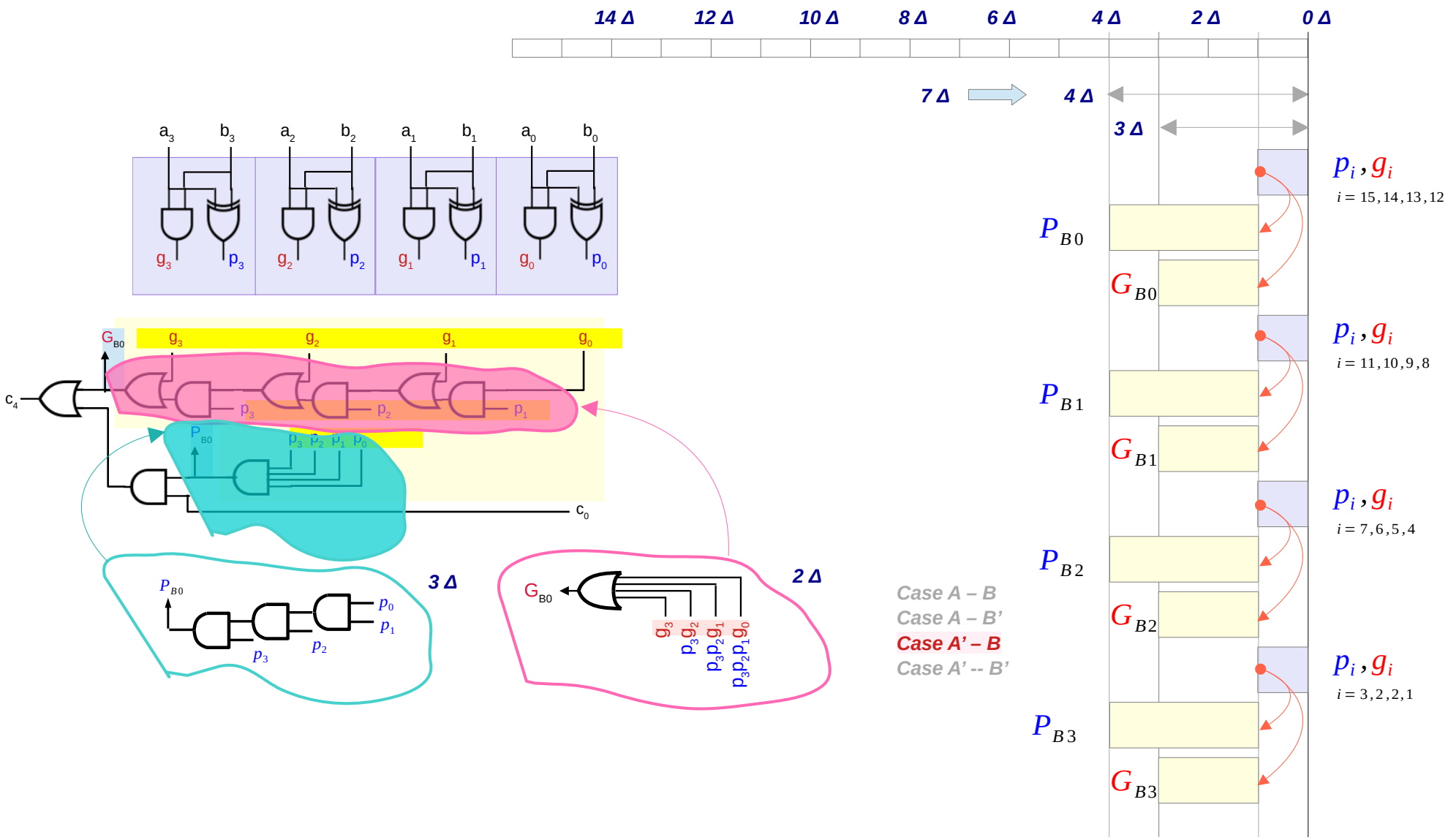
p_i , g_i *can be computed in parallel*

P_{Bj} , G_{Bj} *can be computed in parallel*

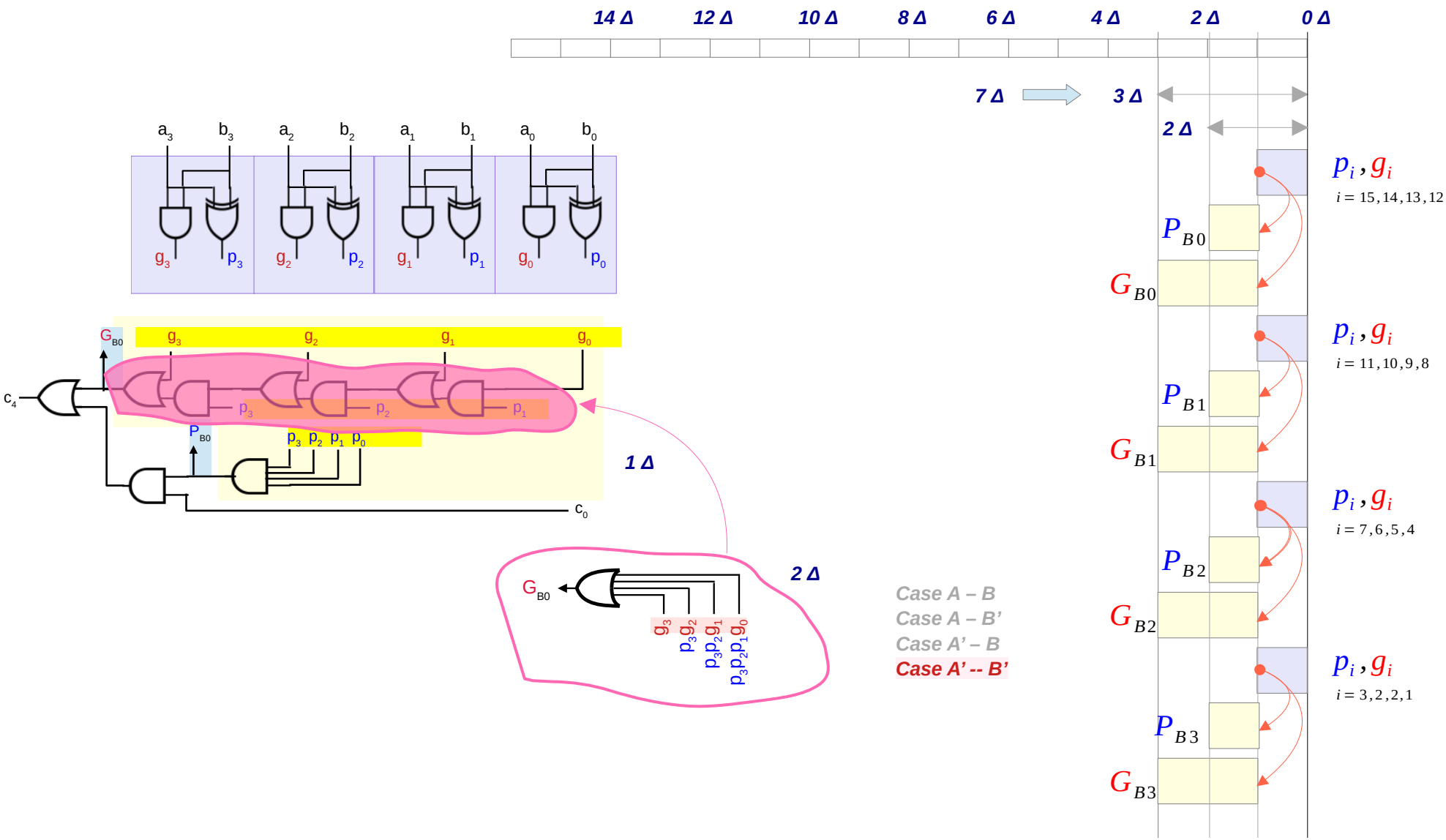
after p_i , g_i *are computed*

In the book of S. & D. Harris's book

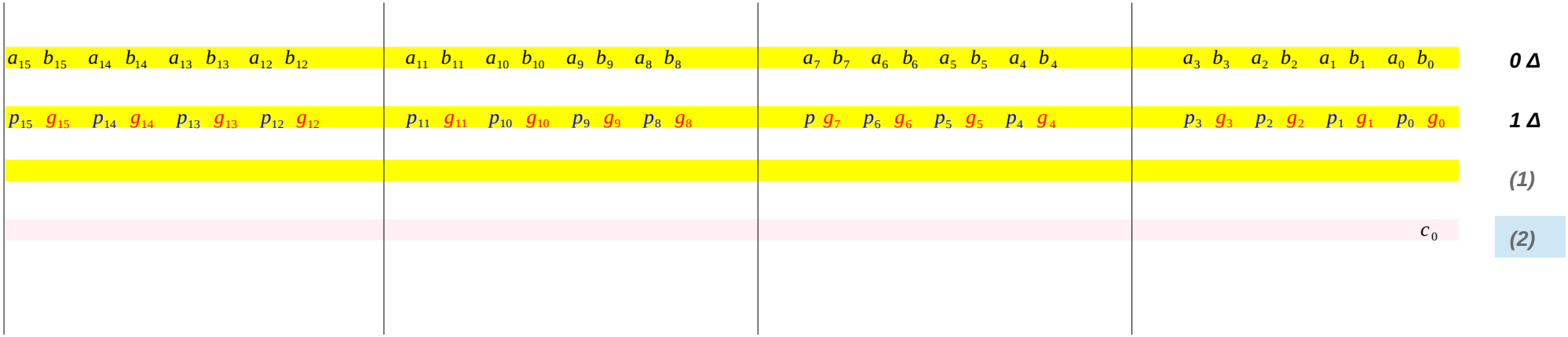
Case 3 (A' - B)


14 Δ
12 Δ
10 Δ
8 Δ
6 Δ
4 Δ
2 Δ
0 Δ
7 Δ
4 Δ
3 Δ
p_i, g_i
$i = 15,14,13,12$
P_{B0}
G_{B0}
p_i, g_i
$i = 11,10,9,8$
P_{B1}
G_{B1}
p_i, g_i
$i = 7,6,5,4$
P_{B2}
G_{B2}
p_i, g_i
$i = 3,2,2,1$
P_{B3}
G_{B3}
a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0
g_3 p_3 g_2 p_2 g_1 p_1 g_0 p_0
G_{B0}
c_4
c_0
3 Δ
2 Δ
Case A – B
Case A – B'
Case A' – B
Case A' -- B'

Case 4 (A' - B')

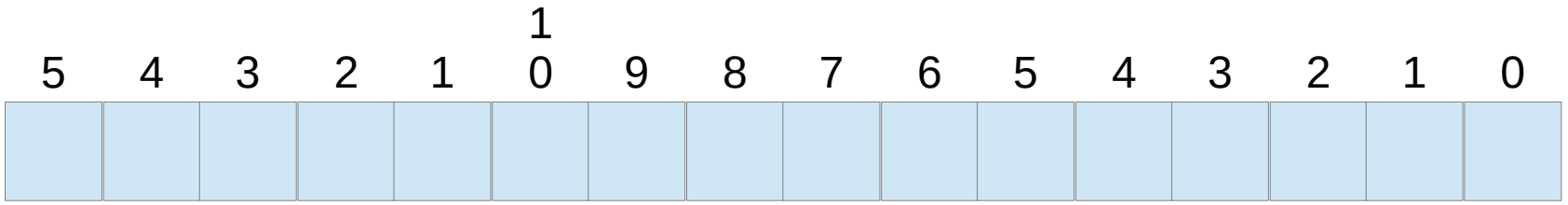

14 Δ
12 Δ
10 Δ
8 Δ
6 Δ
4 Δ
2 Δ
0 Δ
7 Δ
3 Δ
2 Δ
a3 b3 a2 b2 a1 b1 a0 b0
g3 p3 g2 p2 g1 p1 g0 p0
GB0 g3 g2 g1 g0
c4
p3 p2 p1
PB0
p3 p2 p1 p0
c0
1 Δ
GB0
g3
p3g2
p3p2g1
p3p2p1g0
2 Δ
Case A – B
Case A – B'
Case A' – B
Case A' -- B'
pi , gi
i = 15,14,13,12
PB0
GB0
pi , gi
i = 11,10,9,8
PB1
GB1
pi , gi
i = 7,6,5,4
PB2
GB2
pi , gi
i = 3,2,2,1
PB3
GB3

Delays to generate all block carries c_4, c_8, c_{12}, c_{16}



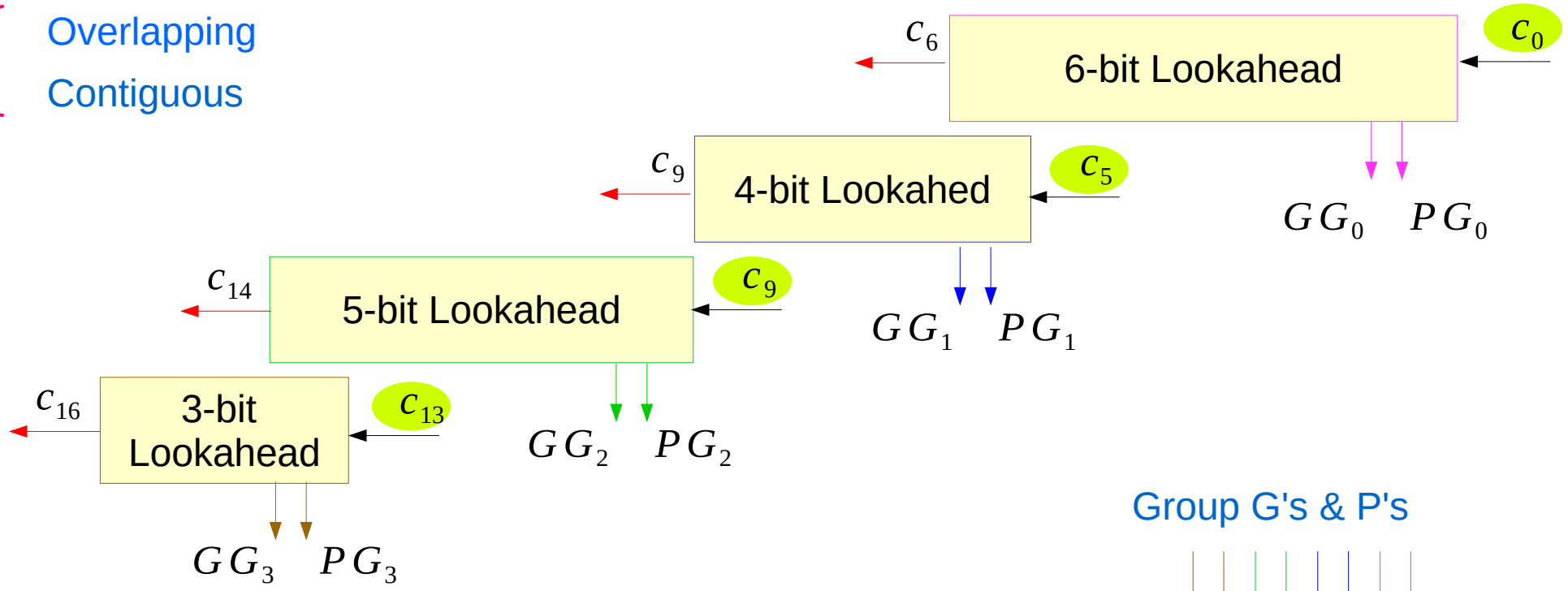
	G		P		
Case A – B	7 Δ	(2)	4 Δ	(1)	7 Δ
Case A – B'	7 Δ	(2)	2 Δ	(1)	7 Δ
Case A' – B	3 Δ	(1)	4 Δ	(2)	4 Δ
Case A' – B'	3 Δ	(2)	2 Δ	(1)	3 Δ

Multi-level Carry Lookahead



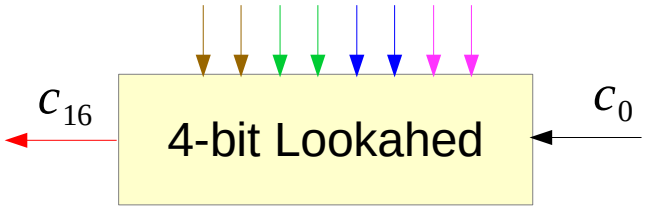
Overlapping

Contiguous

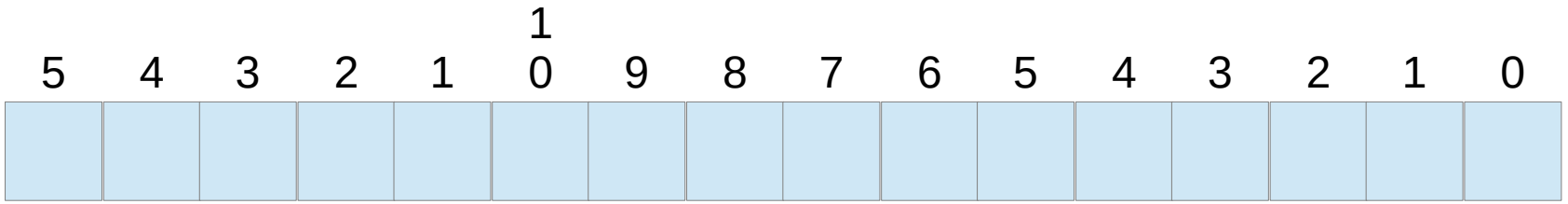


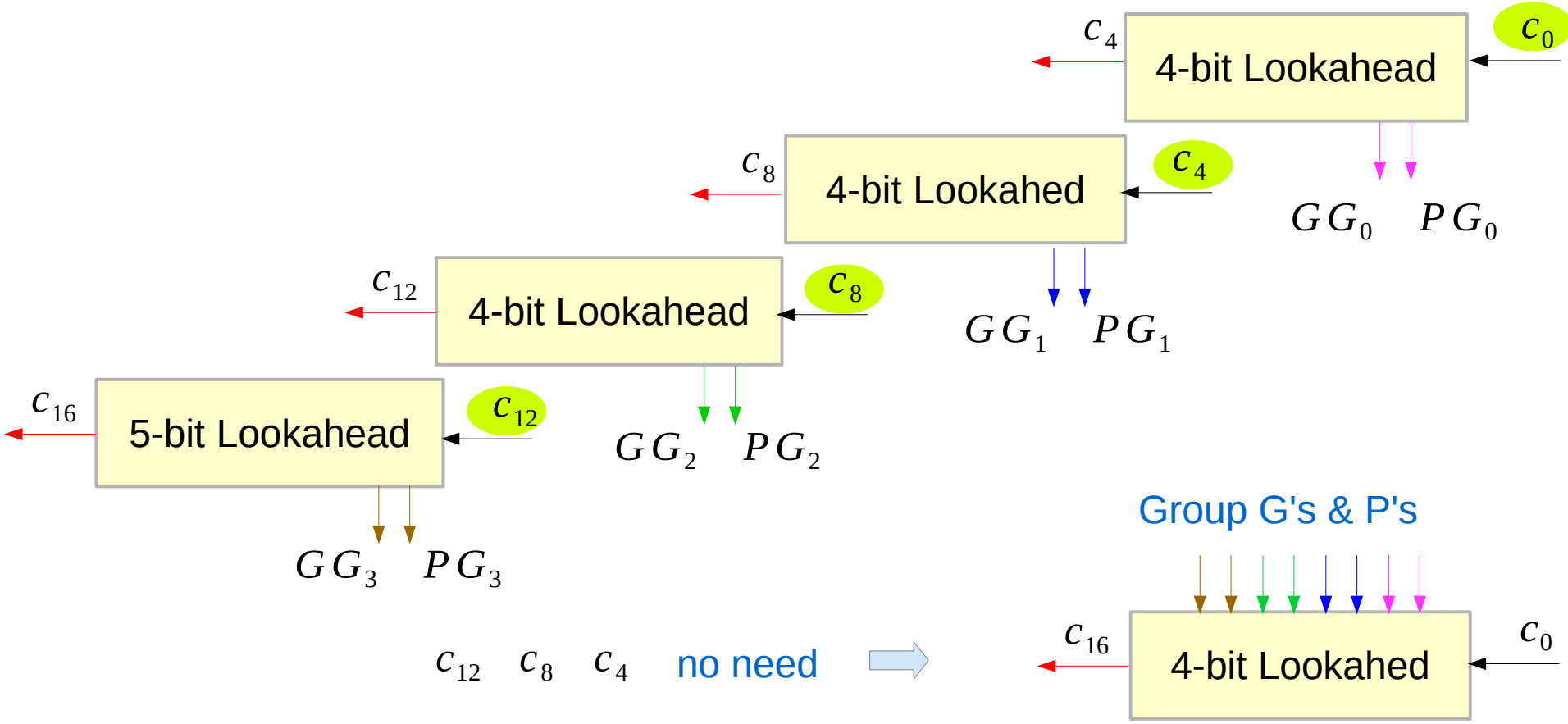
Group G's & P's

c_{14} c_9 c_6 no need

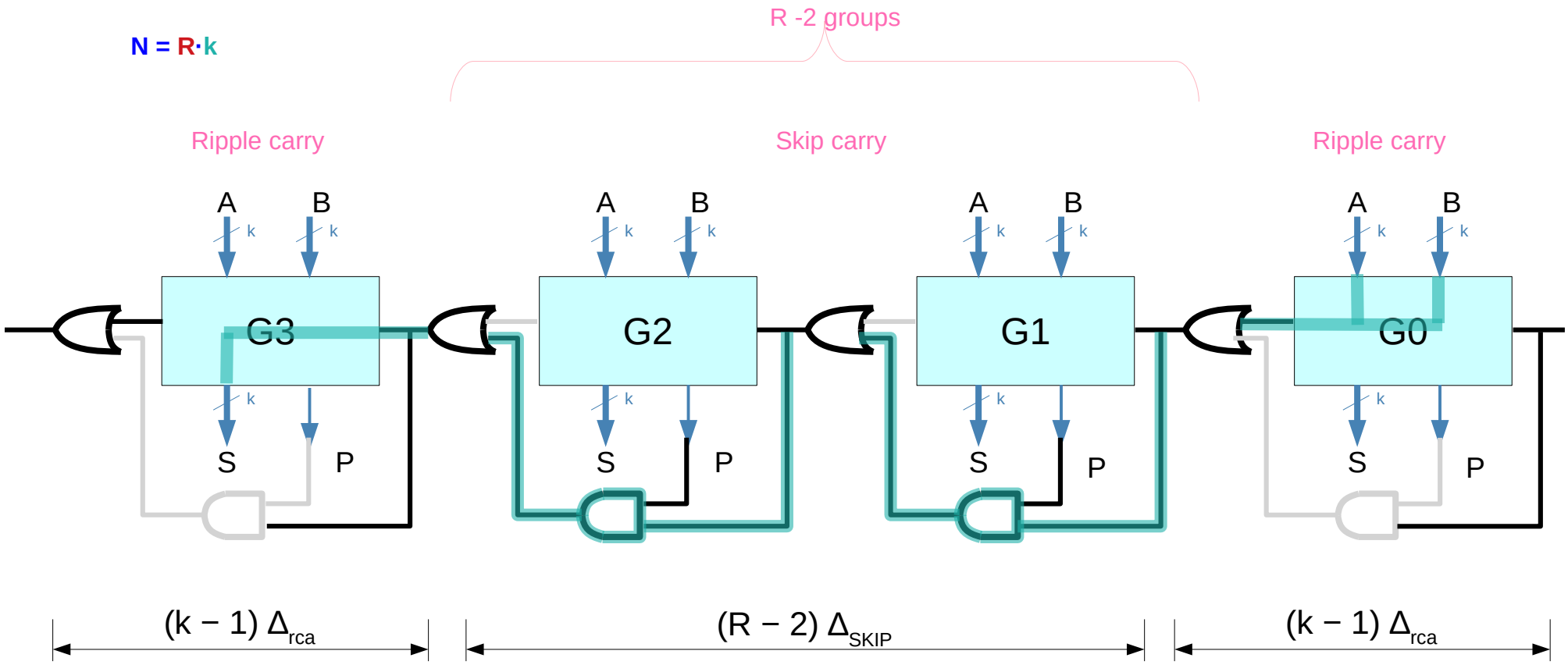


Contiguous Multi-level Carry Lookahead





Carry Skip Adder



Any kill or generate condition results in divided (broken) critical paths

All FA's in R-2 groups must have the propagate condition