

# Carry Skip Adder (5A)

---

- 
-

Copyright (c) 2024 – 2013 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

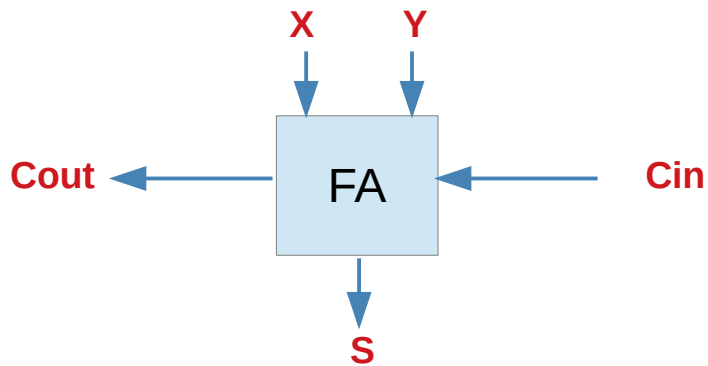
[https://en.wikipedia.org/wiki/AND\\_gate](https://en.wikipedia.org/wiki/AND_gate)  
[https://en.wikipedia.org/wiki/OR\\_gate](https://en.wikipedia.org/wiki/OR_gate)  
[https://en.wikipedia.org/wiki/XOR\\_gate](https://en.wikipedia.org/wiki/XOR_gate)  
[https://en.wikipedia.org/wiki/NAND\\_gate](https://en.wikipedia.org/wiki/NAND_gate)

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice and Octave.

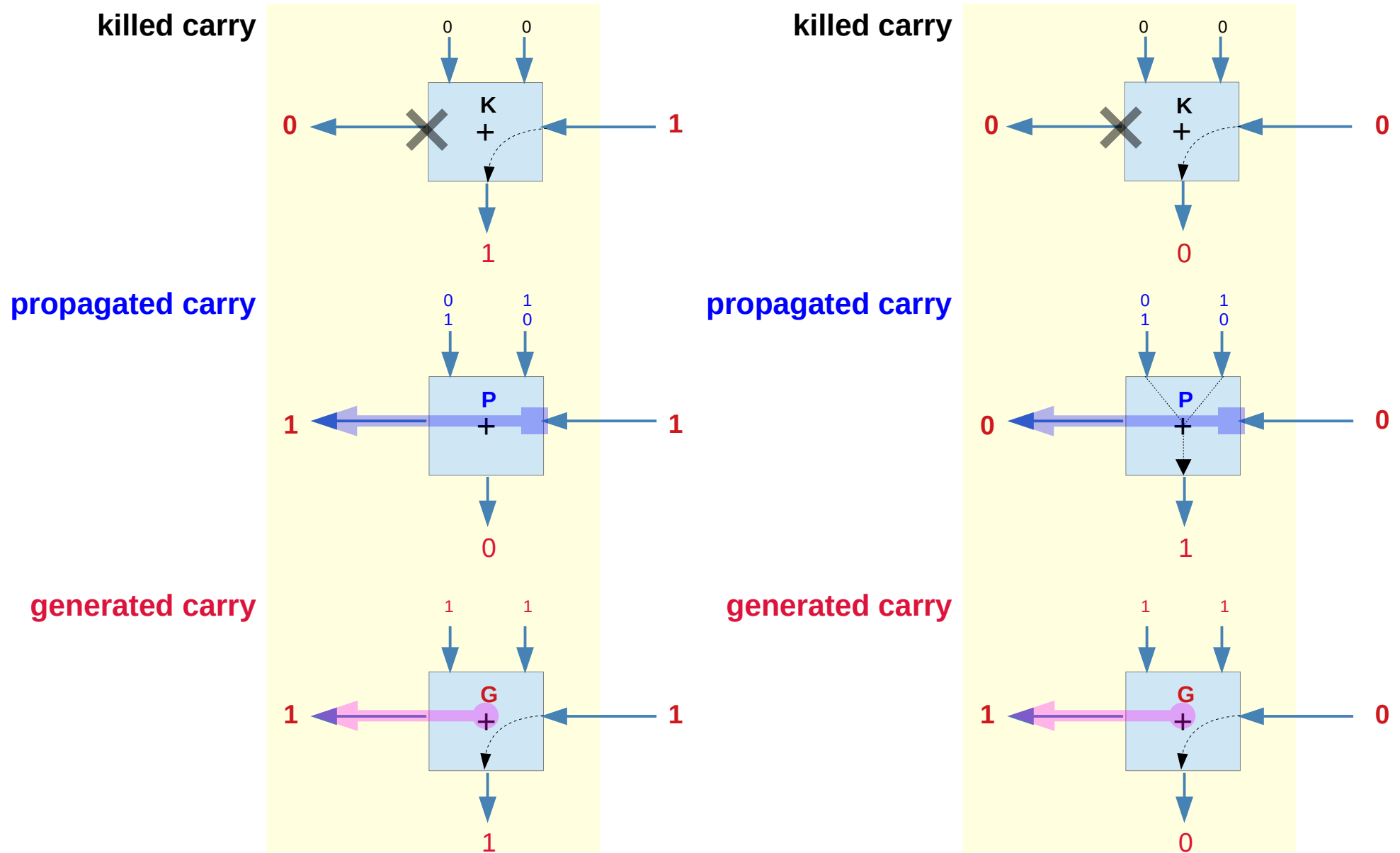
# Carry Kill, Propagate, Generate conditions (1)

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



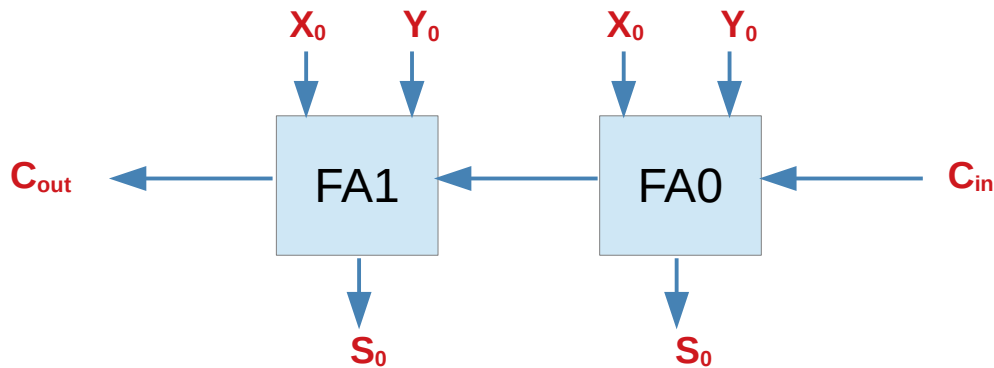
<https://electronics.stackexchange.com/questions/21251/critical-path-for-carry-skip-adder>

# Carry Kill, Propagate, Generate conditions (2)



# K, P, and G conditions in a 2-bit adder (1)

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



Unless the two FA's are in **propagate** mode, the transition of  $C_{in}$  does not affect the transition of  $C_{out}$

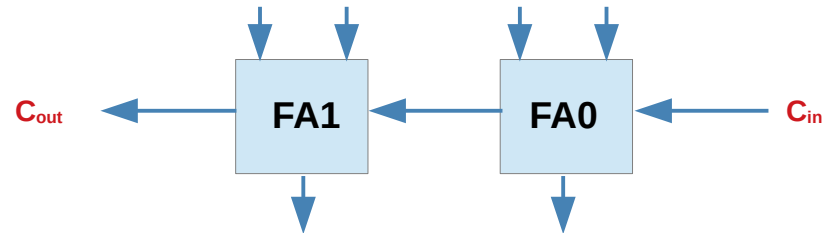
Critical path – all FA's in **propagate** mode

Broken paths for any FA in other mode  
- kill mode, **generate** mode

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

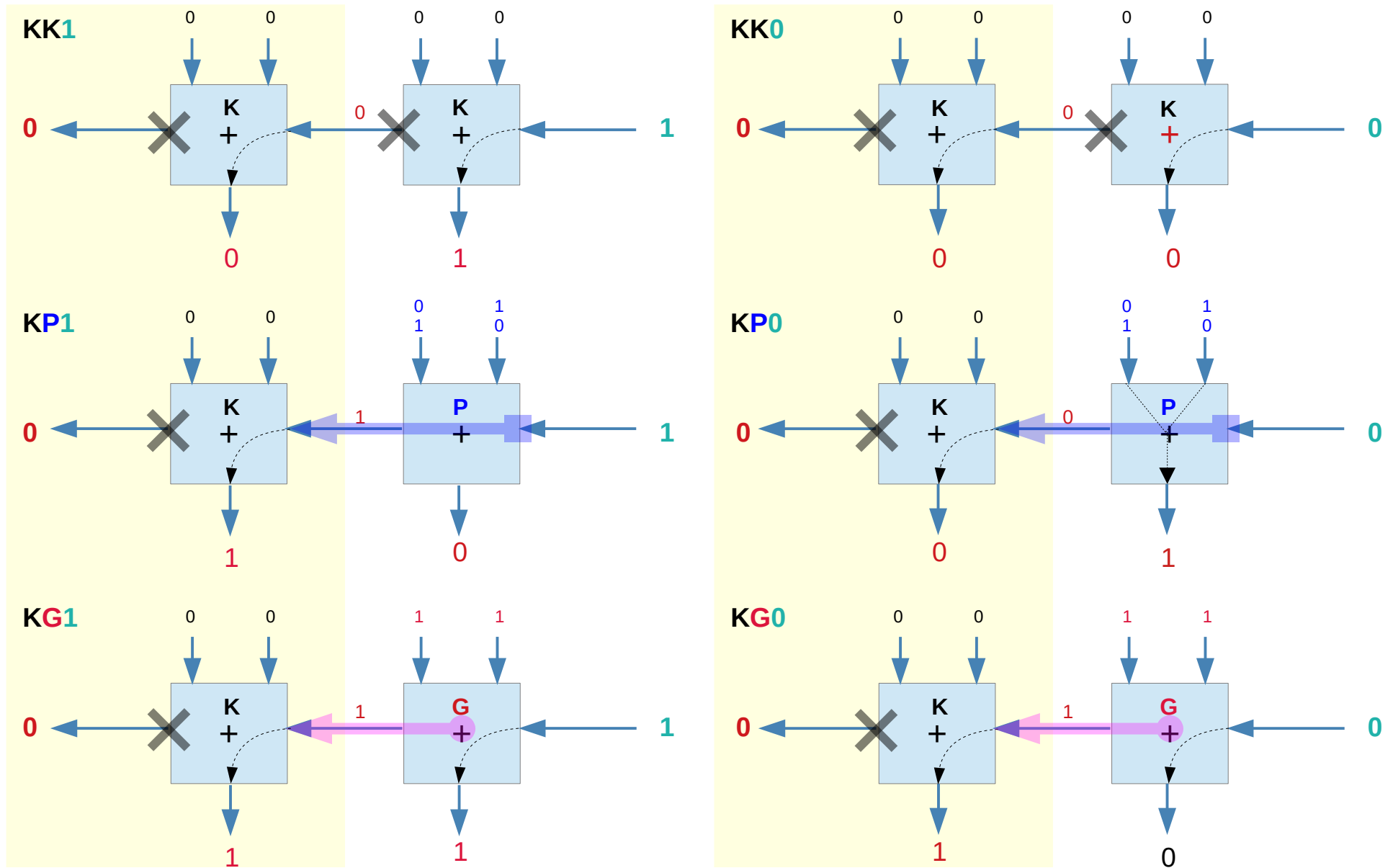
# K, P, and G conditions in a 2-bit adder (2)

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

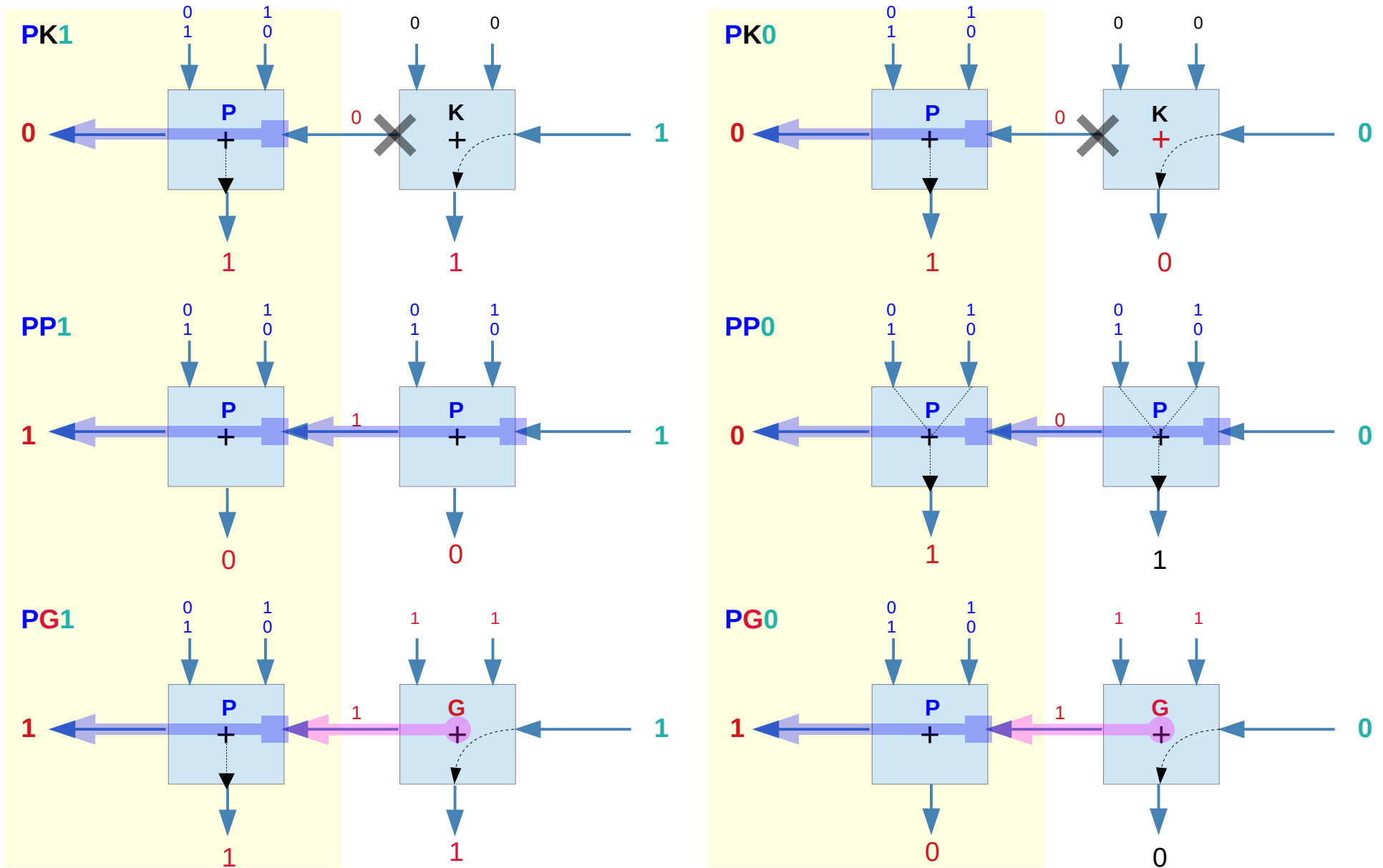


K	K	0
K	K	1
K	P	0
K	P	1
K	G	0
K	G	1
P	K	0
P	K	1
P	P	0
P	P	1
P	G	0
P	G	1
G	K	0
G	K	1
G	P	0
G	P	1
G	G	0
G	G	1

# 1. Cases when **FA1** is in the **K** mode

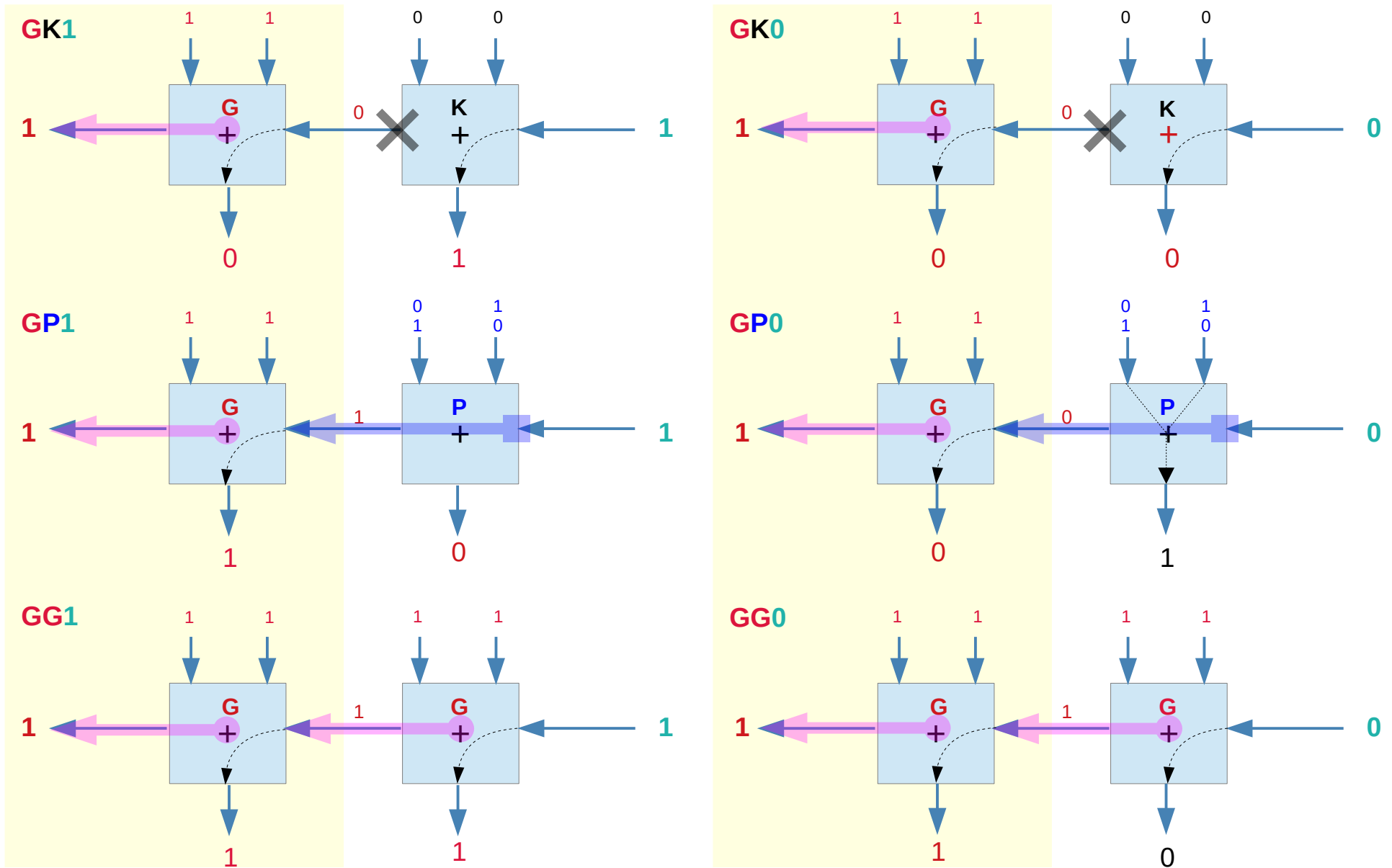


## 2. Cases when **FA1** is in the **P** mode

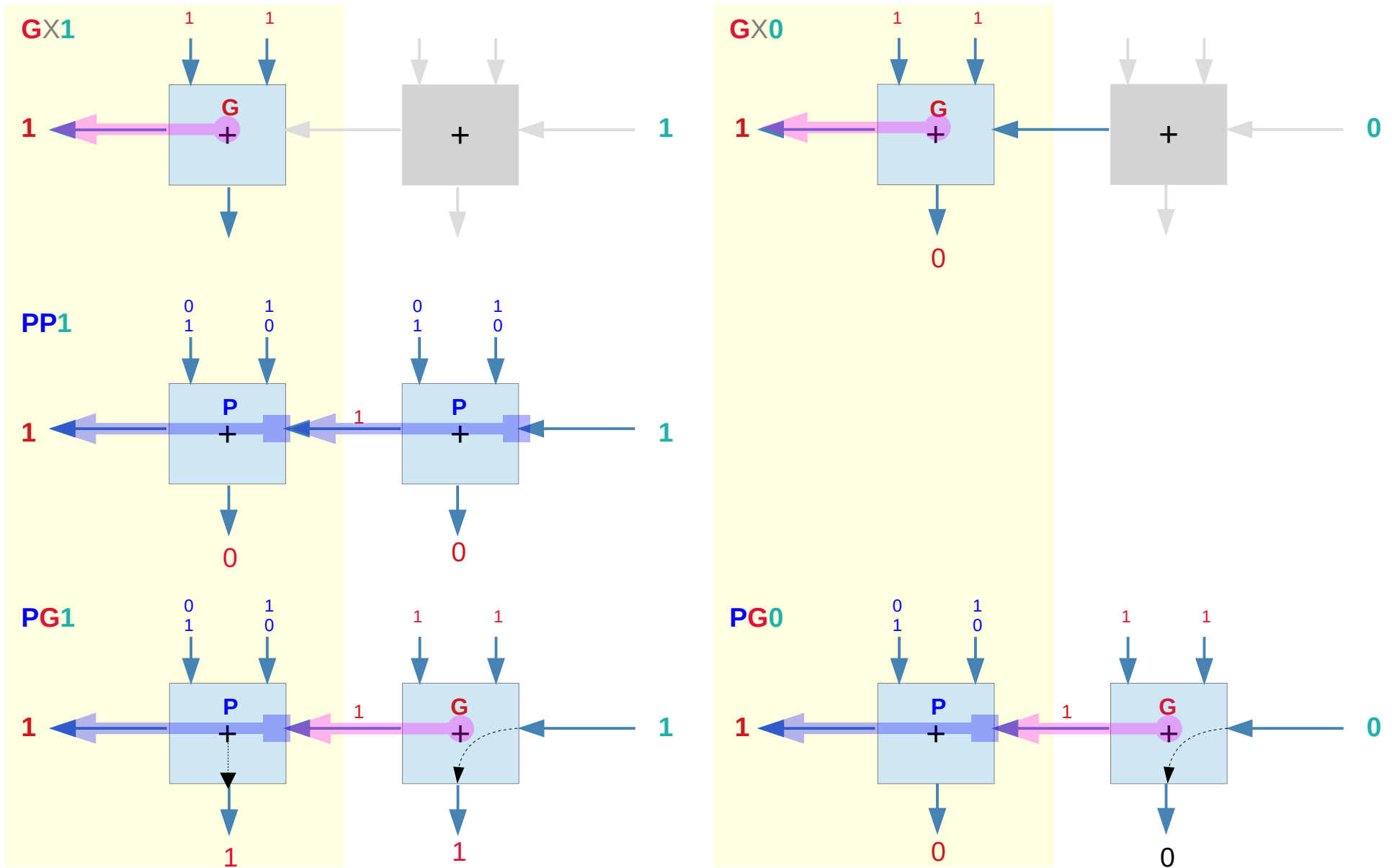




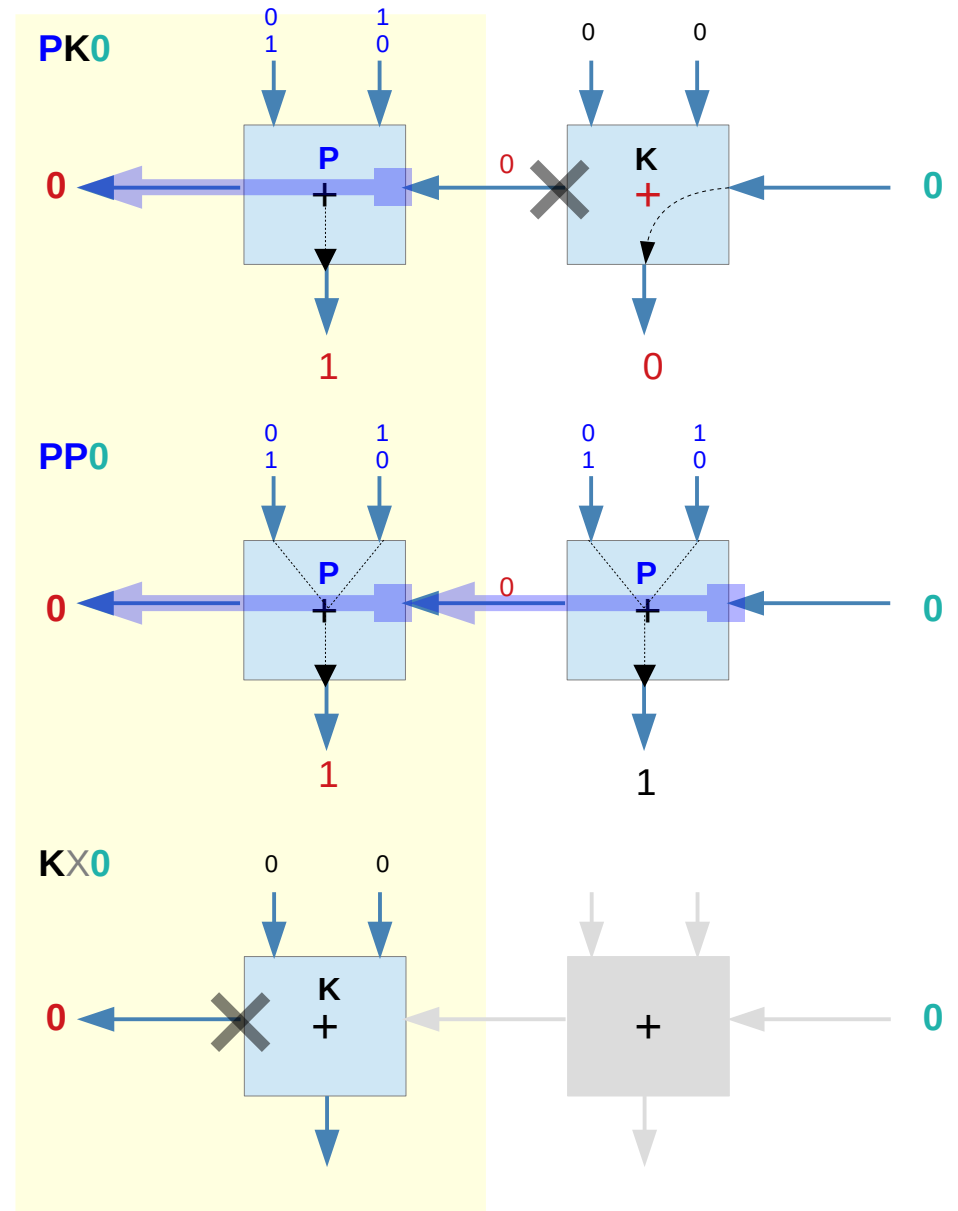
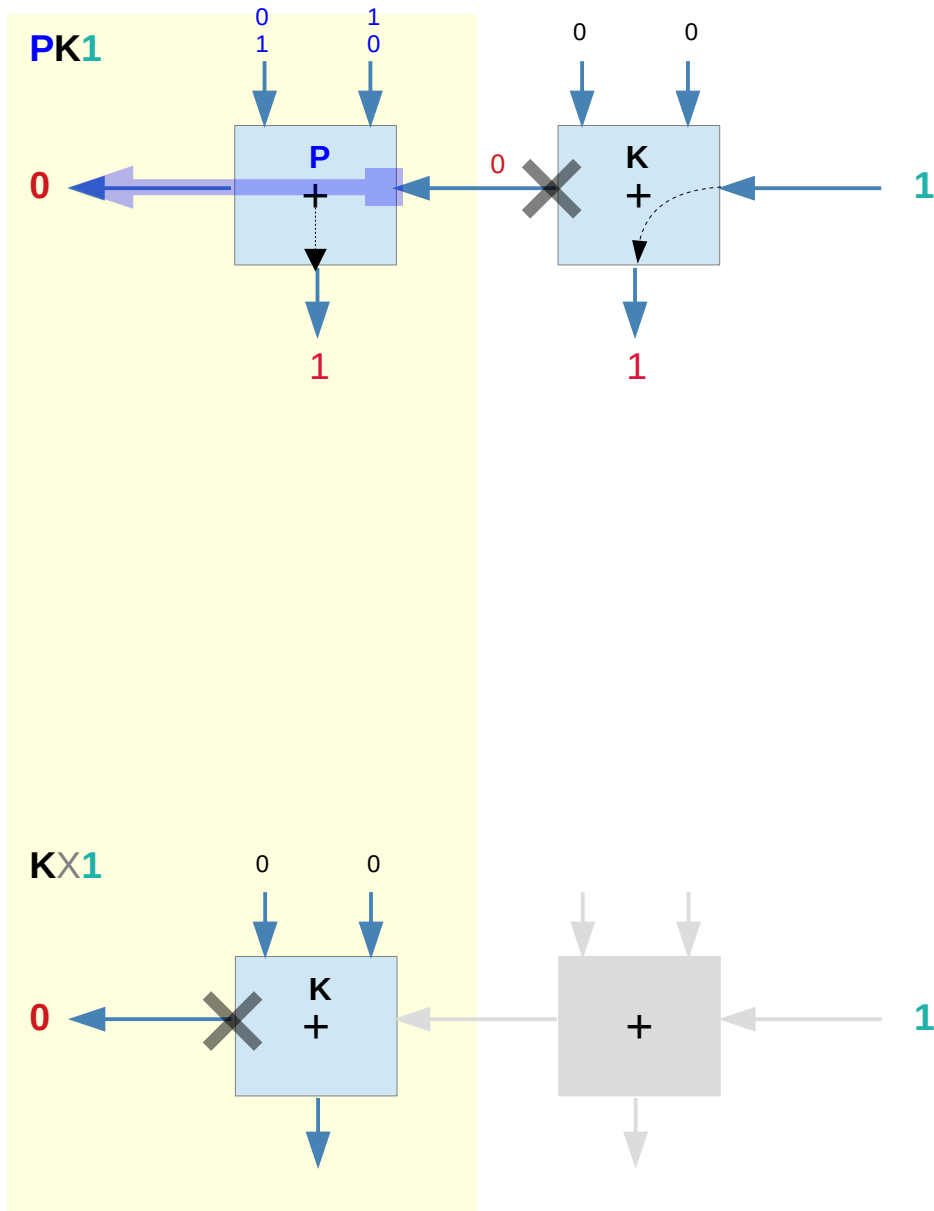
### 3. Cases when **FA1** is in the **G** mode



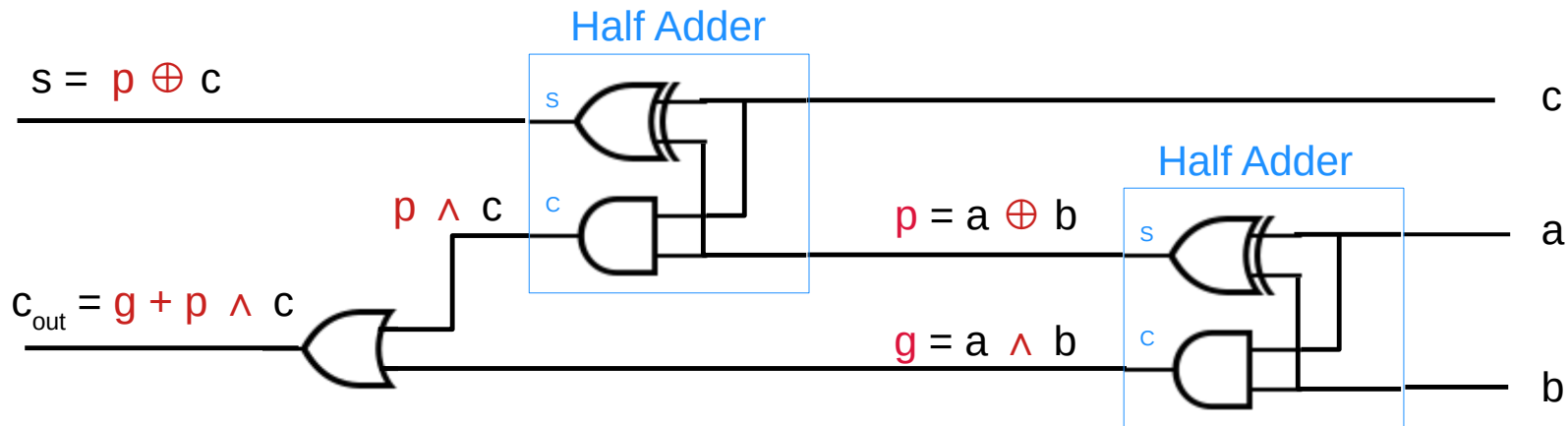
# Cases for $C_{out} = 1$



# Cases for $C_{out} = 0$

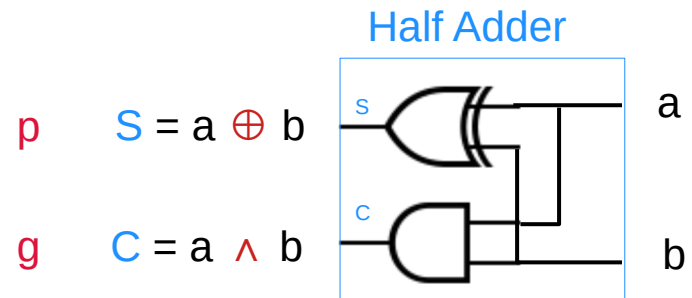


# FA with P & G



Half Adder  
 $S = a \oplus b$   
 $C = a \wedge b$

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



Full adder with additional generate and propagate signals.

[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Ripple Carry Adder

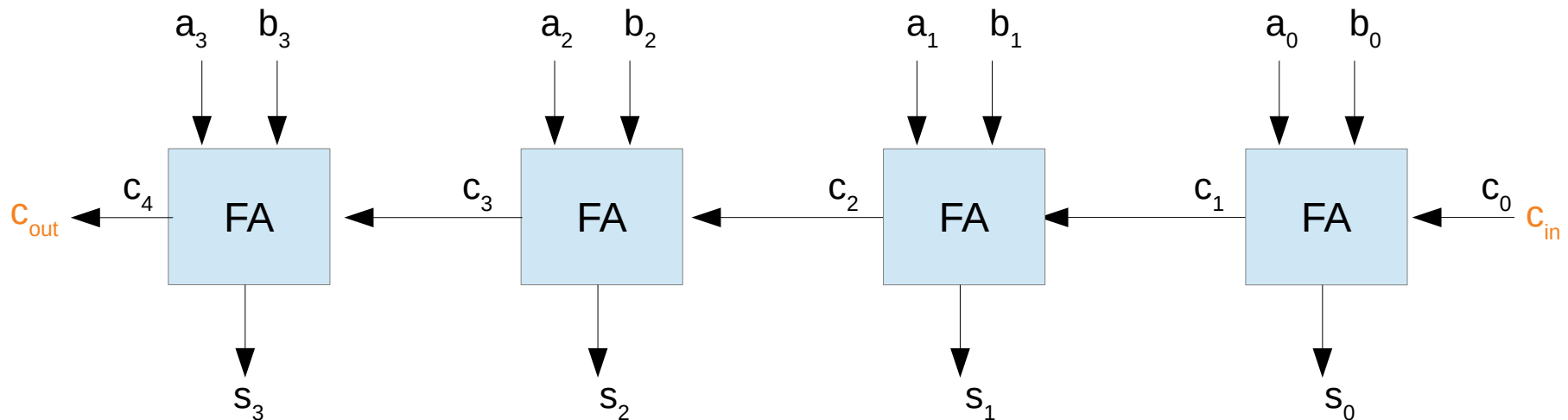
$$p_i = a_i \oplus b_i$$

$$g_i = a_i \wedge b_i$$

$$\begin{aligned} c_1 &= g_0 + p_0 \wedge c_0 \\ c_2 &= g_1 + p_1 \wedge c_1 \\ c_3 &= g_2 + p_2 \wedge c_2 \\ c_4 &= g_3 + p_3 \wedge c_3 \end{aligned}$$

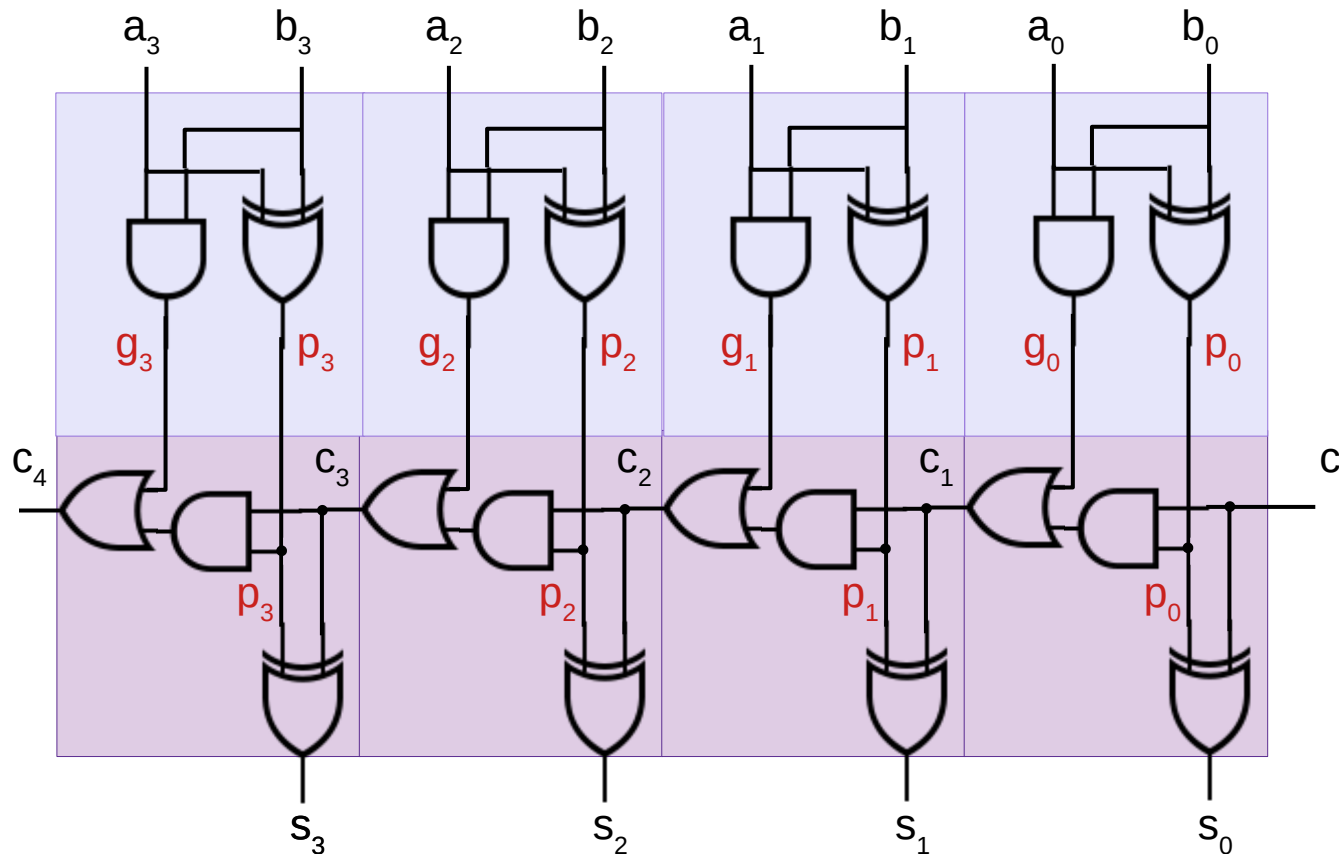
generated carry

propagated carry



[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# 4-bit Full Adder with P and G



**Half Adder**

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \wedge b_i$$

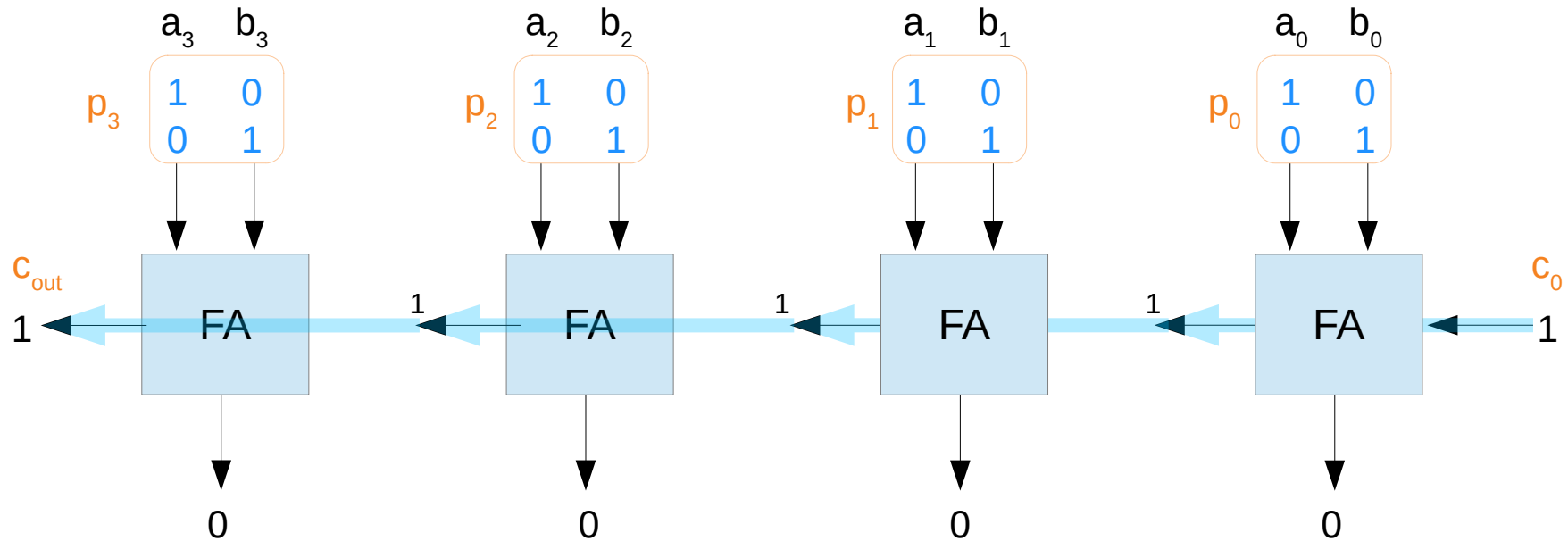
**Half Adder**

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

$$s_i = p_i \oplus c_i$$

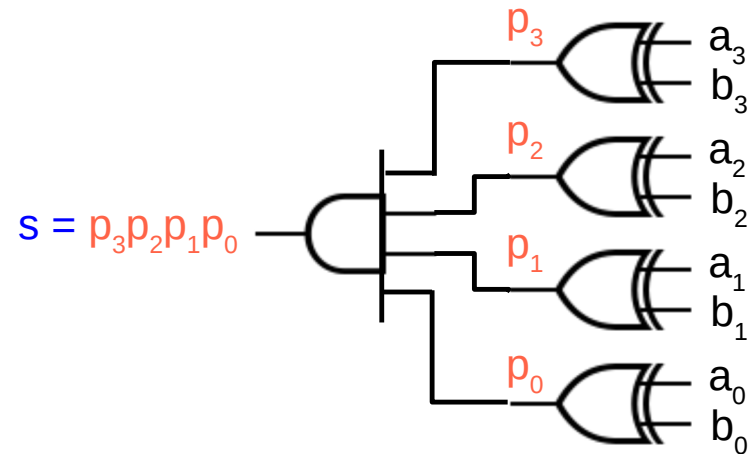
<https://upload.wikimedia.org/wikiversity/en/1/18/RCA.Note.H.1.20151215.pdf>

# $C_0$ propagation condition



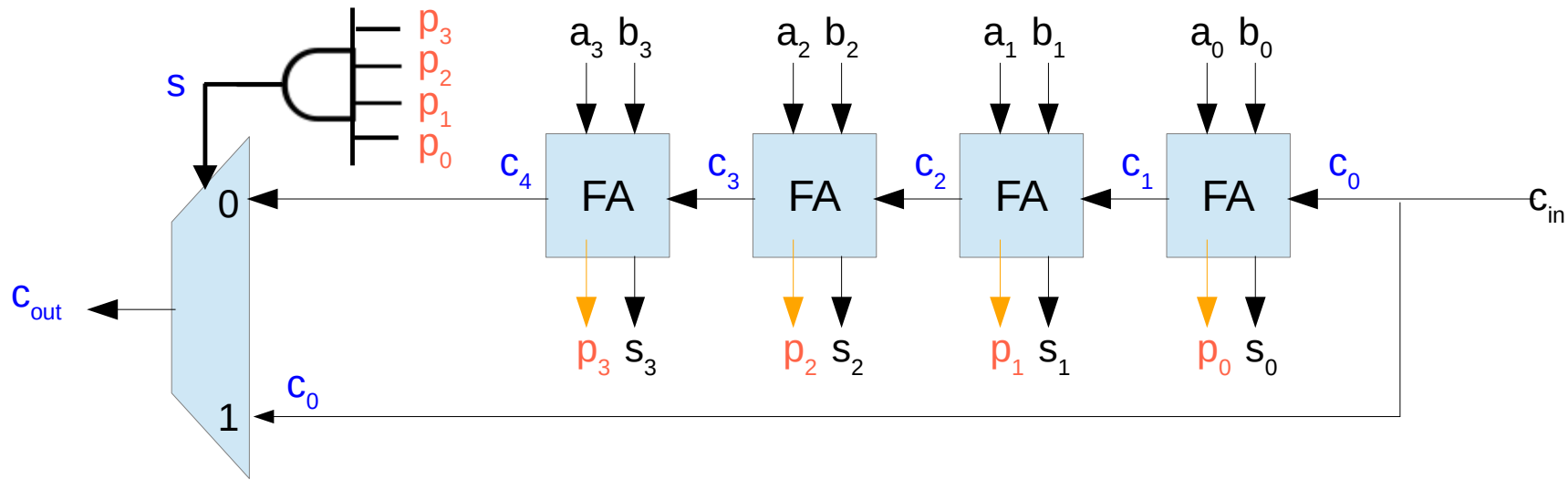
$c_0$  can be propagated to  $c_{out}$  only when  $s = 1$

$$\begin{aligned}
 s &= p_3 \wedge p_2 \wedge p_1 \wedge p_0 = p_{[3:0]} \\
 &= (a_3 \oplus b_3) \\
 &\quad \wedge (a_2 \oplus b_2) \\
 &\quad \wedge (a_1 \oplus b_1) \\
 &\quad \wedge (a_0 \oplus b_0)
 \end{aligned}$$



[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Carry Skip Adder



The n-bit Carry Skip Adder consists of

a n-bit **carry-ripple-chain**,  
a n-input **AND-gate** and  
one **multiplexer**.

a multiplexer switches  
either the last carry-bit  $c_n$  or the carry-in  $c_0$   
to the carry-out signal  $c_{out}$

$$s = p_3 \wedge p_2 \wedge p_1 \wedge p_0 = p_{[3:0]}$$

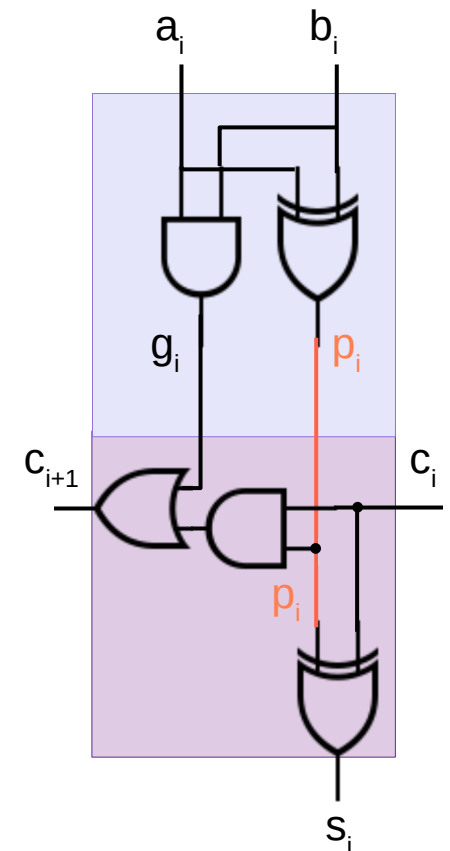
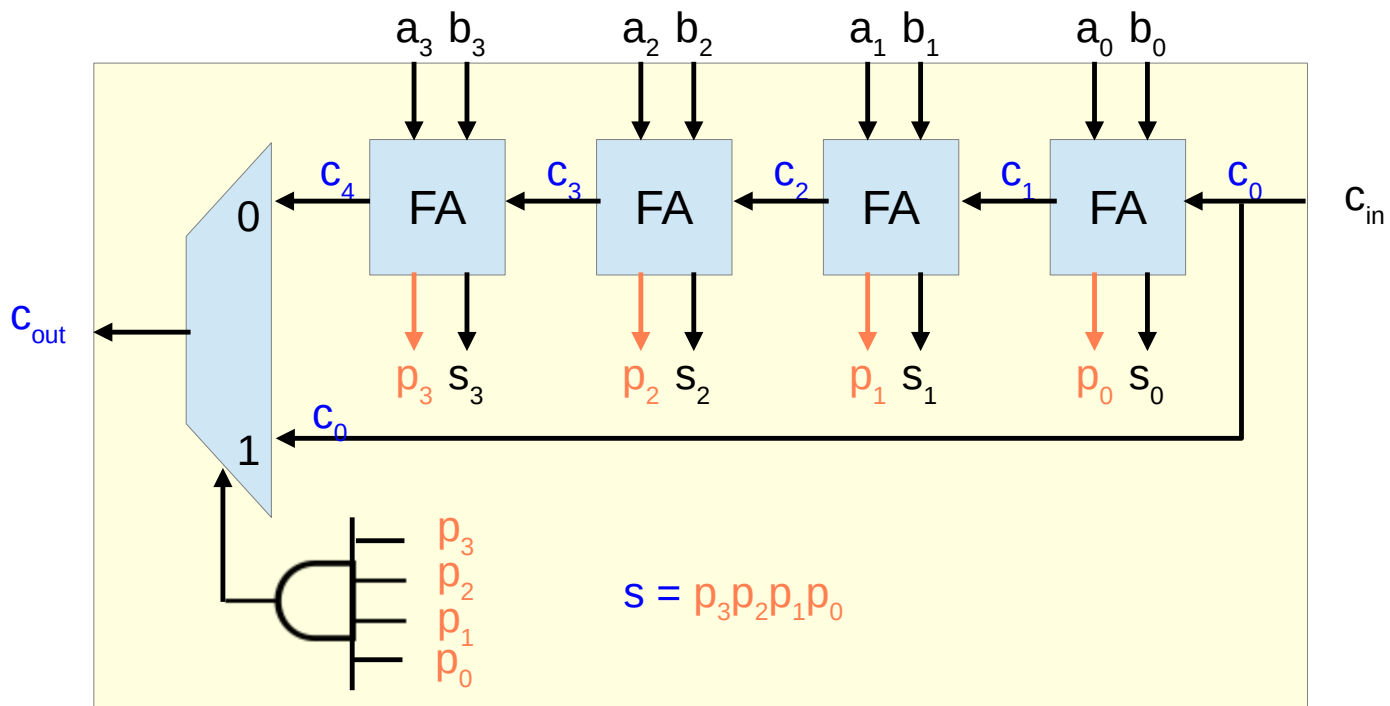
when  $s = 1$ ,  $c_{out} \leftarrow c_0$

otherwise, internally generated carries  
can be propagated to  $c_{out} \leftarrow c_4$

[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)



# Carry Skip Adder



[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Carry Skip Adder

The **critical path** of a Carry Skip Adder begins at the first full adder, passes through all adders and ends at the sum bit  $s_{n-1}$

Since a single *n-bit* Carry Skip Adder has no real speed benefit compared to a *n-bit* Ripple Carry Adder

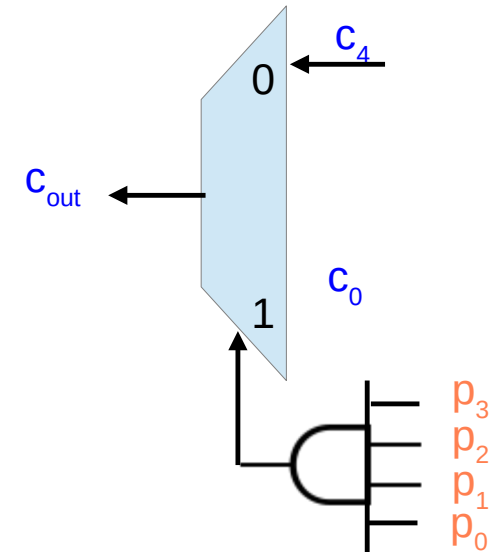
$$T_{CSA}(n) = T_{RCA}(n)$$

Carry Skip Adders are chained to reduce the overall **critical path**, (Block Carry Skip Adders)

The skip logic consists of a **m-input** AND gate and one MUX

$$T_{SK} = T_{AND}(m) + T_{MUX}$$

[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)



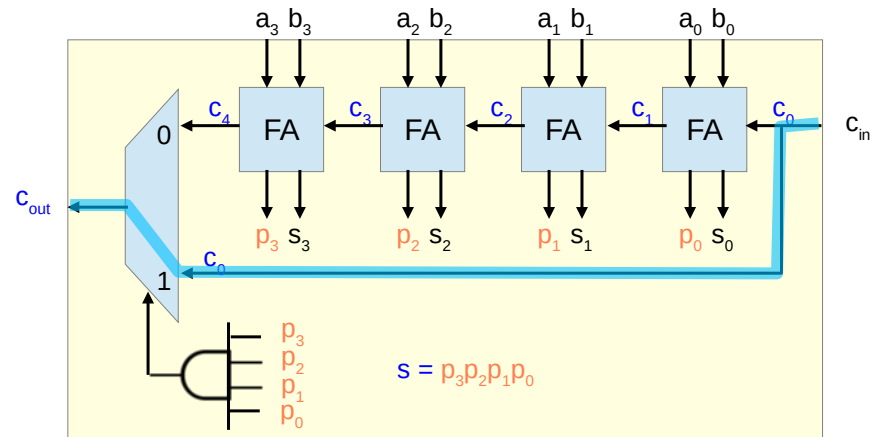
# Carry Skip Adder

As the **propagate** signals are computed in parallel and are early available,

$$p_i = a_i \oplus b_i$$

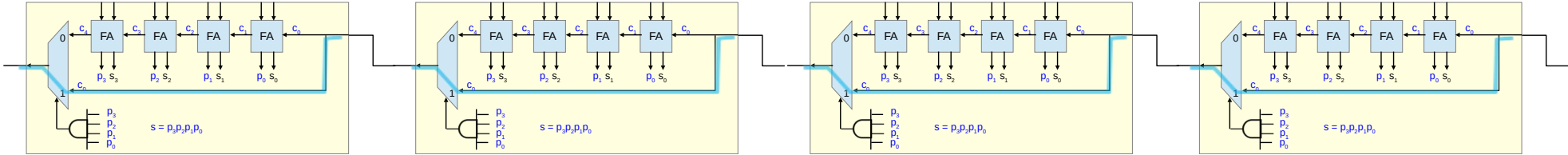
the **critical path** for the skip logic in a Carry Skip Adder consists of the delay imposed by the multiplexer (conditional skip)

$$T_{CSK} = T_{MUX} = 2D$$



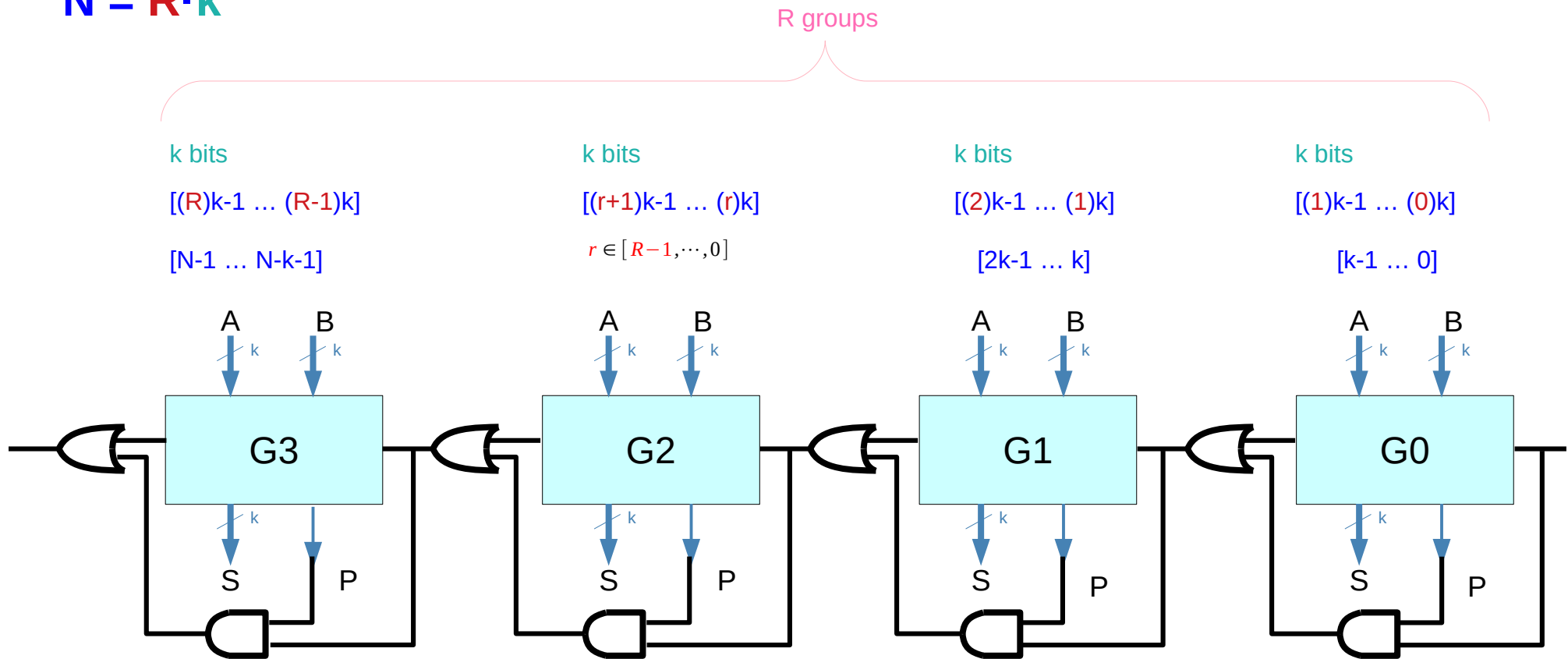
[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Carry Skip Adder

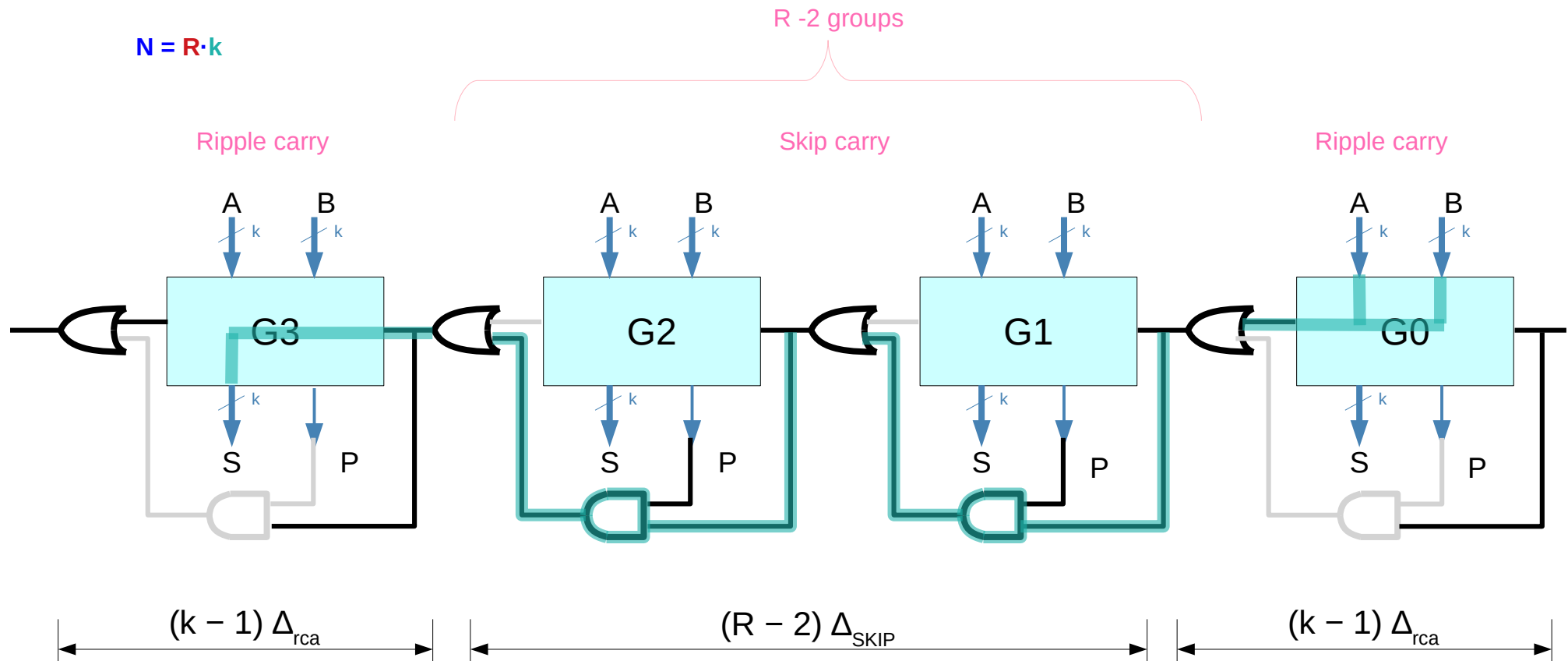


# Carry Skip Adder

$$N = R \cdot k$$



# 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

# Carry Skip Adder

Fixed size block carry skip adders split the  $n$  bits of the input bits into blocks of  $k$  bits each, resulting in  $R = n / k$  blocks.

The critical path consists of

the ripple path and the skip element of the first block,  $T_{\text{CRA}[0:\text{cout}]}(k) + T_{\text{CSK}}$

The skip paths that are enclosed between the first and the last block,  $(R-2)T_{\text{CSK}}$

And finally the ripple path of the last block  $T_{\text{CRA}}(k)$

$$\begin{aligned} T_{\text{FCSA}}(n) &= T_{\text{CRA}[0:\text{cout}]}(k) + T_{\text{CSK}} + (R-2)T_{\text{CSK}} + T_{\text{CRA}}(k) \\ &= 3D + k \cdot 2D + (R-1)2D + (k+2)2D \\ &= (2R+k)2D + 5d \end{aligned}$$

R groups

k bits

[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Carry Skip Adder

$$\begin{aligned}T_{\text{FCSA}}(n) &= T_{\text{CRA}[0:\text{cout}]}(k) + T_{\text{CSK}} + (R-2)T_{\text{CSK}} + T_{\text{CRA}}(k) \\ &= 3D + k \cdot 2D + (R-1)2D + (k+2)2D \\ &= (2R+k)2D + 5d\end{aligned}$$

The optimal block size for a given adder width  $n$  is derived by equating to 0

$$dT_{\text{FCSA}}(n) / dk = 0$$

$$2D(2-n(1/k^2)) = 0$$

$$k_{1,2} = \pm\sqrt{n/2} \quad k = \sqrt{k/2}$$

[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)



# Carry Skip Adder

Block carry skip adders are composed of a number of carry skip adders

There are two types of block carry skip adders

The two operands  $A = (a_{n-1}, a_{n-2}, \dots, a_1, a_0)$  and  $B = (b_{n-1}, b_{n-2}, \dots, b_1, b_0)$

Are split in  $k$  blocks of  $(m_k, m_{k-1}, \dots, m_2, m_1)$  bits

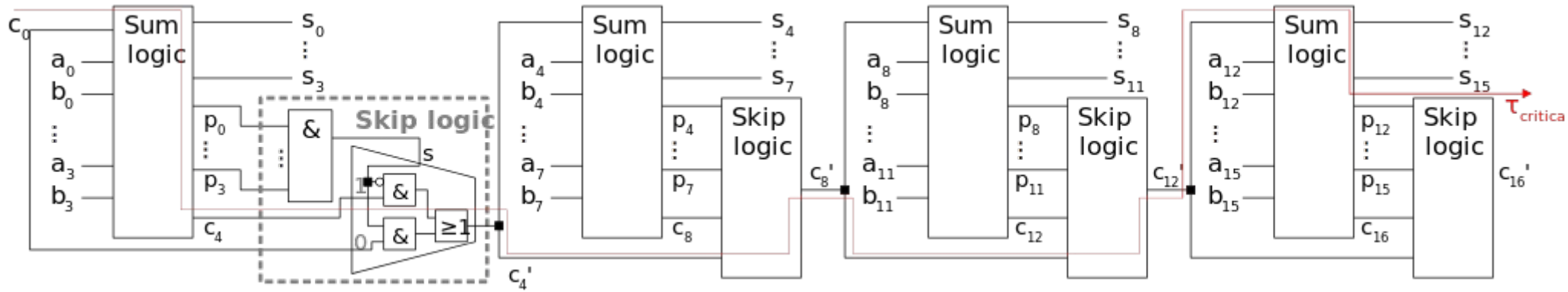
Why are block carry skip adders used

Should the block size be constant or variable?

Fixed block width vs. variable block width

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

# Block Carry Skip Adder



[https://en.wikipedia.org/wiki/Carry-skip\\_adder](https://en.wikipedia.org/wiki/Carry-skip_adder)

# Carry Skip Adder

Since the **Cin-to-Cout** represents the longest path in the ripple-carry-adder, an obvious attempt is to accelerate carry propagation through the adder.

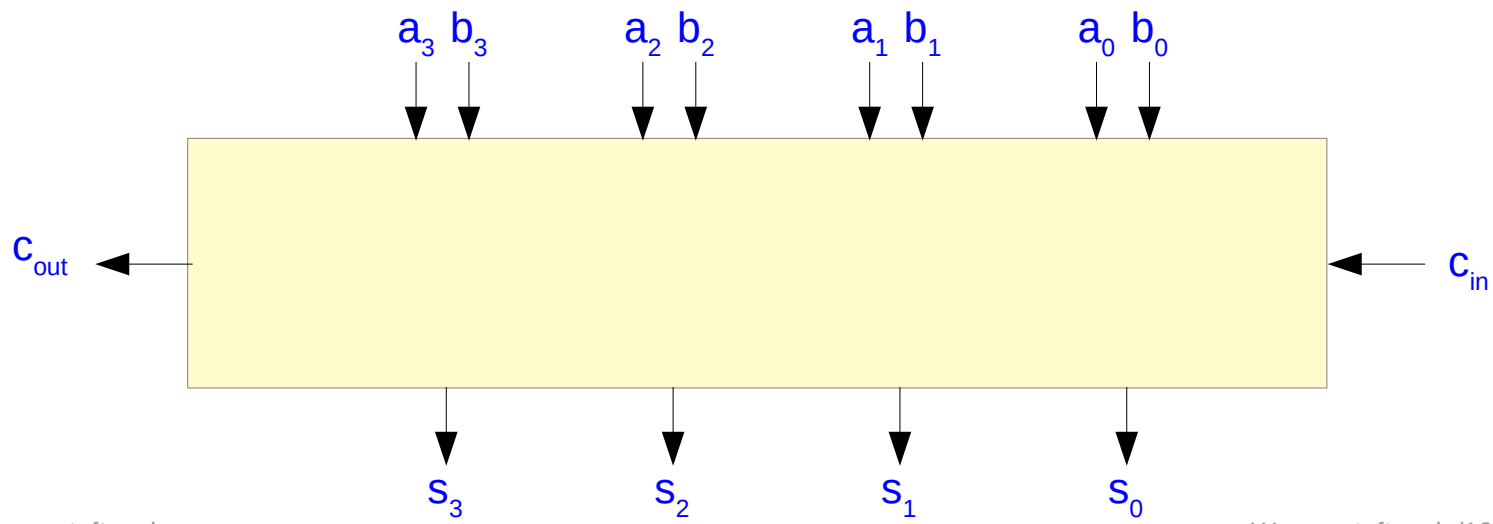
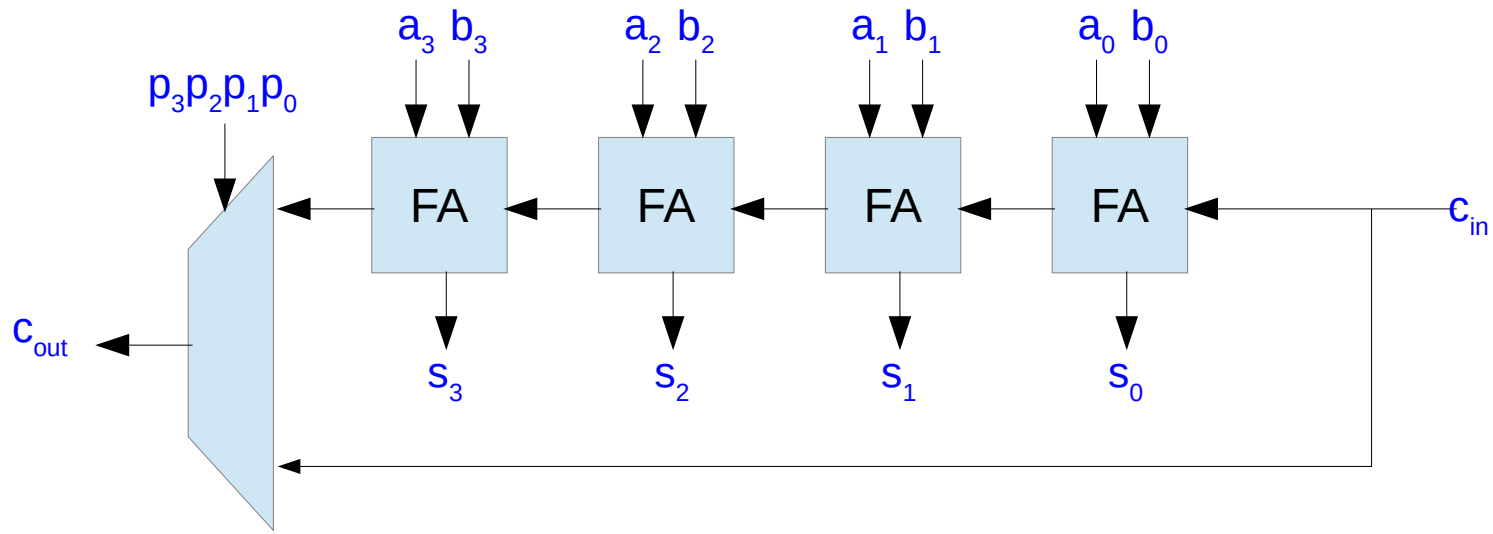
This is accomplished by using **Carry-Propagate**  $p_i$  signals within a group of bits.

If all the  $p_i$  signals within the group are  $p_i = 1$ , the condition exist for the carry to bypass the entire group:

$$P = p_i \cdot p_{i+1} \cdot p_{i+2} \cdot \dots \cdot p_{i+k-1}$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

# Carry Skip Adder



# Carry Skip Adder

The **Carry Skip Adder** (CSKA) divides the words to be added into groups of equal size of **k-bits**.

The basic structure of an **N-bit Carry Skip Adder**

Within the group, carry propagates in a ripple-carry fashion.

In addition, an AND gate is used to form the **group propagate** signal **P**.

$$P = p_i \cdot p_{i+1} \cdot p_{i+2} \cdot \dots \cdot p_{i+k-1}$$

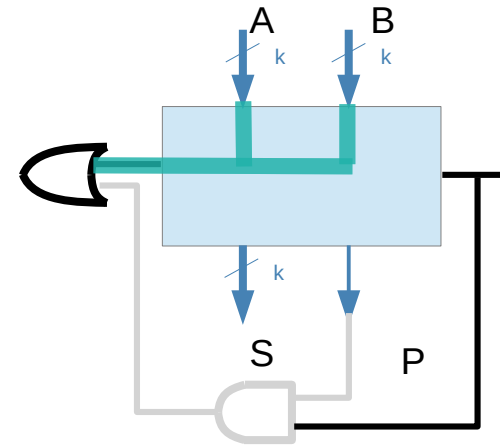
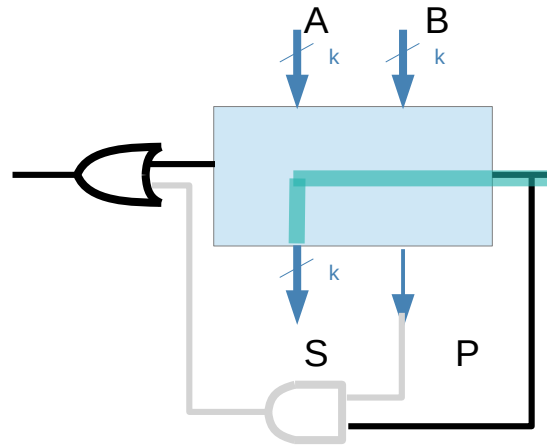
If  $P = 1$  the condition exists for carry to bypass (skip) over the group

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

# Carry Skip Adder

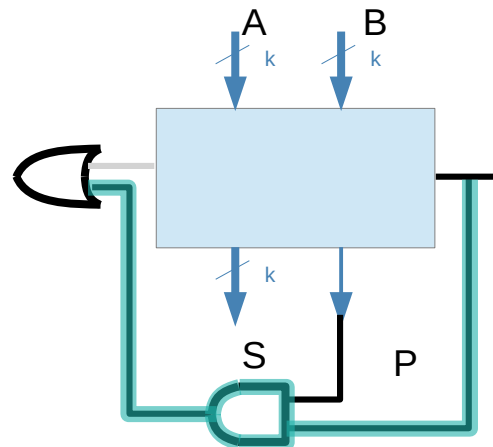
Ripple through  $k-1$  bits

$$(k-1) \Delta_{rca}$$



Skip carry

$$\Delta_{SKIP}$$



Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

# Carry Skip Adder

The maximal delay  $\Delta$  of a Carry Skip Adder is encountered when **carry** is generated in the **least-significant bit** position,

- rippling through  $k-1$  bit positions,
- skipping over  $R-2 = N/k-2$  groups in the middle,
- rippling to the  $k-1$  bits of most significant group and
- being assimilated in the  $N$ -th bit position to produce the sum  $S_N$  :

$$\begin{aligned}\Delta_{\text{CSA}} &= (k - 1) \Delta_{\text{rca}} + (R - 2) \Delta_{\text{SKIP}} + (k - 1) \Delta_{\text{rca}} \\ &= 2 (k - 1) \Delta_{\text{rca}} + (R - 2) \Delta_{\text{SKIP}} \\ &= 2 (k - 1) \Delta_{\text{rca}} + (N/k - 2) \Delta_{\text{SKIP}}\end{aligned}$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

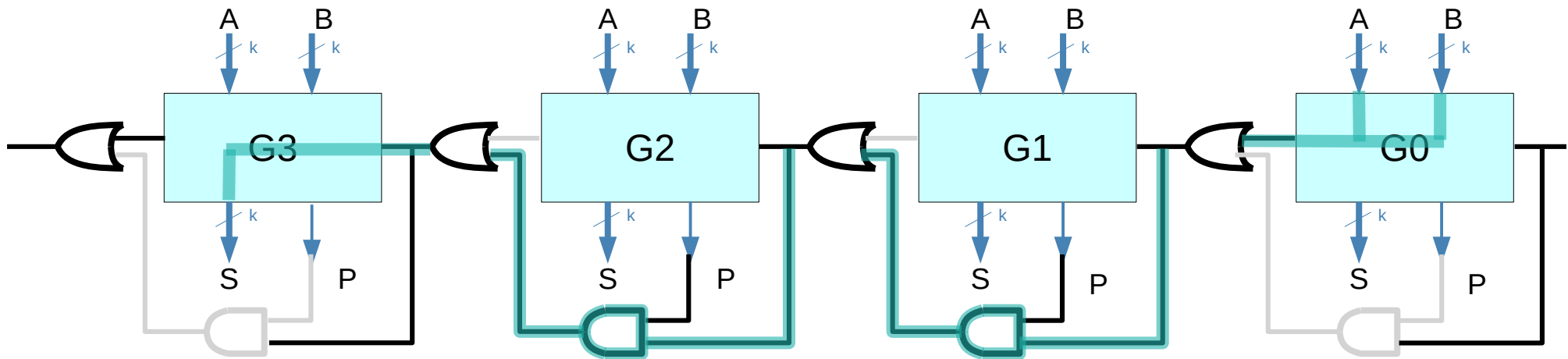
# Carry Skip Adder

$$\begin{aligned}\Delta_{\text{CSA}} &= (k - 1) \Delta_{\text{rca}} + (R - 2) \Delta_{\text{SKIP}} + (k - 1) \Delta_{\text{rca}} \\ &= 2(k - 1) \Delta_{\text{rca}} + (R - 2) \Delta_{\text{SKIP}} \\ &= 2(k - 1) \Delta_{\text{rca}} + (N/k - 2) \Delta_{\text{SKIP}}\end{aligned}$$

Carry Skip Adder is faster than RCA at the expense of a few relatively simple modifications.

The delay is still linearly dependent on the size of the adder  $N$ , however this linear dependence is reduced by a factor of  $1/k$

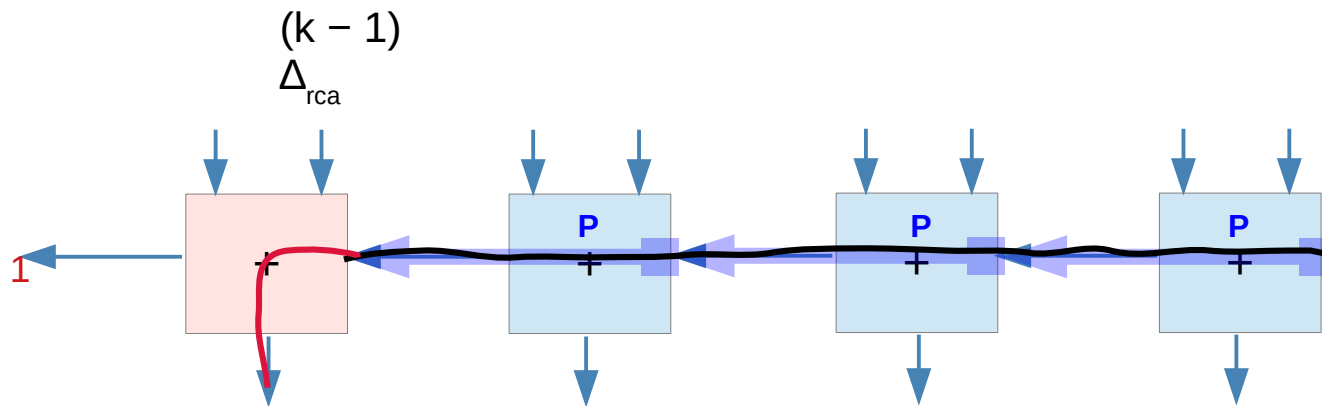
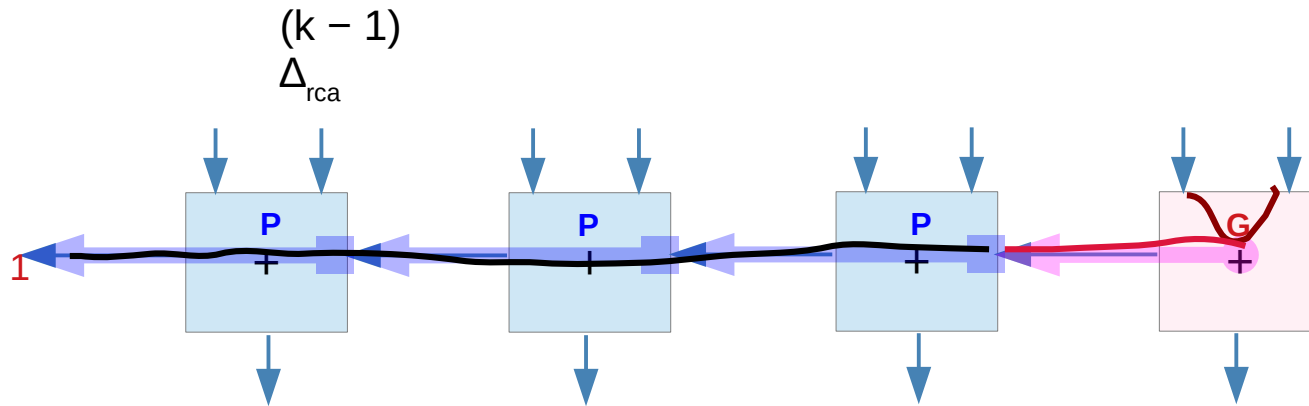
$$N = R \cdot k$$



Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers



# Design C (9) – When Cout1 = 1



High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

# Carry Skip Adder

---

If an arbitrary block generated a carry by itself,  
The carry will always propagate to the next block  
However, if the second block generates a carry itself,  
Or kill the carry, then that is the end of the critical path

If the second block propagates the carry, then we see  
The advantage of the CSA architecture

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

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

---

## References

- [1] [en.wikipedia.org](http://en.wikipedia.org)
- [2] Parhami, “Computer Arithmetic Algorithms and Hardware Designs”