Revisit Half Adder and Full Adder

- Half adder (HA) adds two binary bits
- Full adder (FA) adds three binary bits (Why do we need this?)

Half Adder Truth Table

A	*B*	

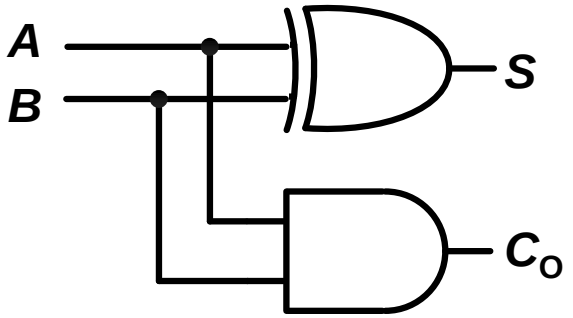
Full Adder Truth Table

A	*B*	C_I	C_O	*S*

Revisit Circuit Design for HA

Half Adder Truth Table

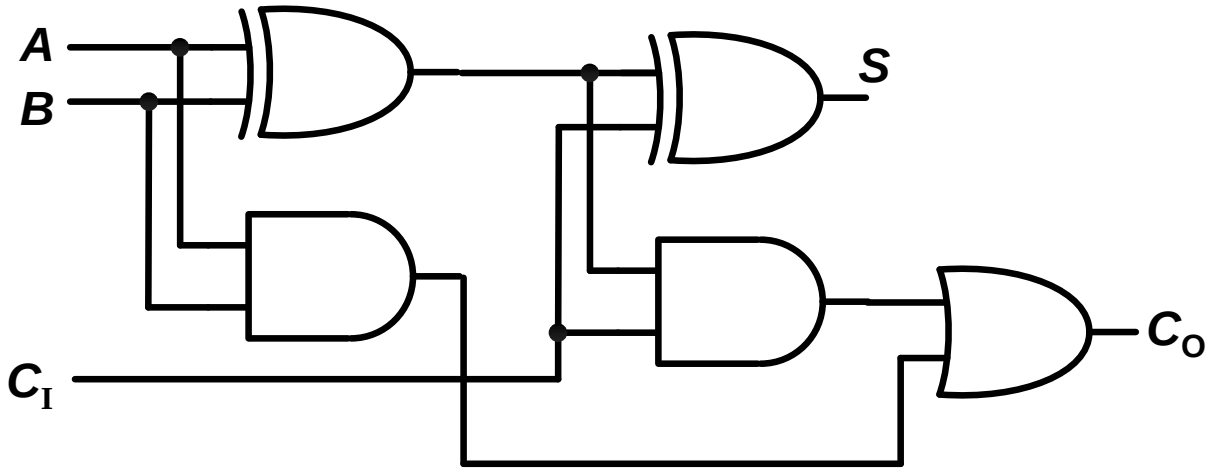
A	*B*	C_O	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Revisit Circuit Design for FA

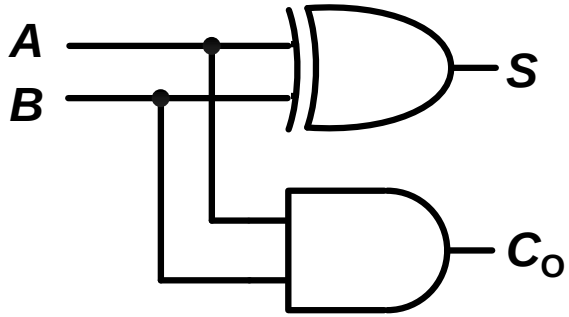
Full Adder Truth Table

A	B	C_I	C_O	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

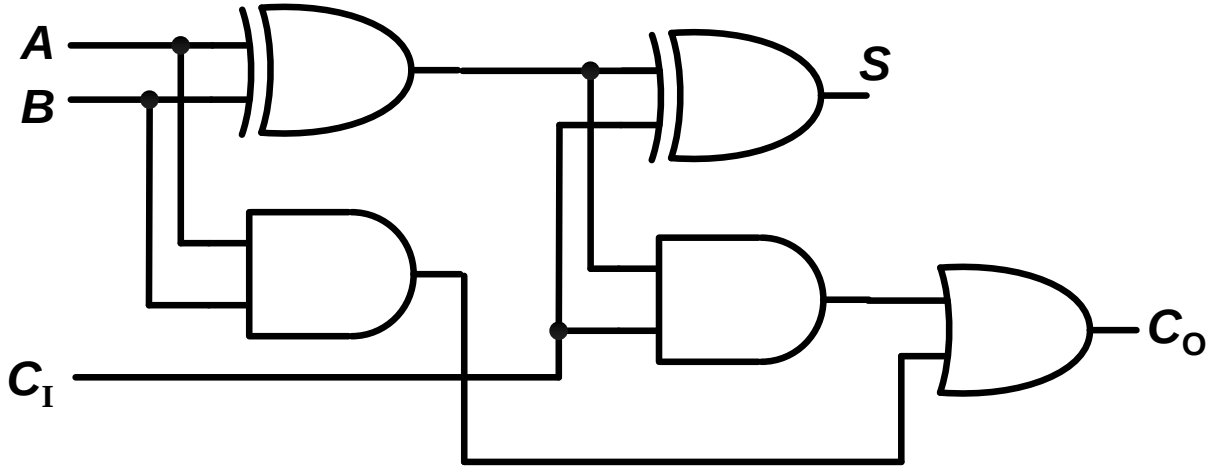


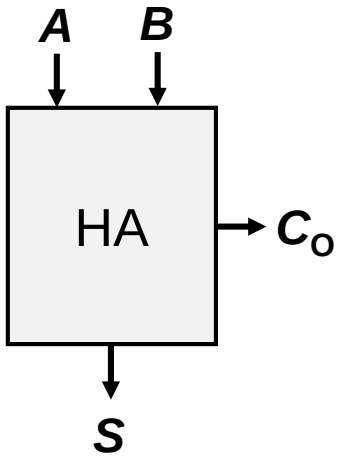
Symbolize HA & FA

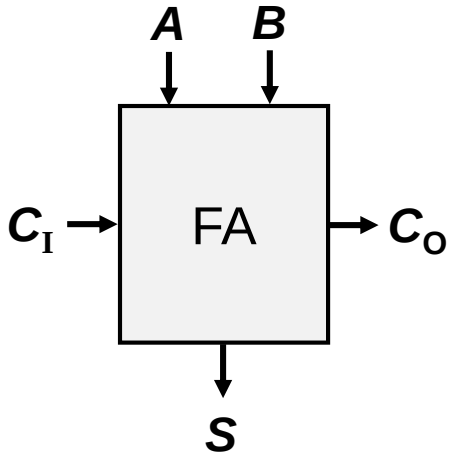
Half Adder



Full Adder

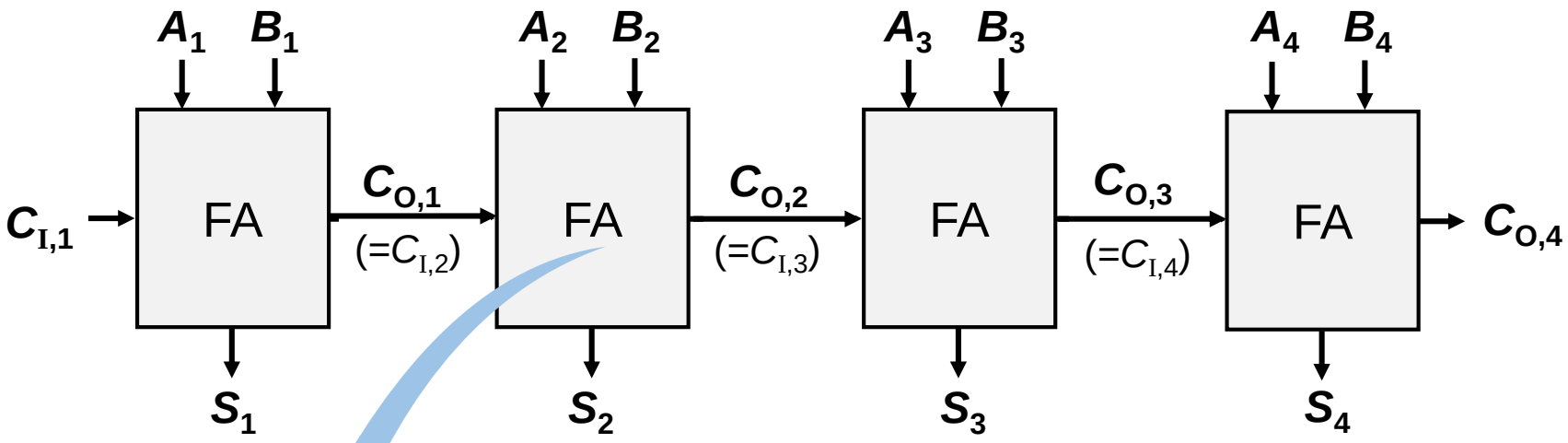


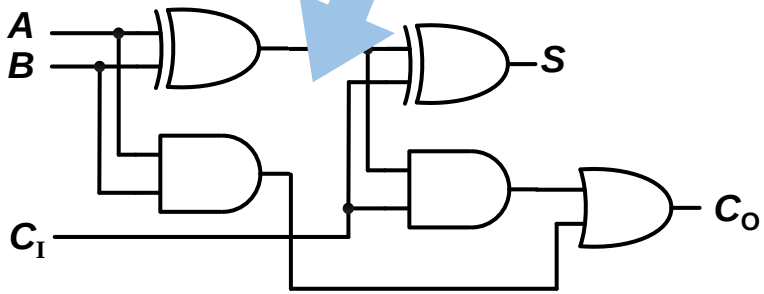




Multibit Data Addition; 4bit
Carry Ripple Adder

- $A_4A_3A_2A_1 + B_4B_3B_2B_1 = C_{O,4}S_4S_3S_2S_1$





Adder Delay t_{Adder}=?

$t_{Adder} = (N-1)t_{Carry} + t_{Sum}$

Can't we make this faster?

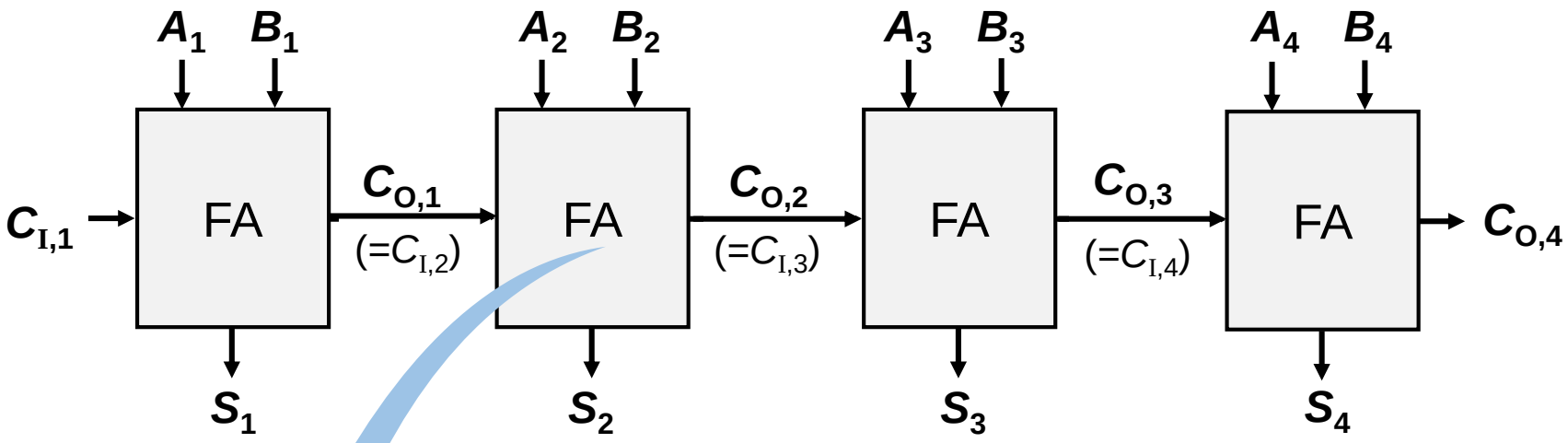
Adder Circuits (1)

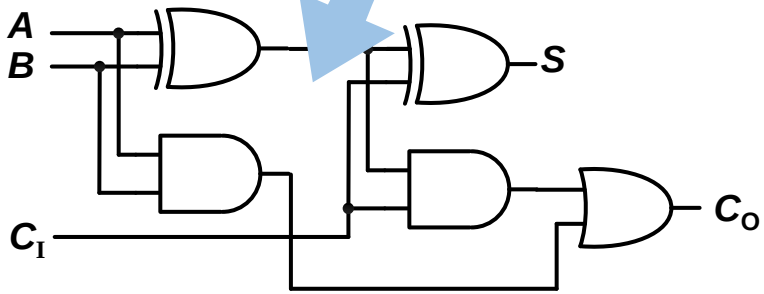
Hanwool Jeong

hwjeong@kw.ac.kr

Multibit Data Addition; 4bit
Carry Ripple Adder

- $A_4A_3A_2A_1 + B_4B_3B_2B_1 = C_{O,4}S_4S_3S_2S_1$



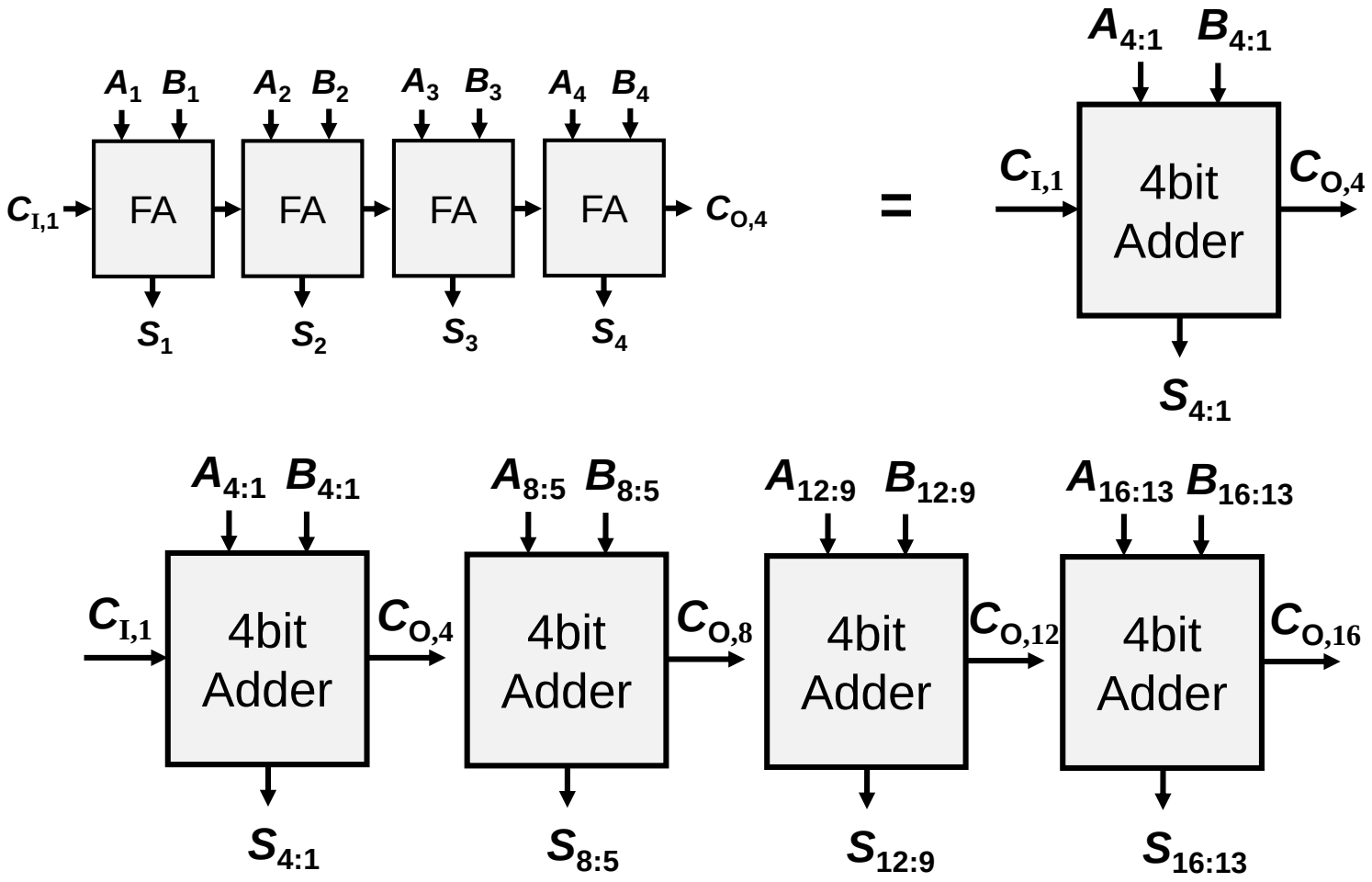


Adder Delay t_{Adder}=?

$t_{Adder} = (N-1)t_{Carry} + t_{Sum}$

Can't we make this faster?

Implementation of 16 Bit Adder w/ 4bit Adder



Ripple? Very Slow!

Prefix Definition

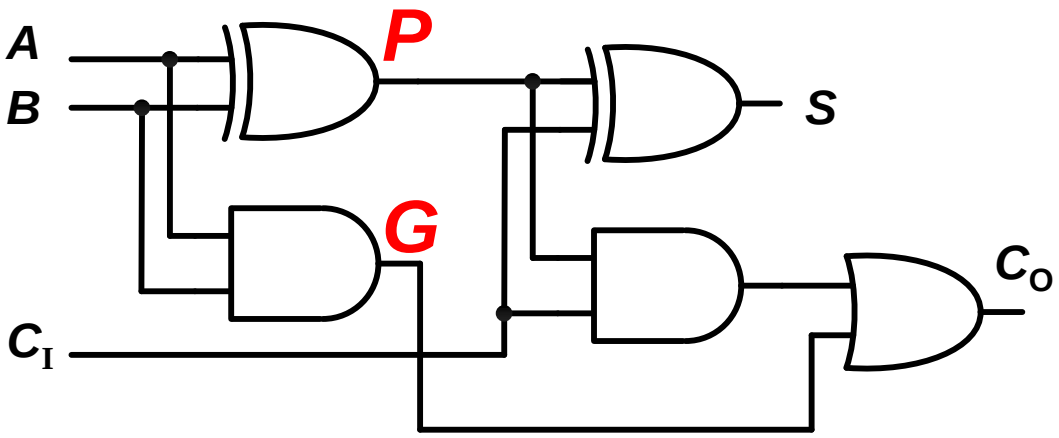
- For each stage *k*,

$$C_{O,k} = A_k B_k + C_{I,k} (A_k \oplus B_k)$$

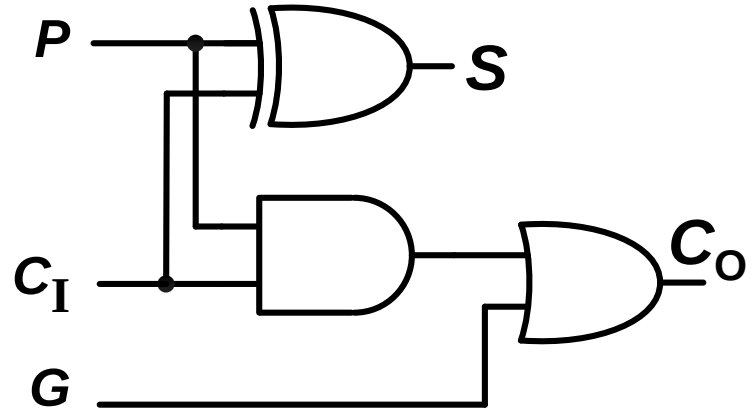
- If we define prefix P_k and G_k at k^{th} stage,

$$P_k = A_k \oplus B_k,\ G_k = A_k B_k,$$

$$C_{O,k} = G_k + P_k C_{I,k}$$



Full adder



Carryout Evaluation through P&G Prefix

- We saw that CO can be evaluated as follows

$$C_{O,1} = G_1 + P_1C_{I,1} = C_{I,2}$$

$$C_{O,2} = G_2 + P_1C_{I,1} = C_{I,3} = G_2 + P_2(\mathbf{G_1 + P_1C_{I,1}})$$

$$C_{O,3} = G_3 + P_3C_{I,2} = C_{I,4} = G_3 + P_3(G_2 + P_2C_{I,2})$$

$$= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_{I,1}$$

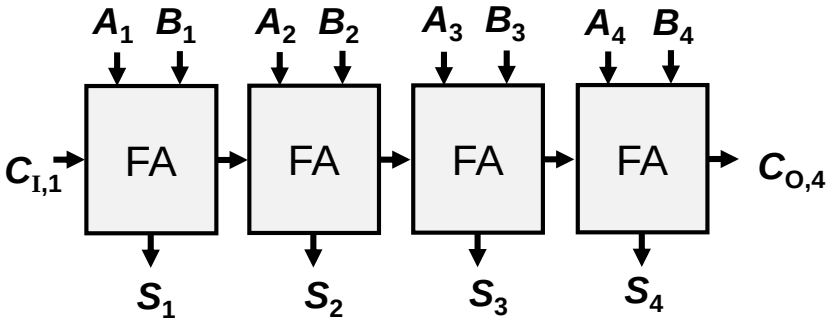
$$C_{O,4} = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_{I,1}$$

Lower level Implementation of Carry Lookahead Adder

- $C_{,O3}$ can be generated with **P & G generation + two-level**!!

$$C_{O,4} = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_{I,1}$$

Faster than Carry ripple adder?

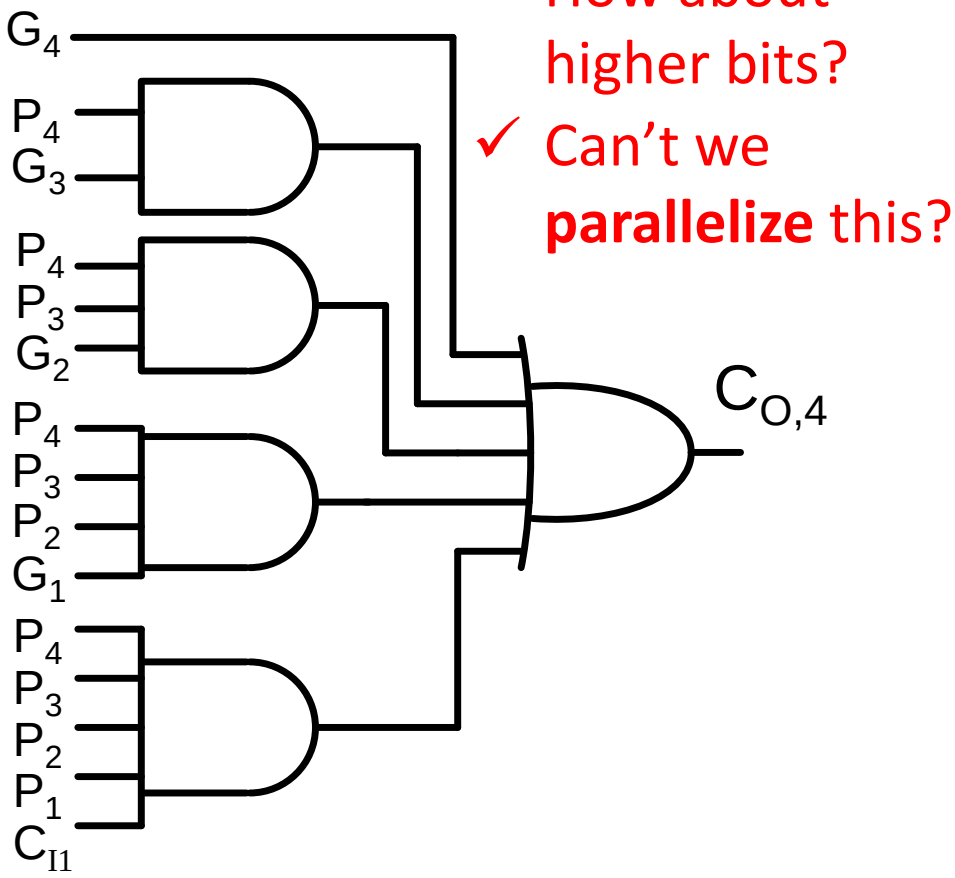


➔ **Not necessarily**

Depends on how to implement AND4 and AND5 operation.

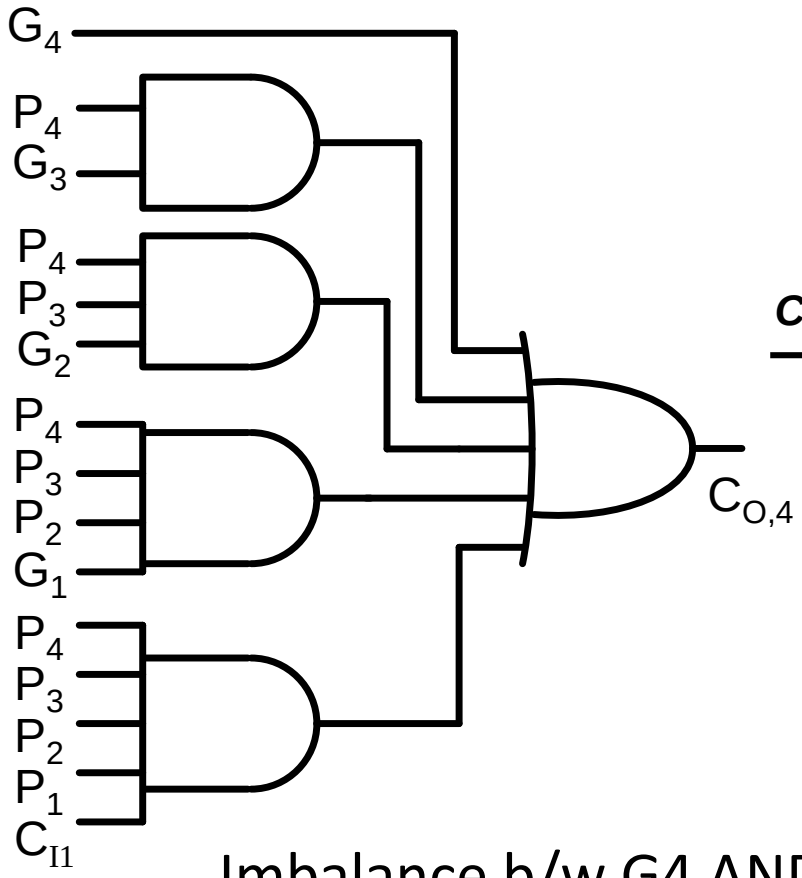
✓ How about higher bits?

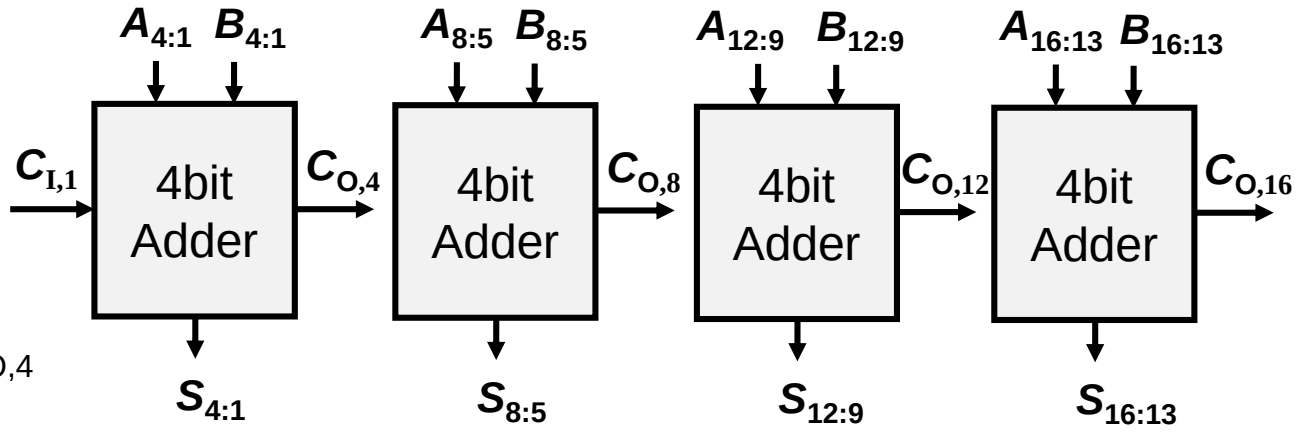
✓ Can't we **parallelize** this?



Imagine Higher Bit Situation

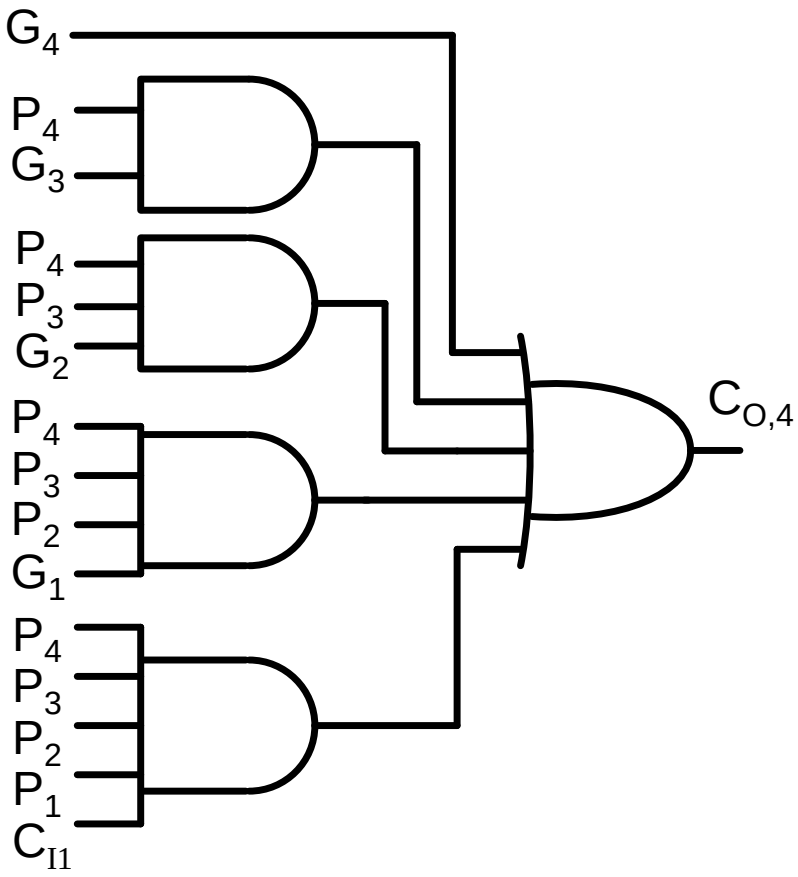
- You will use larger input-width AND & OR?

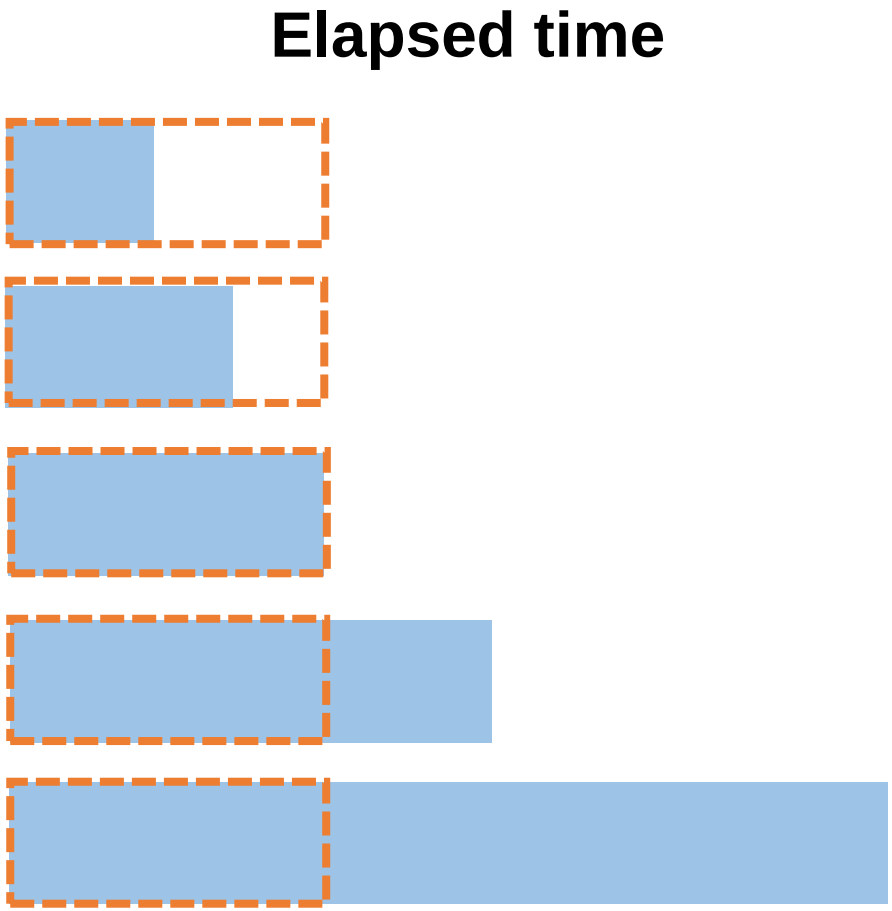




Imbalance b/w G4 AND2, AND3, … AND5

Imagine Higher Bit Situation





Systematical Methodology for Gradually Merging P_k & G_k

- 8bit example inputs A & B:

  (A_8, B_8) (A_7, B_7) (A_6, B_6) (A_5, B_5) (A_4, B_4) (A_3, B_3) (A_2, B_2) (A_1, B_1) $+C_i$

- Creating G & P

  (G_8, P_8) (G_7, P_7) (G_6, P_6) (G_5, P_5) (G_4, P_4) (G_3, P_3) (G_2, P_2) (G_1, P_1) $+C_i$

  $C_{O,8} = G_8 + P_8G_7 + P_8P_7G_6 + P_8P_7P_6G_5 + P_8P_7P_6P_5G_4 + \ldots$

  Is there any better method to efficiently merge them?
  Instead of **accumulating** P & G, **parallel** merging them!

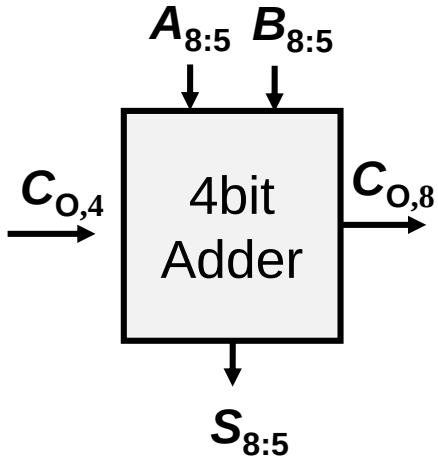
- Final goal:

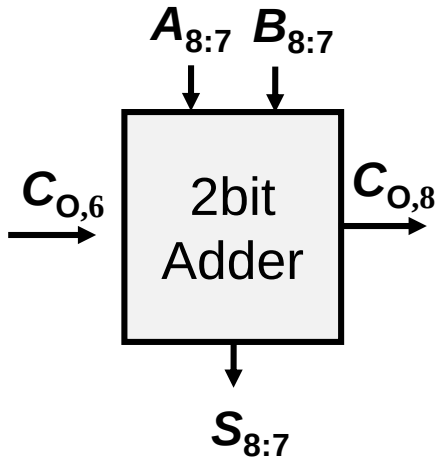
  $C_{O,8}$, $S_8S_7S_6S_5S_4S_3S_2S_1S_0$

Motivation for Group P&G; 2bit Case

- Can we define the meaning of $G_{8:5}$ and $P_{8:5}$? Or more simply, $G_{8:7}$ and $P_{8:7}$?
  - $G_{8:7}$ makes $C_{O,8} = 1$ independent of $C_{O,6}$ (= $C_{I,7}$)
  - $P_{8:7}$ passes $C_{O,6}$ (=needs external info)
- Boolean equation?
  - $G_{8:7} = A_8B_8 + P_8G_7 = G_8 + P_8G_7$
  - $P_{8:7} = P_8P_7$
- Then: $C_{O,8} = G_{8:7} + P_{8:7}C_{O,6}$

We can decide this only utilizing internal information $A_{8:7}$, $B_{8:7}$
We don't need external information!





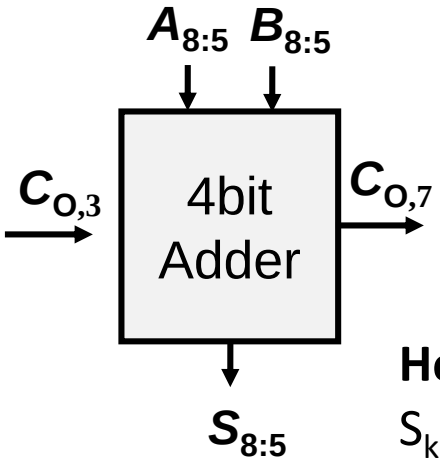
Group P&G; 4bit Case

- Can you derive $G_{8:5}$ and $P_{8:5}$? Focus on their meanings!

$G_{8:5}$ =

$P_{8:5}$ =

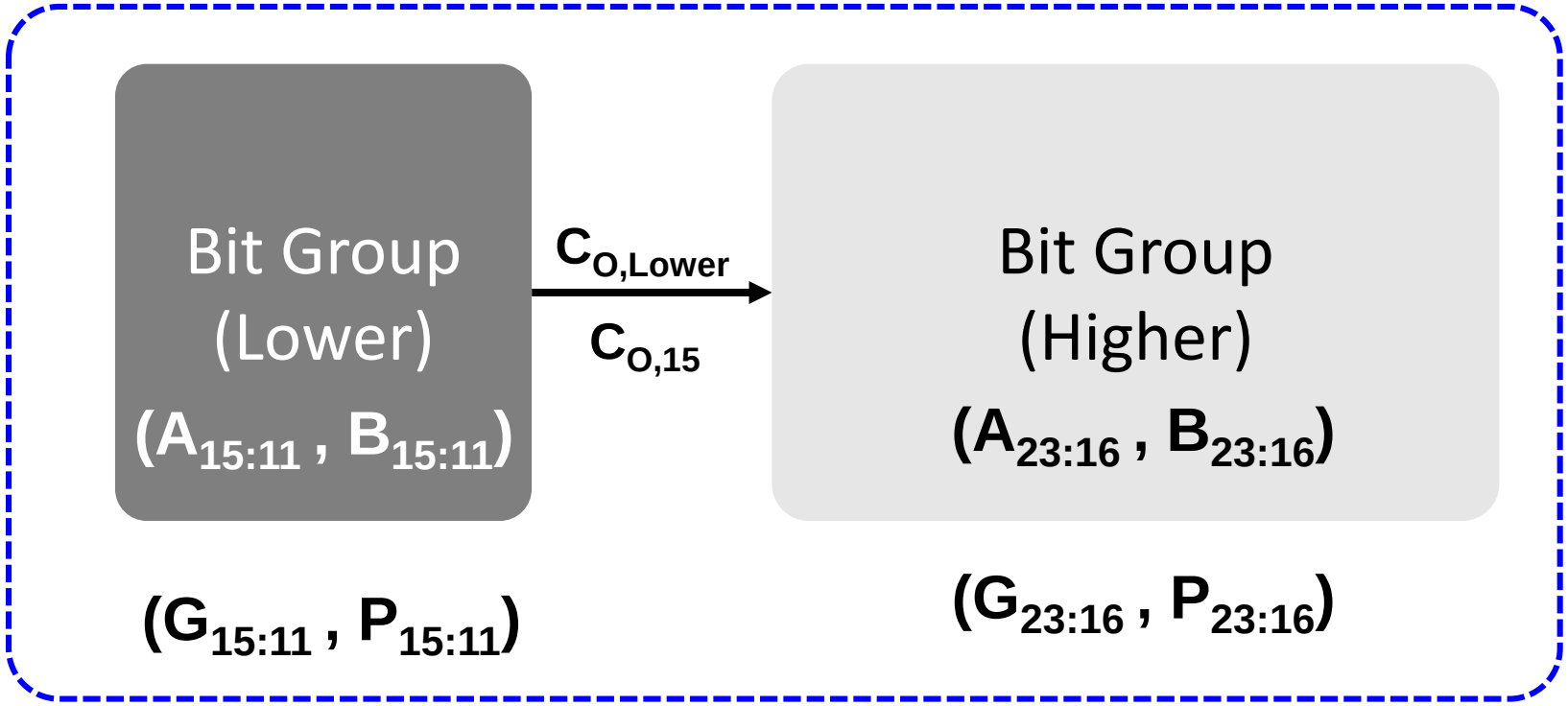
$C_{O,3}$ vs. $C_{O,7}$?



How about $S_{8:5}$?

$S_k = P_k \oplus C_{O,k-1}$

Getting a Feel With Group P&G Definition (1)



- ✓ Can we consider this as a group $(A_{23:11}, B_{23:11})$?
- ✓ Then, Can you derive $(G_{23:11}, P_{23:11})$?

4bit Group vs. 2bit Group

$$G_{8:5} = A_8B_8 + P_8A_7B_7 + P_8P_7A_6B_6 + P_8P_7P_6A_5B_5$$

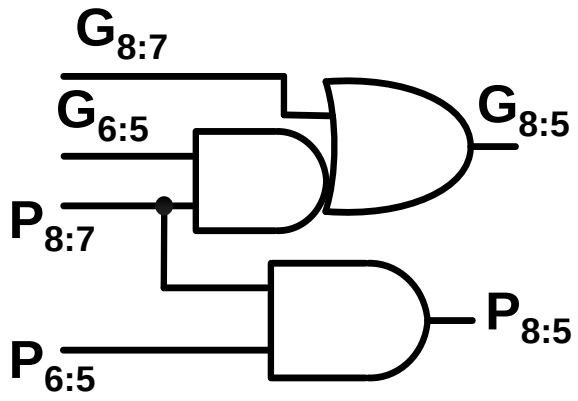
$$P_{8:5} = P_8\,P_7\,P_6\,P_5$$

$$G_{8:7} = A_8B_8 + P_8A_7B_7$$

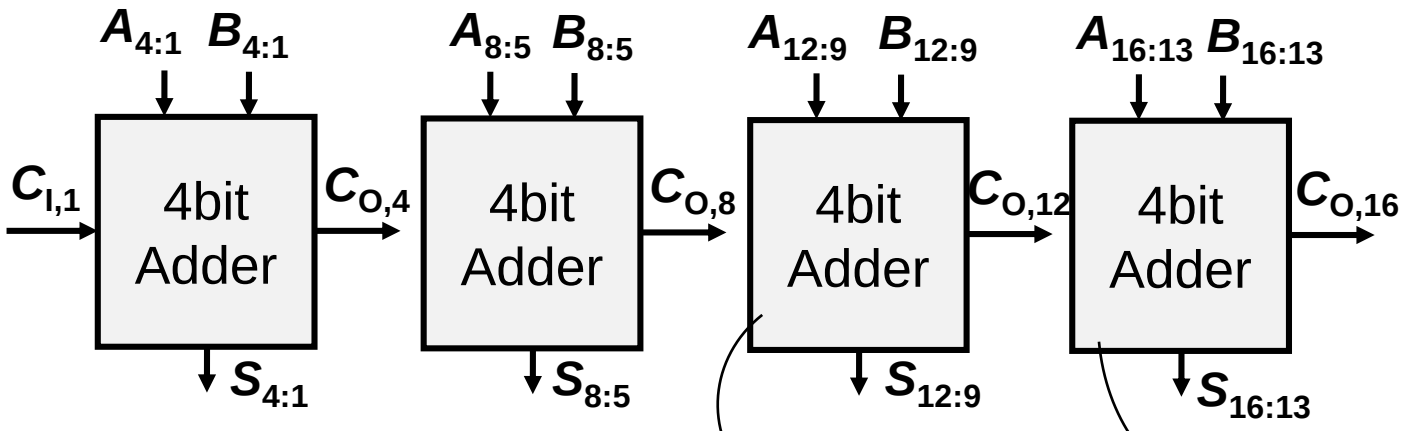
$$P_{8:7} = P_8\,P_7$$

$$\mathbf{G_{8:5} = G_{8:7} + P_{8:7}G_{6:5}}$$

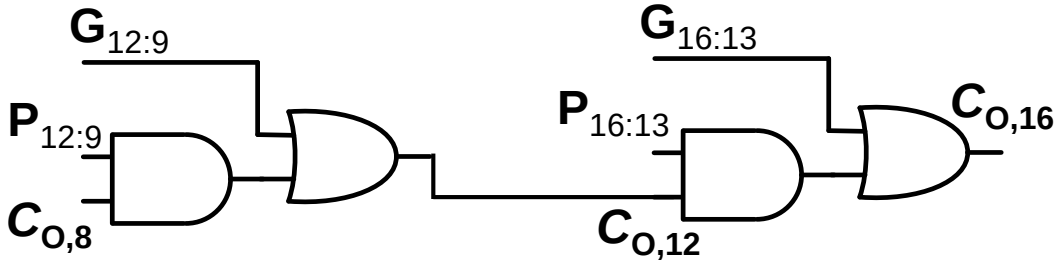
$$\mathbf{P_{8:5} = P_{8:7}P_{6:5}}$$



Getting a Feel With Group P&G Definition (2)

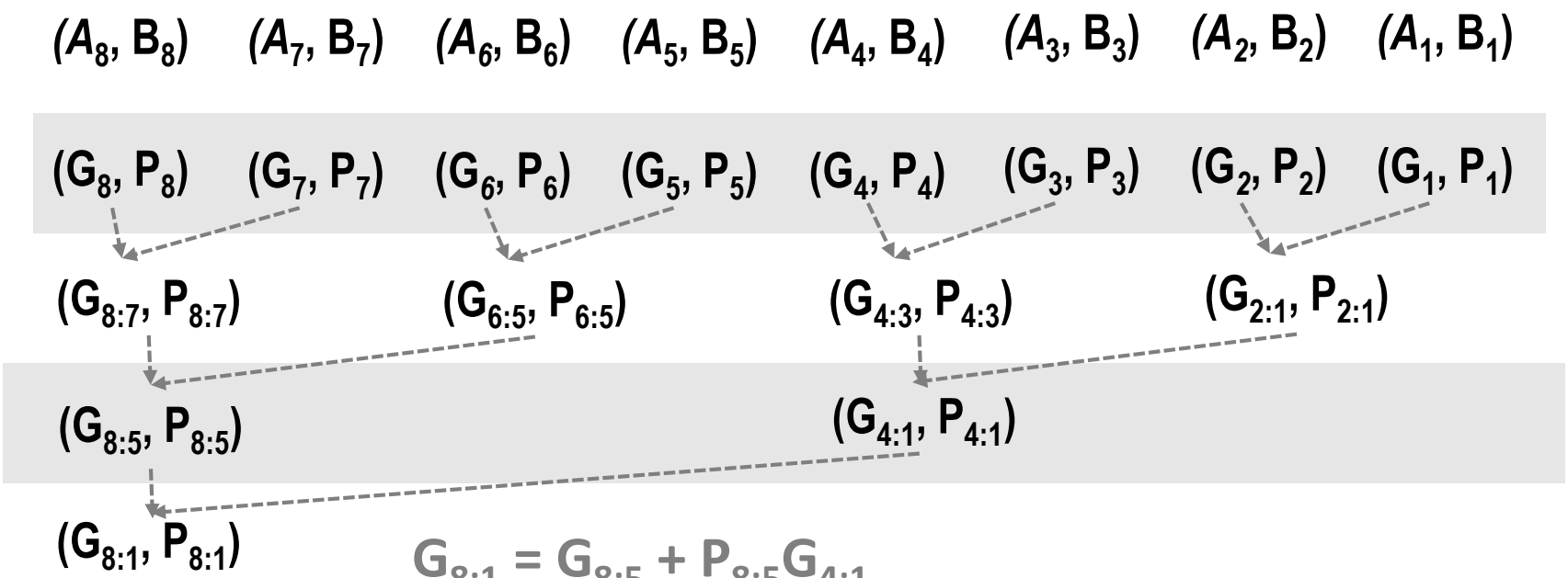


…



- ✓ **We can do the required procedures before carry-ins arrive**
- ✓ **Let's visualize this in timing axis from (A_k,B_k) to (S, C_O)**

Timing Axis in 8 Bit Addition Case



$$G_{8:1} = G_{8:5} + P_{8:5}G_{4:1}$$
$$P_{8:1} = P_{8:5}P_{4:1}$$

Check! Bear in Mind that

- G & P are not our goal; **Our goal is to derive C_O and S_k.**

$$S_k = P_k \oplus C_{O,k-1}$$

$$C_{O,k} = G_k + PC_{O,k-1} = \mathbf{G_{k:j} + P_{k:j}C_{O,j-1}}$$

- Focusing on C_O only, even you have $G_{8:1}$ and $P_{8:1}$, we should additional calculation:

$$C_{O,8} = \mathbf{G_{8:1} + P_{8:1}C_{I,1}}$$

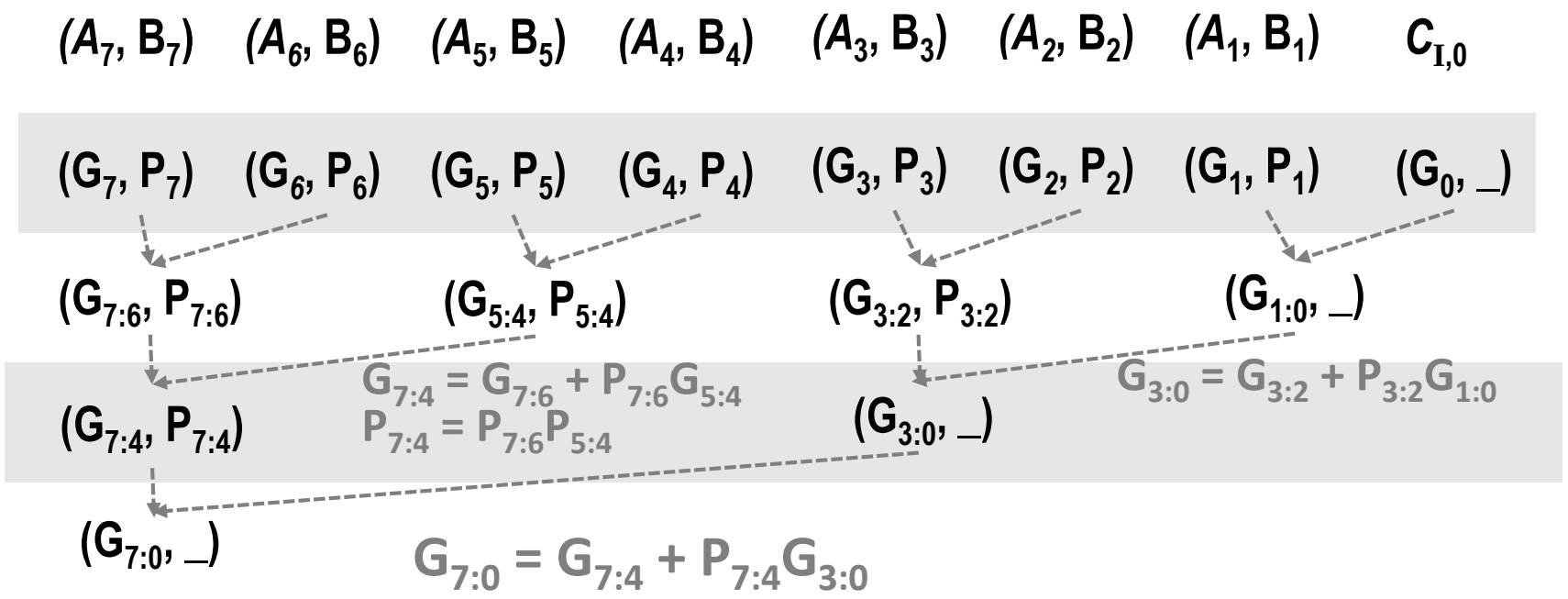
- Intuitively, anyway we need $C_{I,1}$. Can you relate $C_{I,1}$ to term **G**?

We can say $C_{I,1} = \mathbf{G_0}$

- Then, we can define $G_{8:0}$ as well. What does it mean?

$$C_{O,8} = \mathbf{G_{8:0}}$$

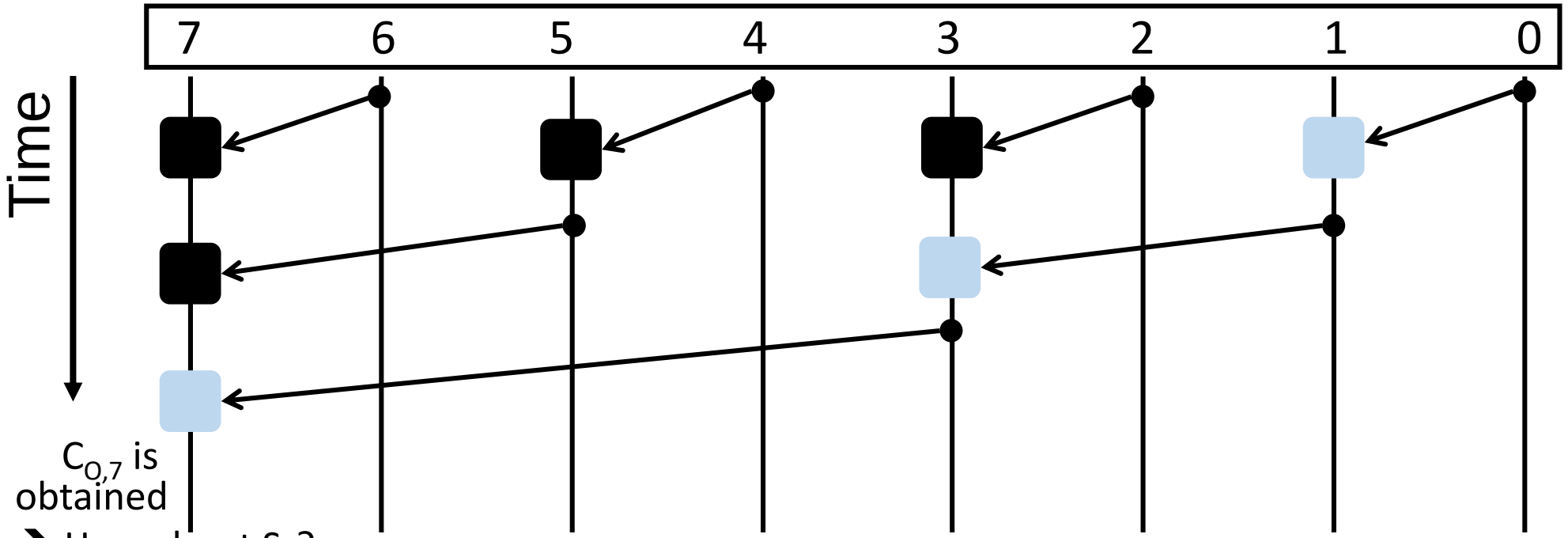
Timing Axis in 7 Bit Addition Case



We should evaluate S_k

That means we need $C_{O,k}$ for k=1~6

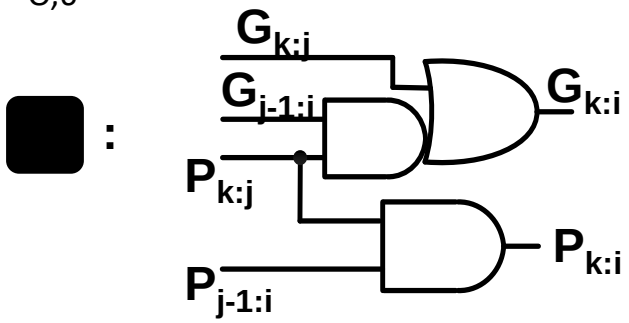
Simplified Representation; Tree Adder

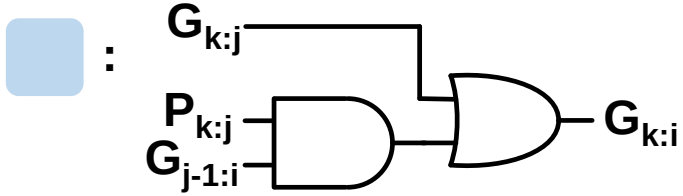


$C_{O,7}$ is obtained

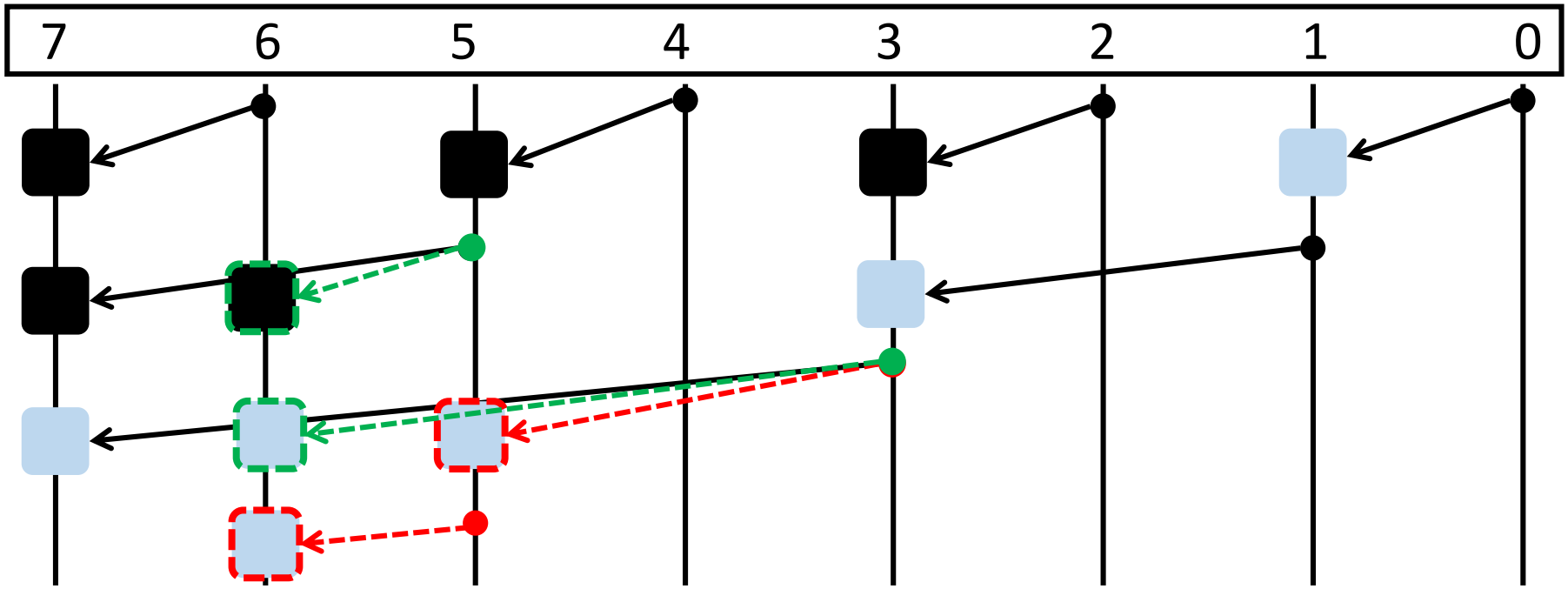
➔ How about S_7?

➔ We need $C_{O,6}$



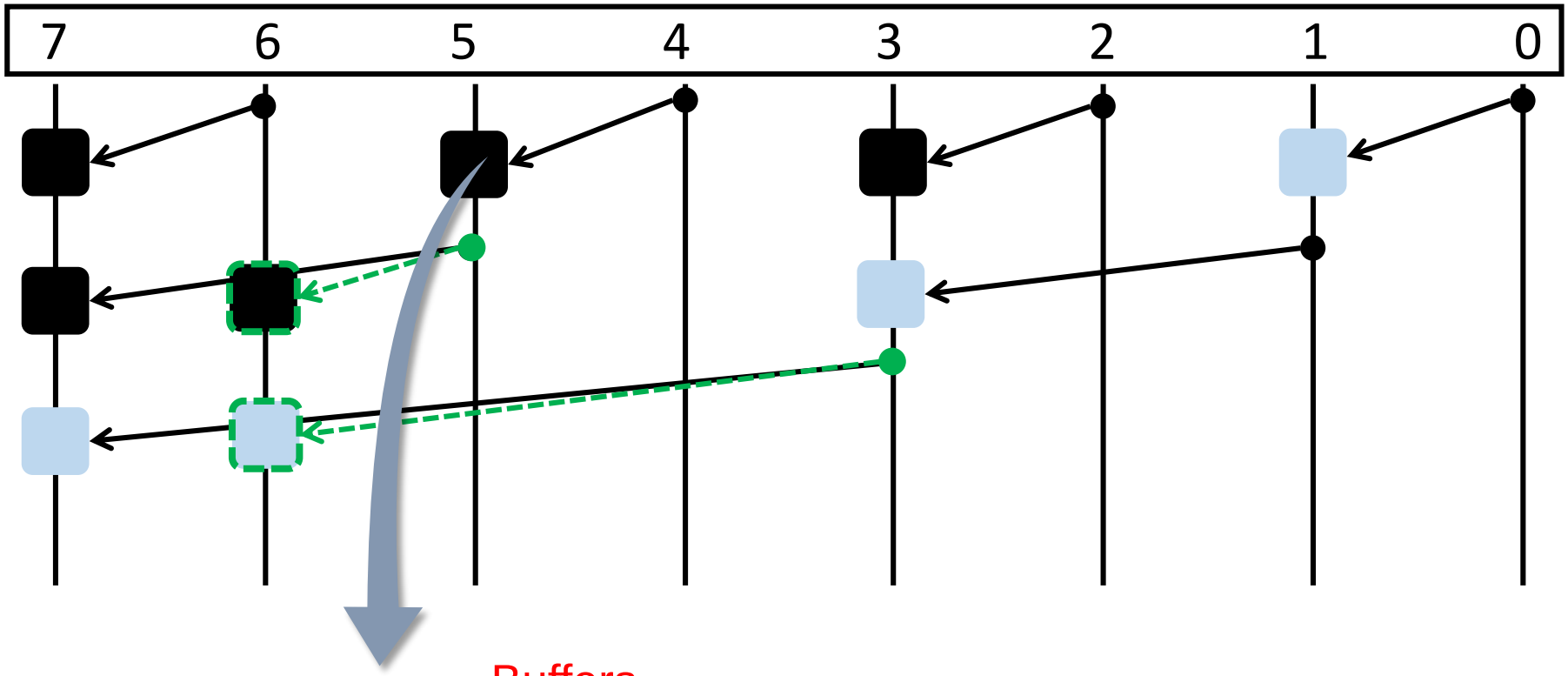


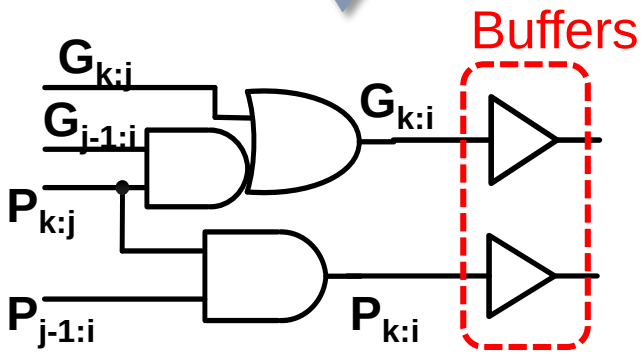
We Need $C_{O,6}$!



Is it okay?

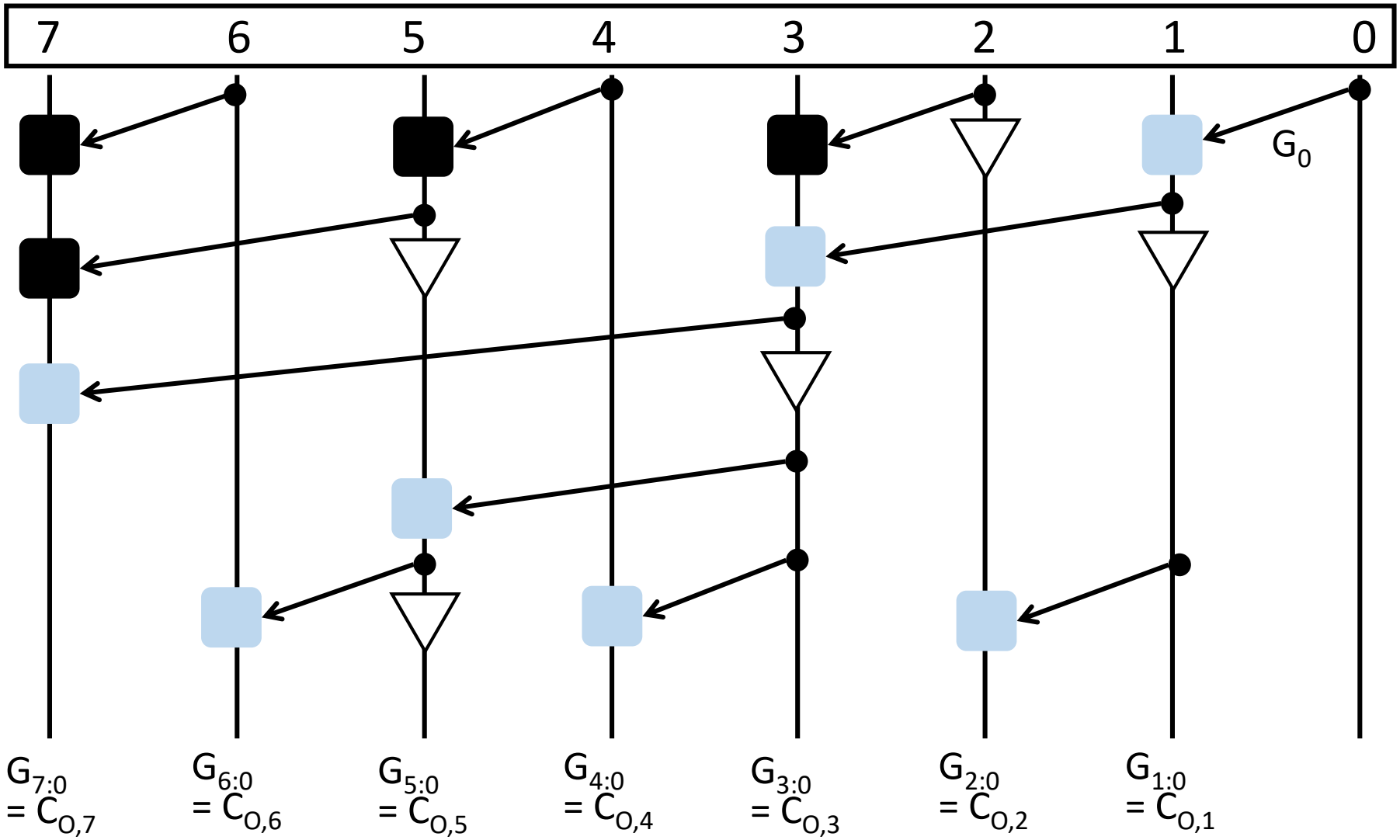
Considering Fan-out



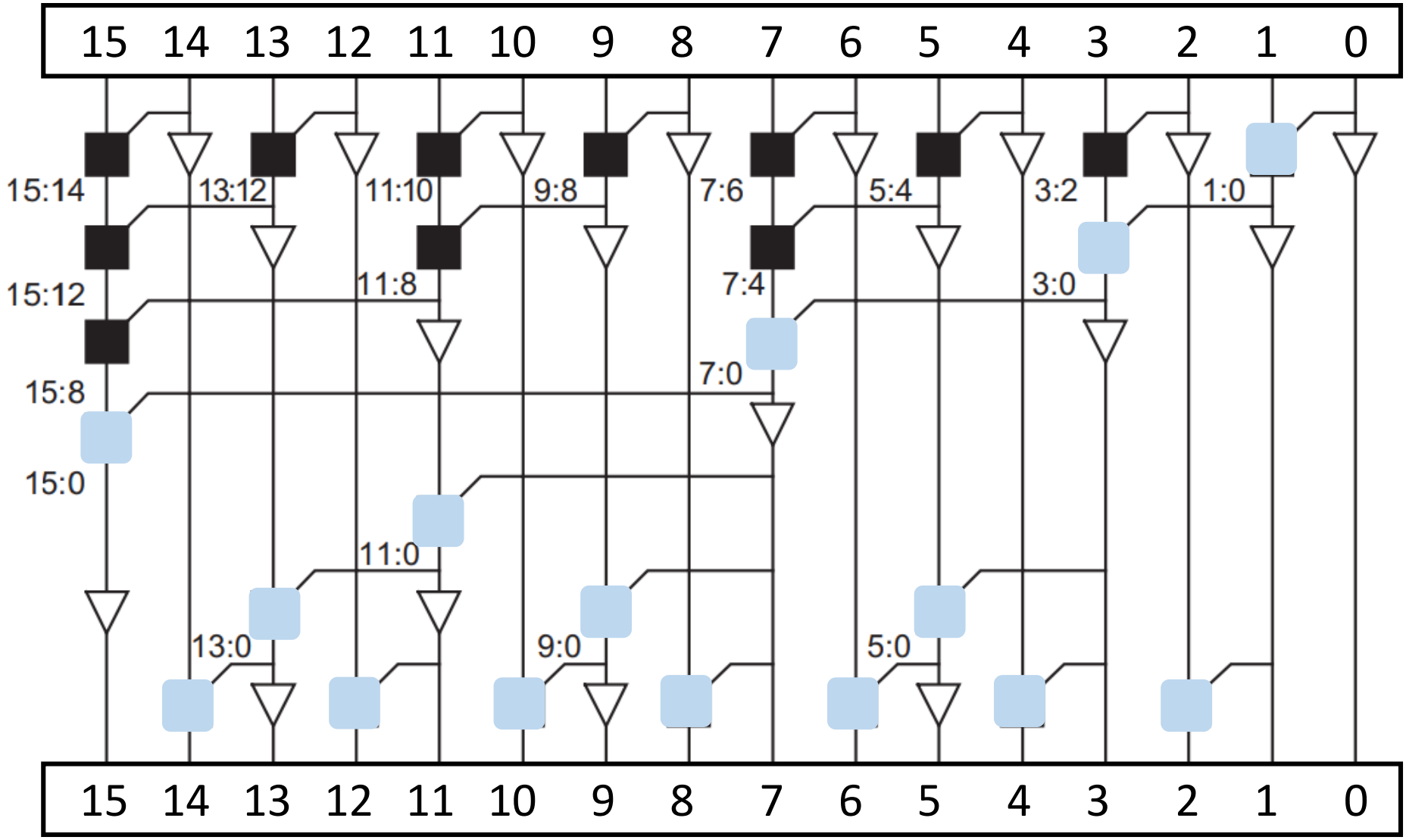


- ✓ Logic gates have finite driving strength.
- ✓ If too much load are connected to the gate, signal slope becomes very degraded ➔ slow
- ✓ Buffers can resolve this limitation.

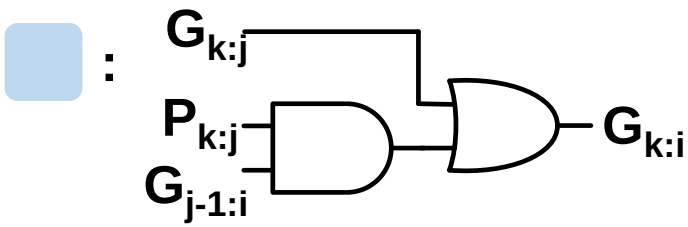
Brent-Kung Adder (1982)

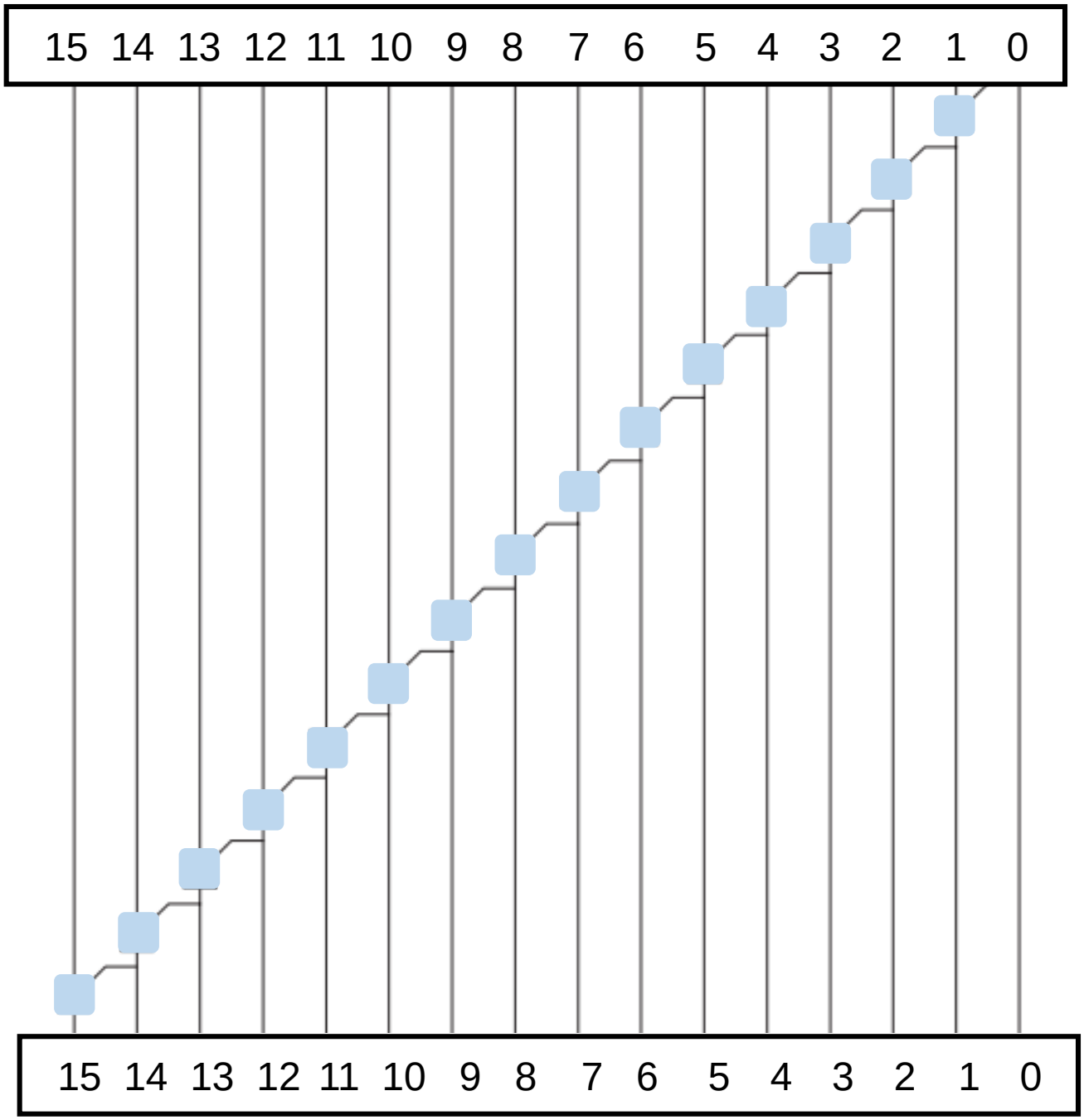


15-bit Configuration

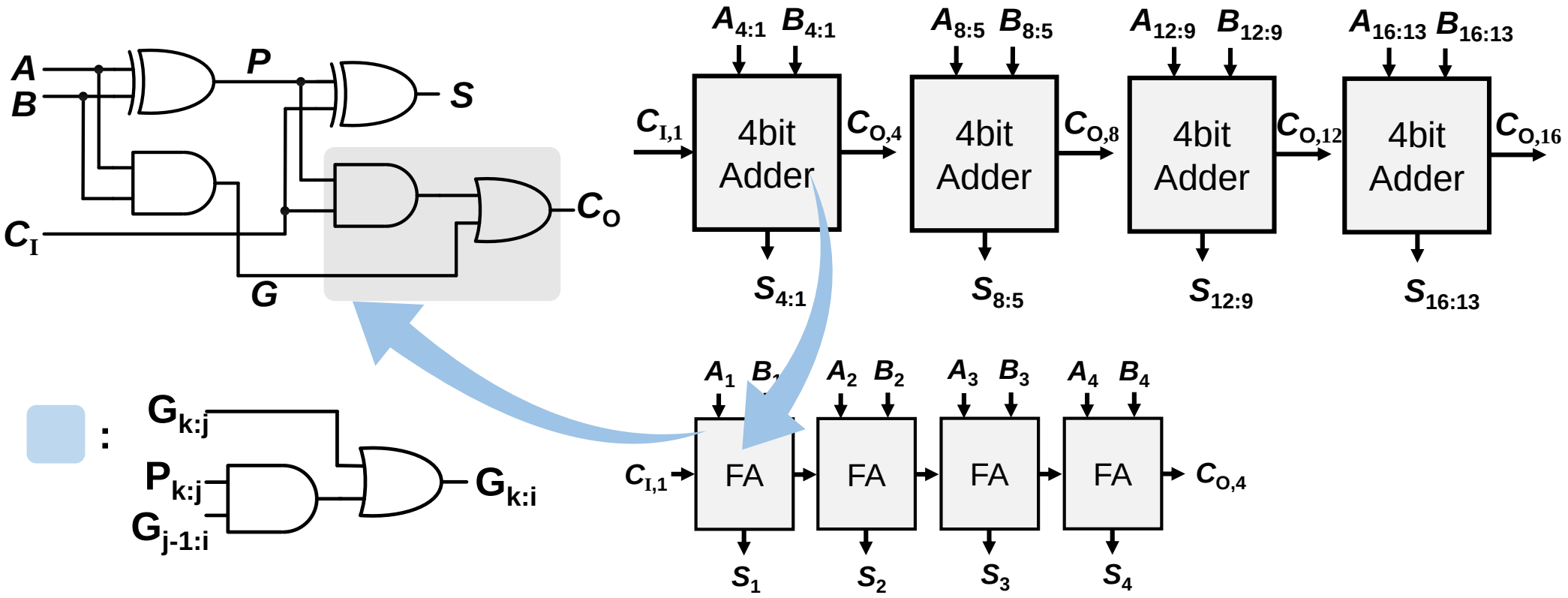


Carry Ripple?

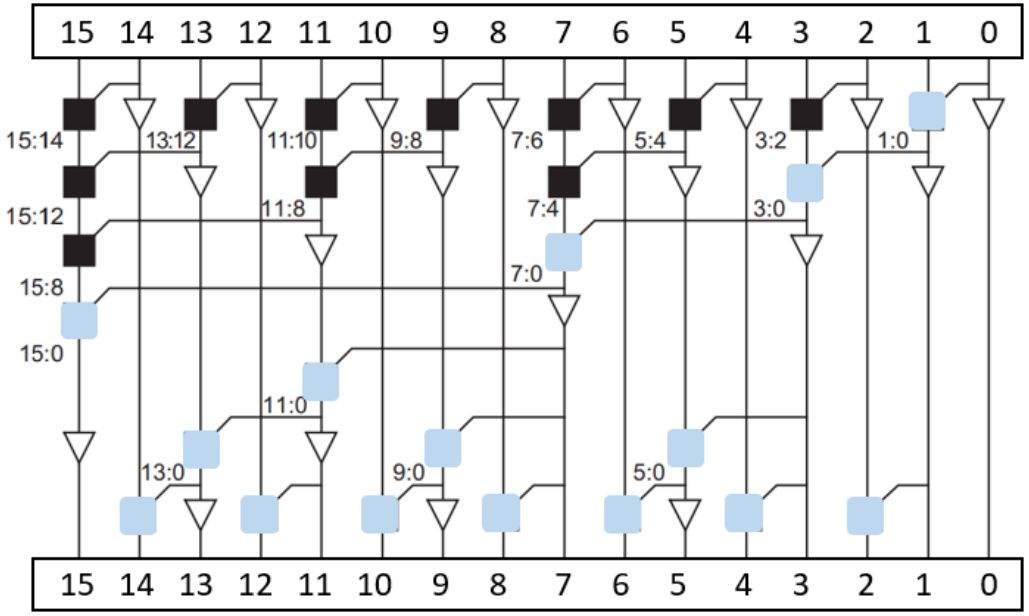




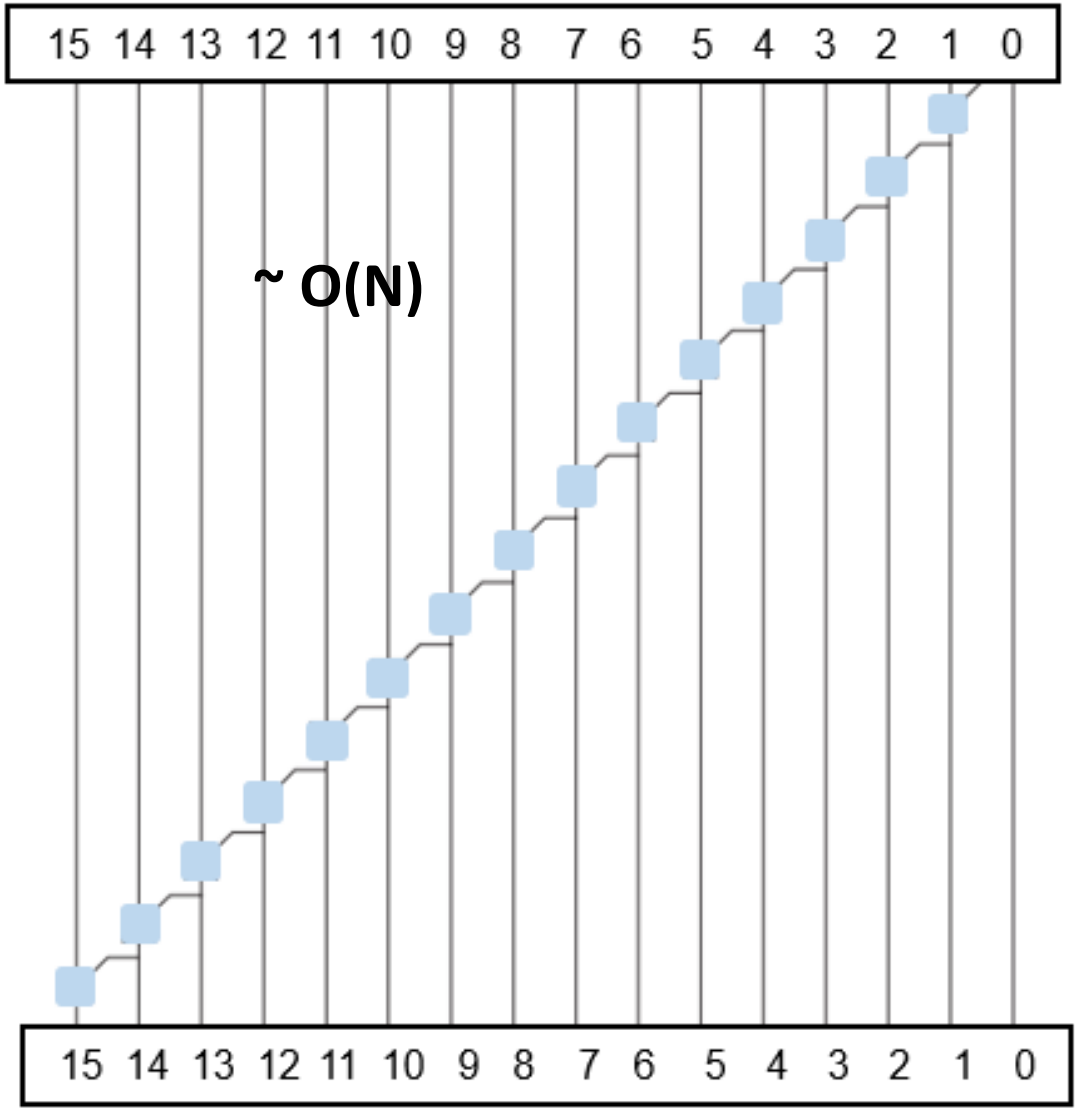
Revisit Carry Ripple Adder



Comparison



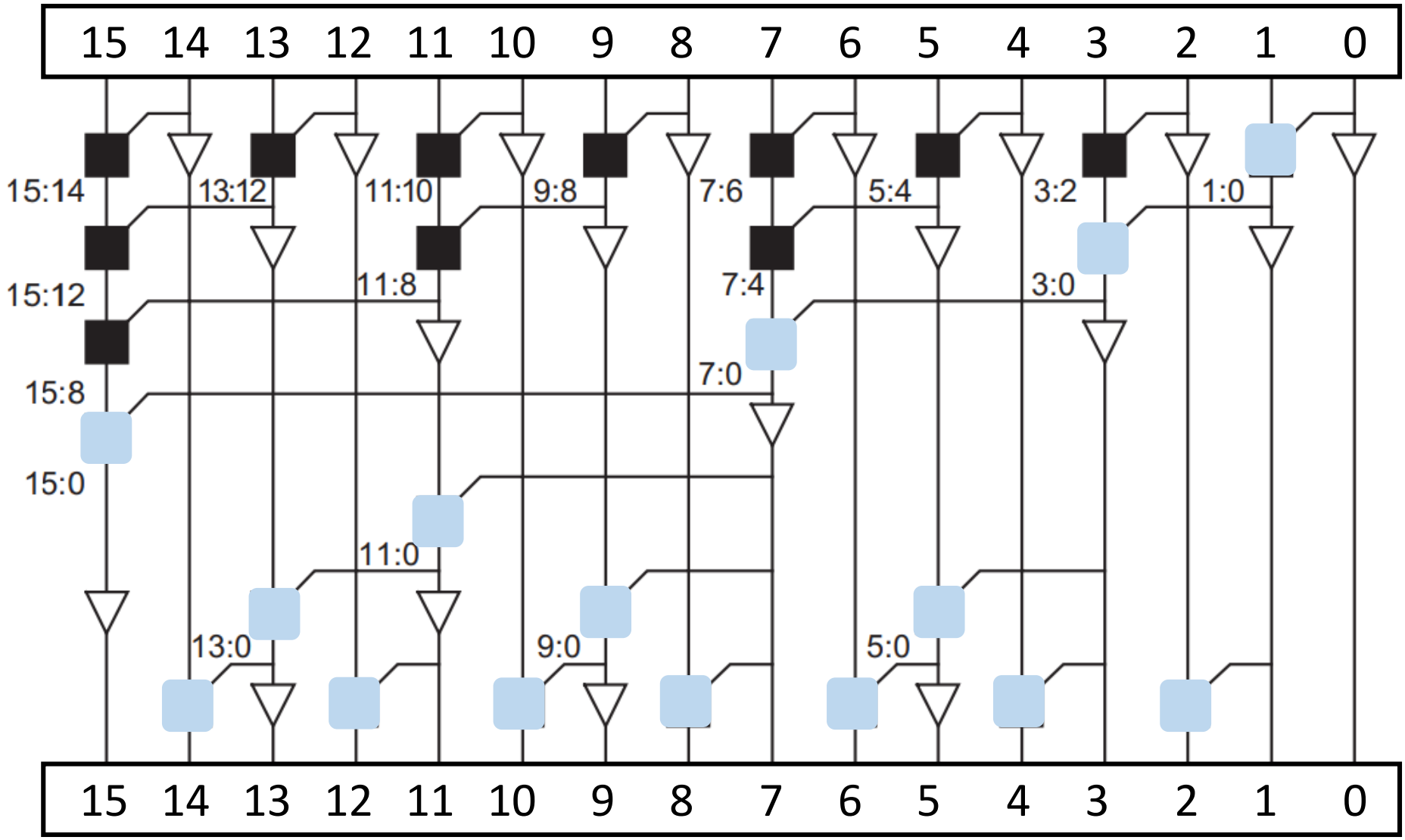
~ $O(\log_2 N)$?



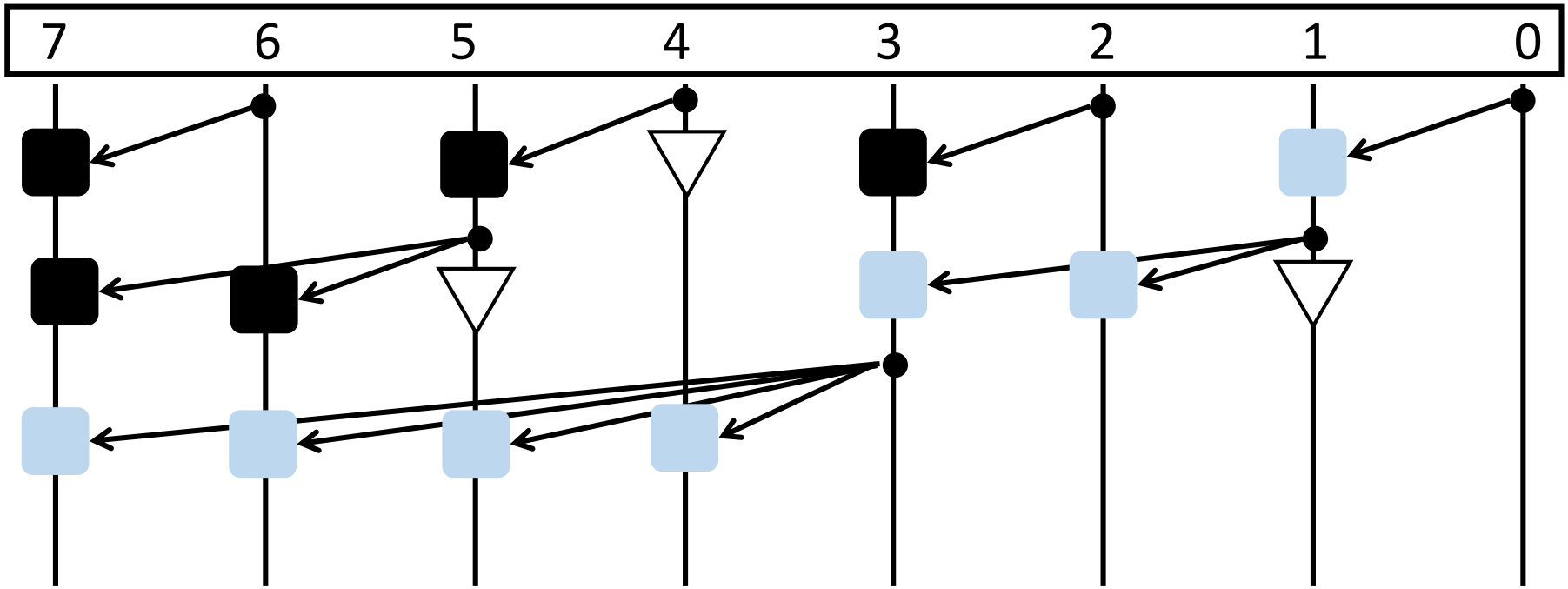
Checkpoints

- ✓Review P & G signal in carry lookahead
- ✓Limitation of P & G signals
- ✓Parallel approach
- ✓Tree adder
- ✓Coming-up next: various structure of tree adders

Too Many Logic Levels

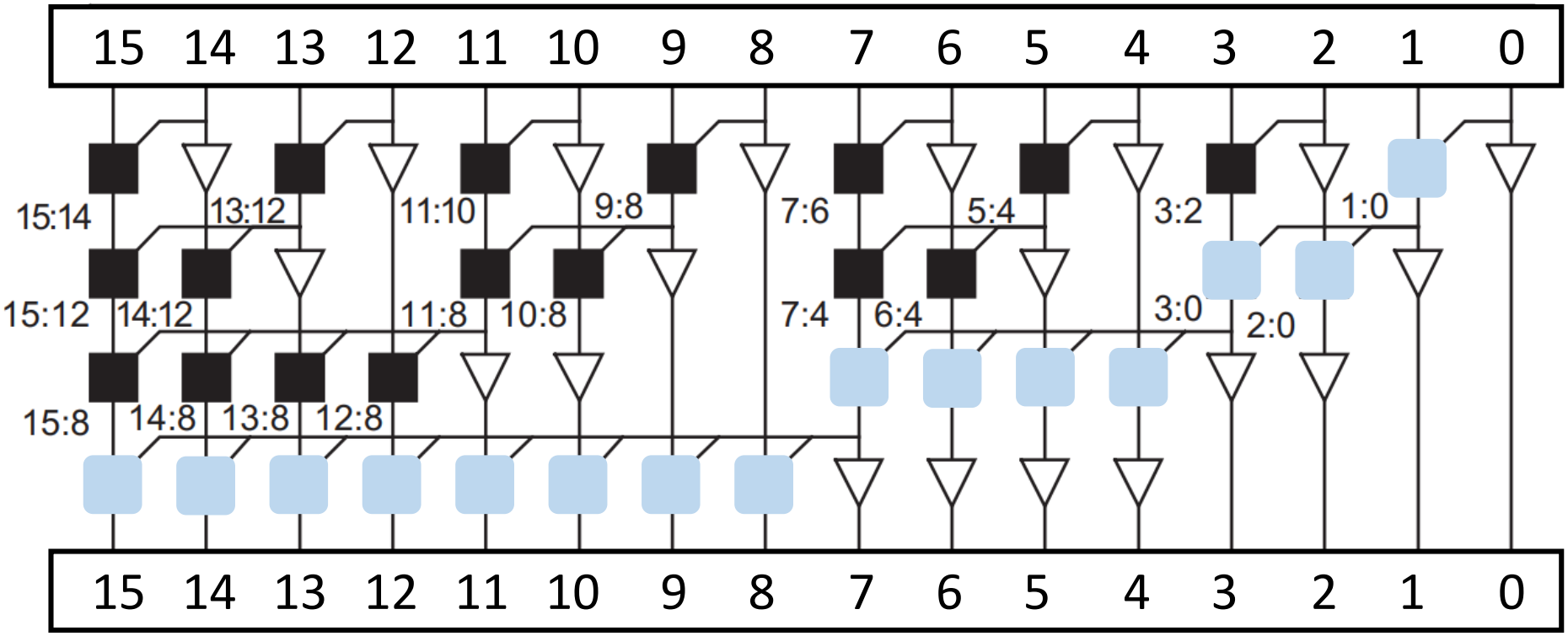


Sklansky Adder (1960)

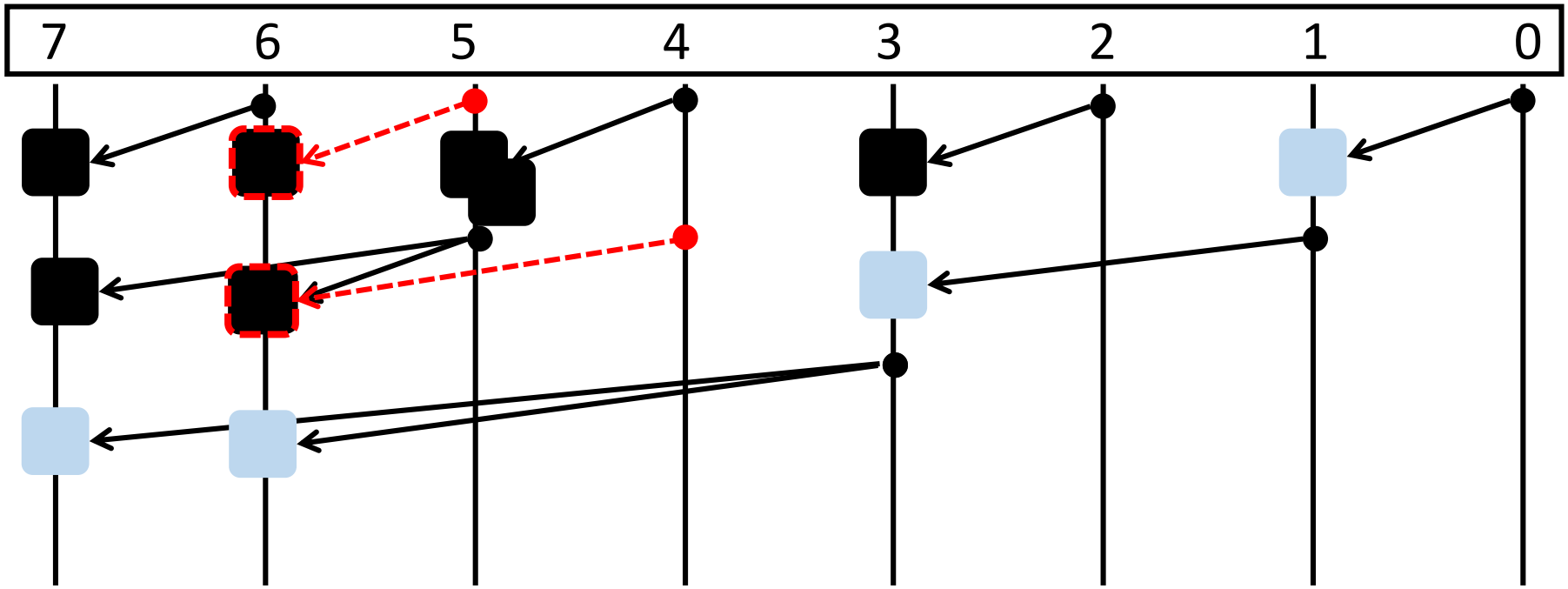


Logic level is priority!!

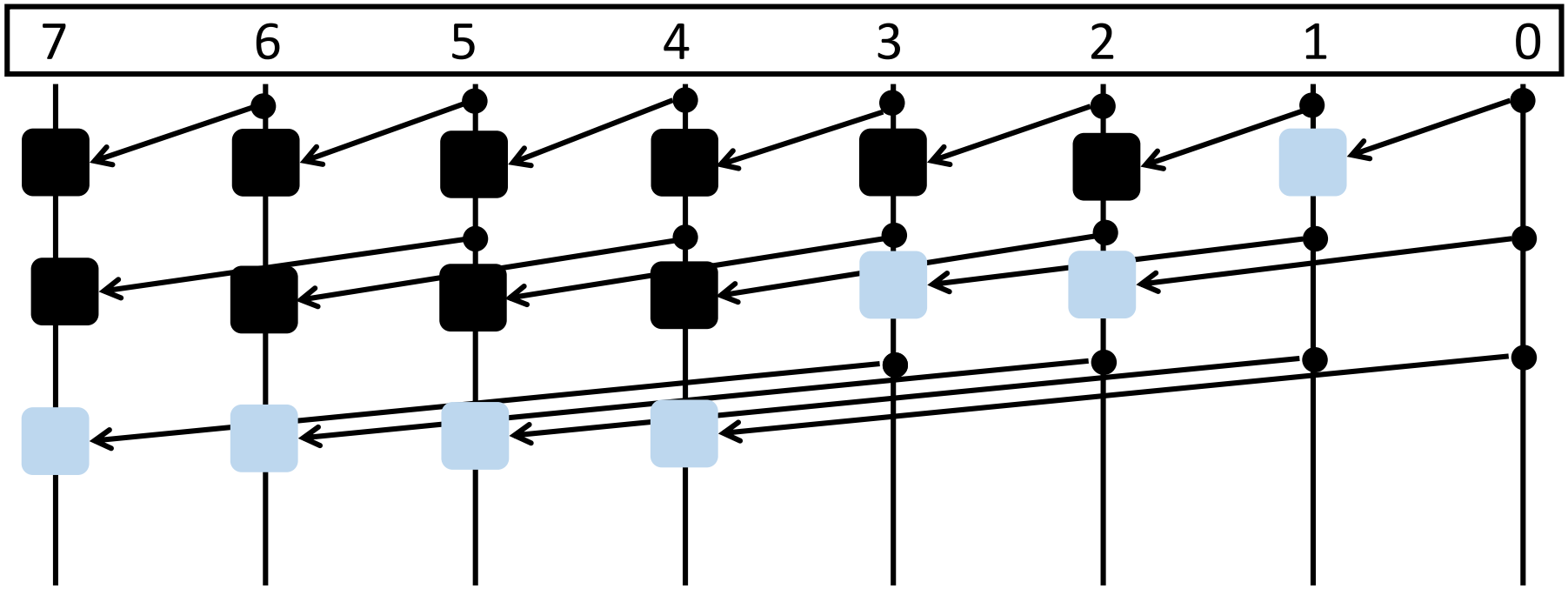
Sklansky (1960)
15 bit Configuration



Any Method to Maintain Logic Level & Fan-out?



Kogge-Stone Adder

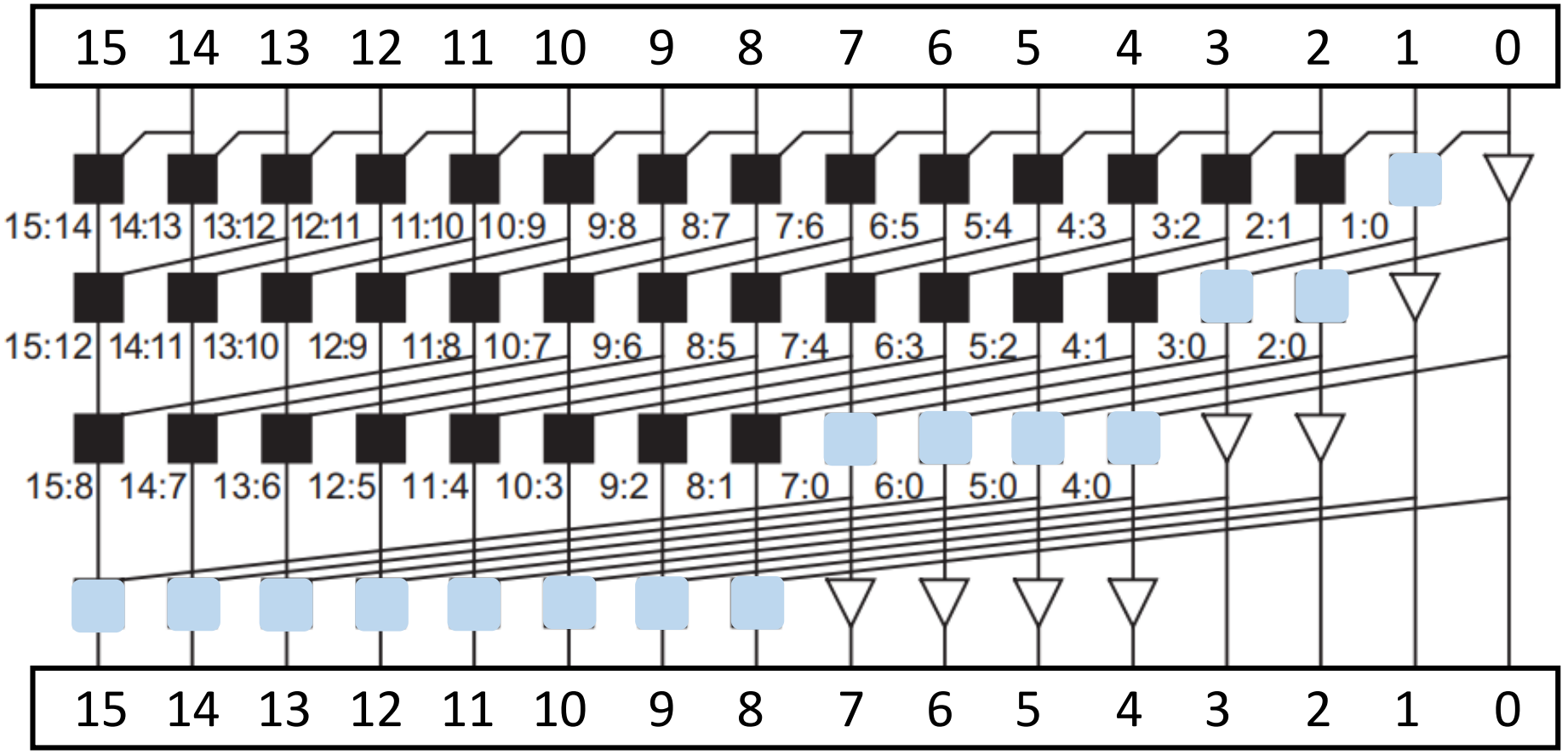


Logic level & FO are simultaneously reduced!

Kogge-Stone Adder (1973)

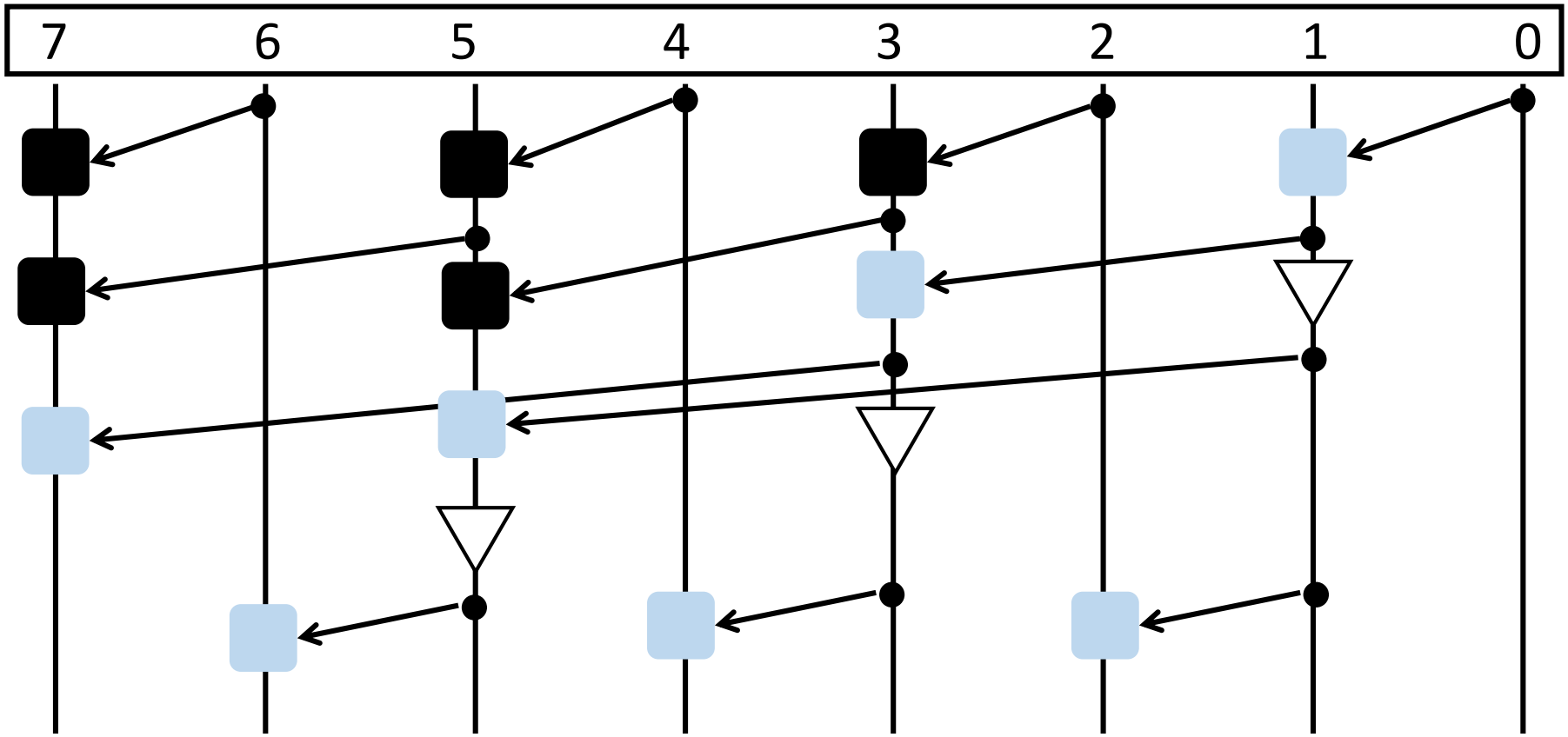
But, wire tracks are very high

15-Bit Kogge-Stone Adder



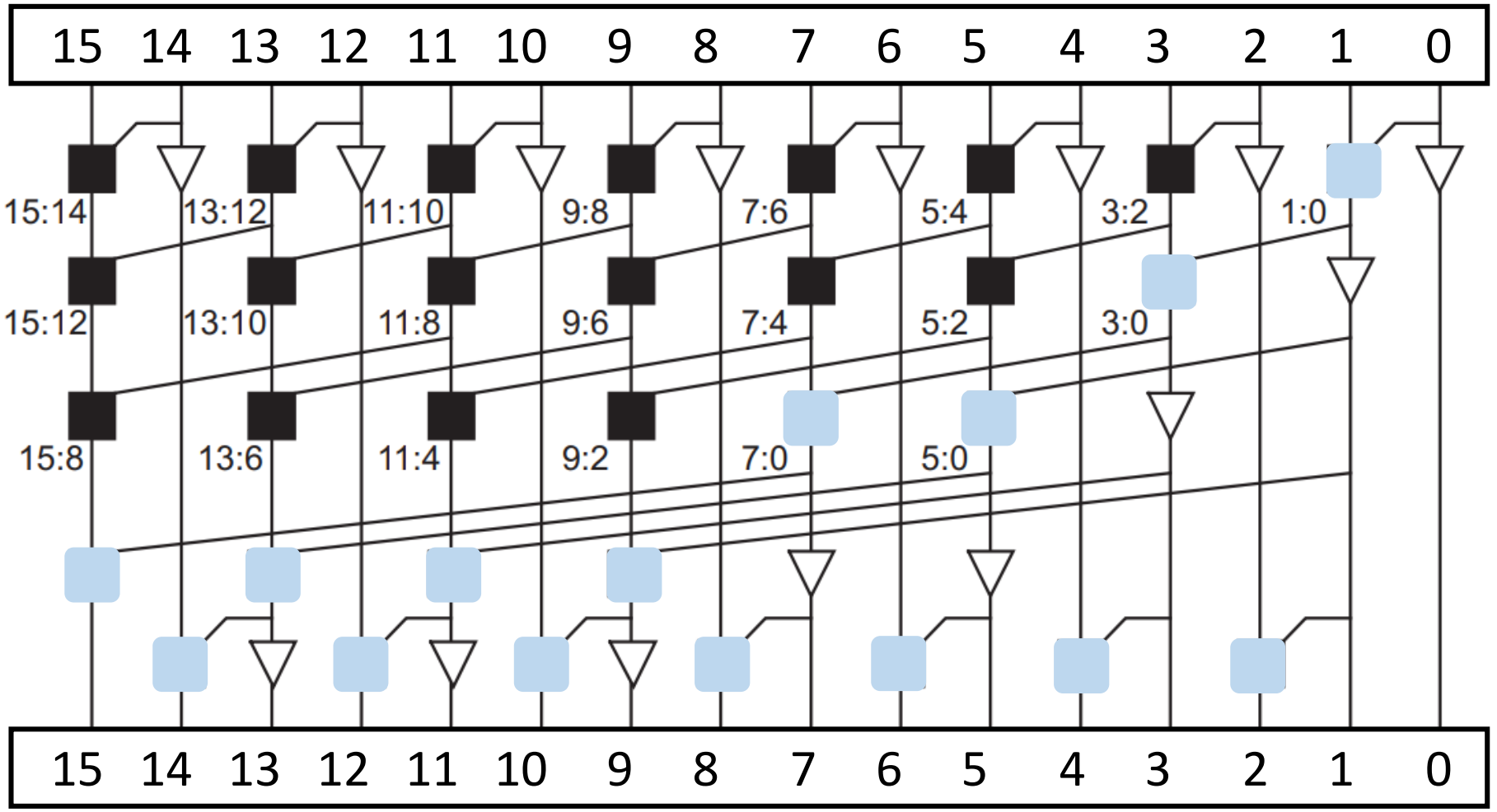
How can we reduce number of squares?

Reducing Number of Nodes

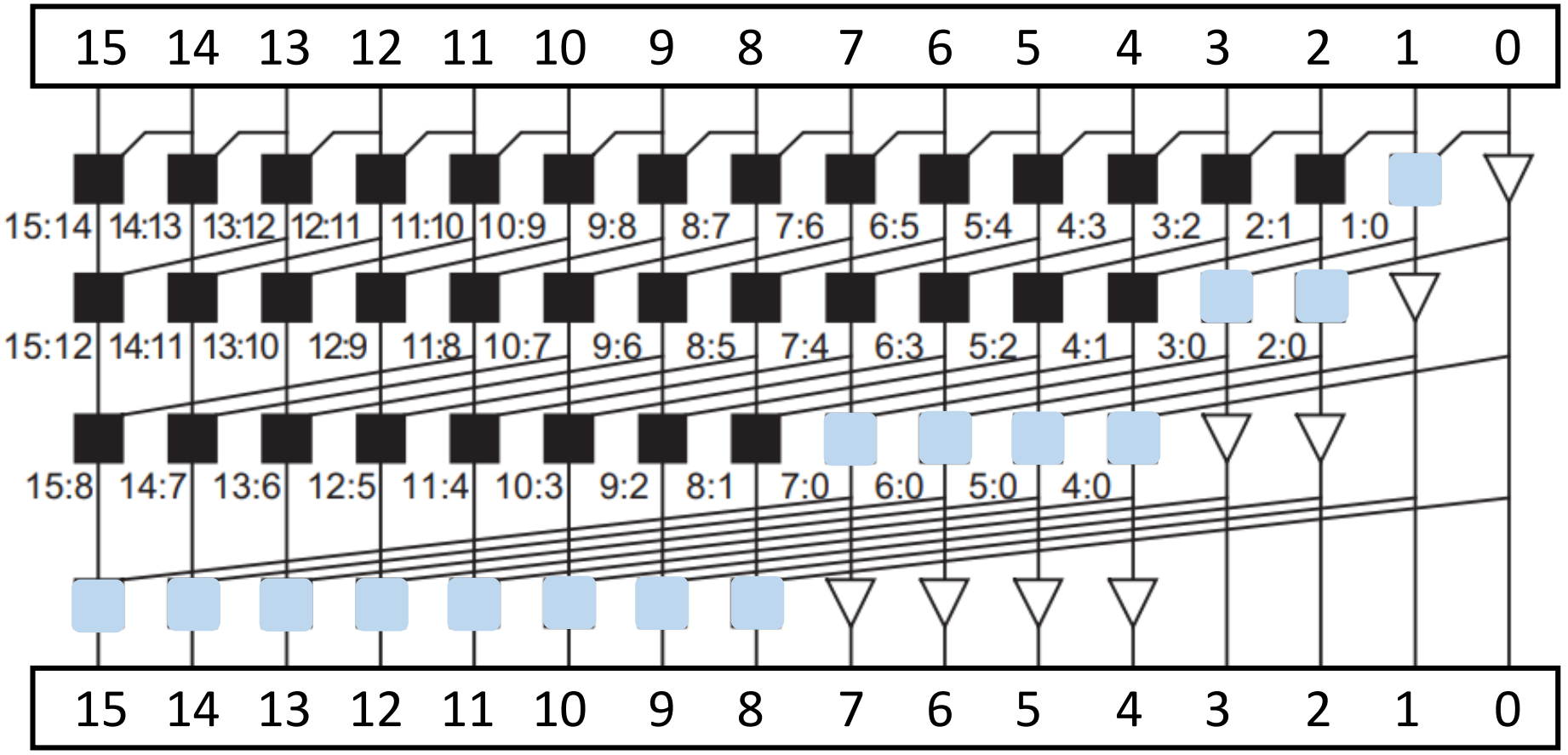


Han-Carlson Adder (1987)

15-Bit Han-Carlson Adder



Revisit 15-Bit Kogge-Stone Adder

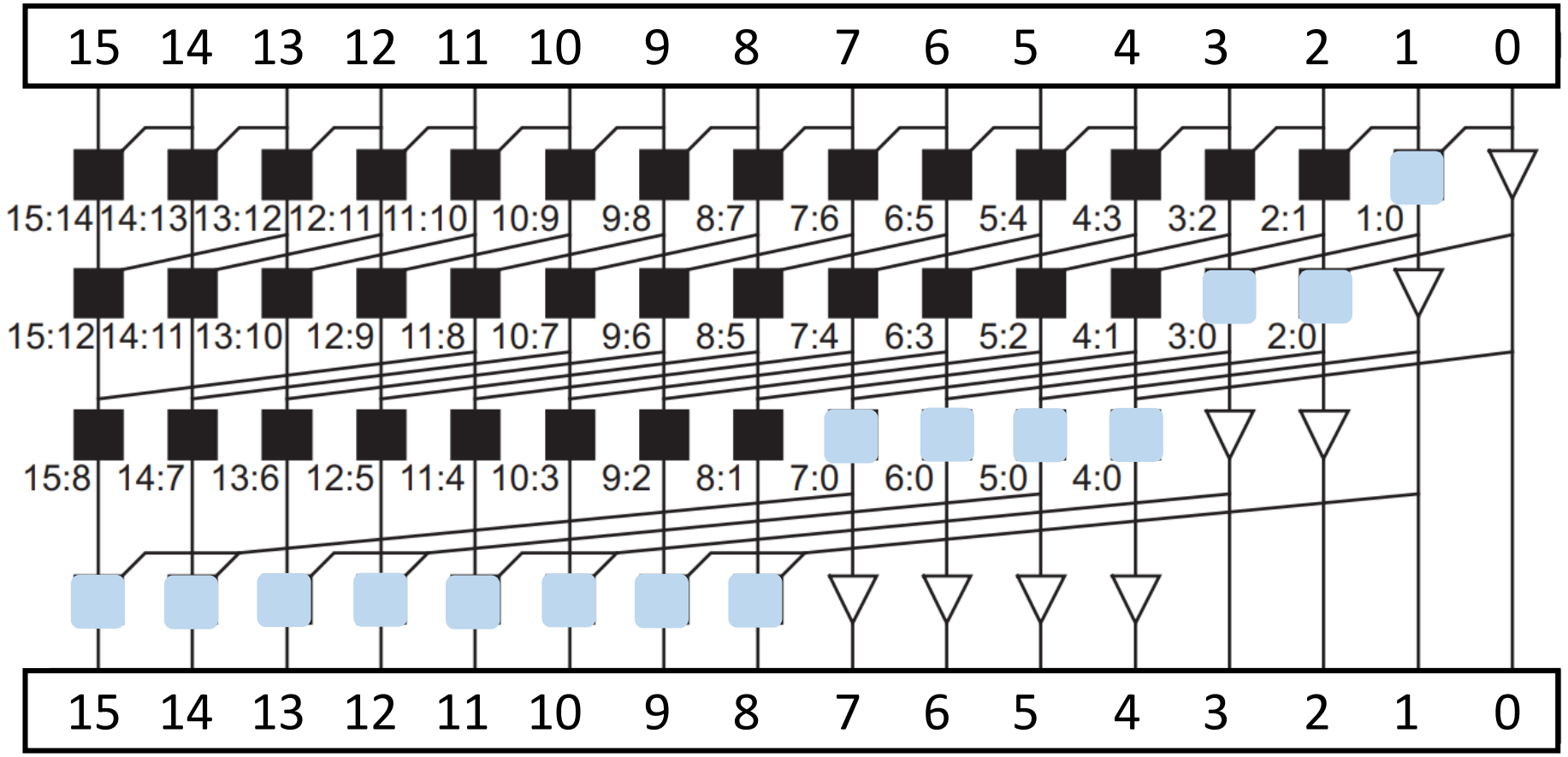


$G_{14:0} = G_{14:7} + P_{14:7}G_{6:0}$

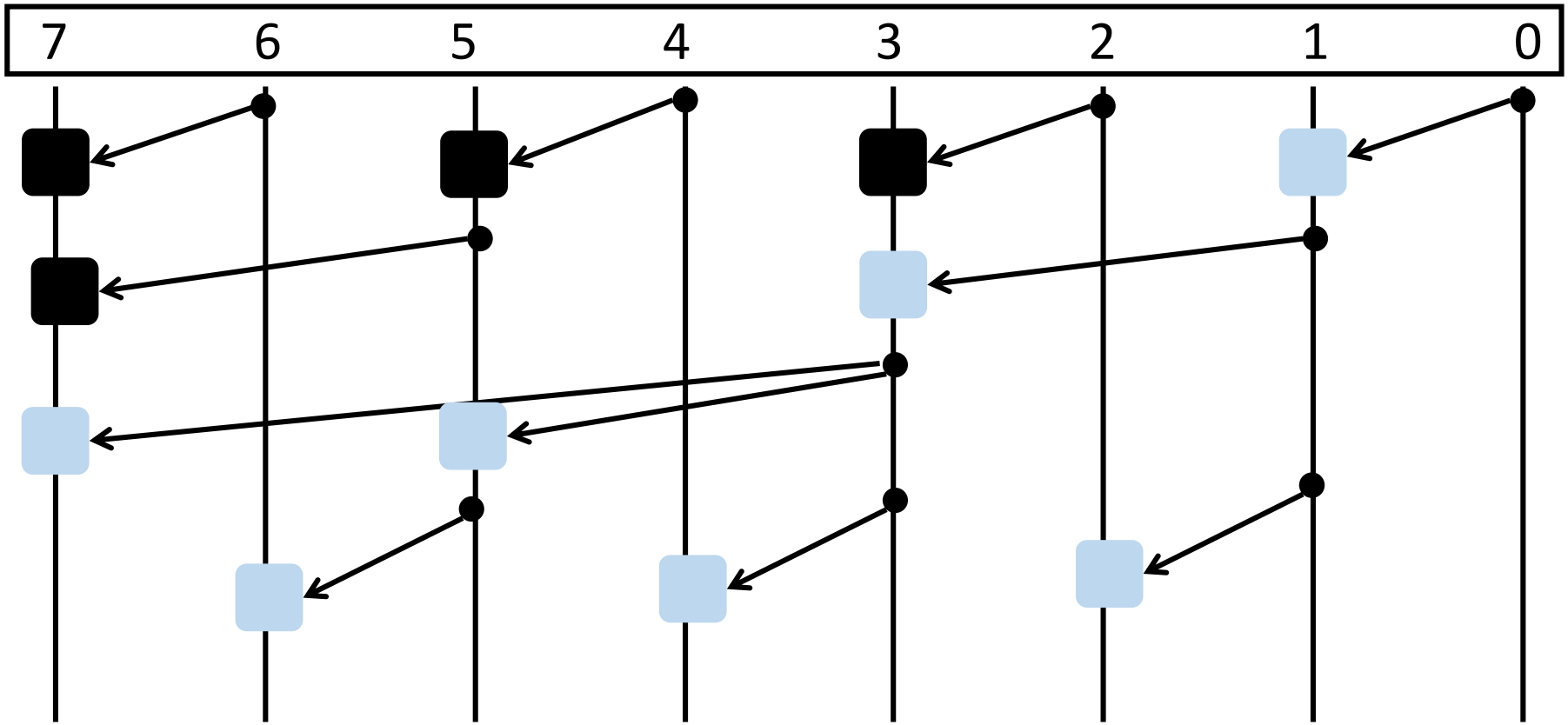
← How about $G_{14:7} + P_{14:7}G_{7:0}$?

← $G_{14:7} + P_{14:7}(G_7+P_7G_{6:0})$

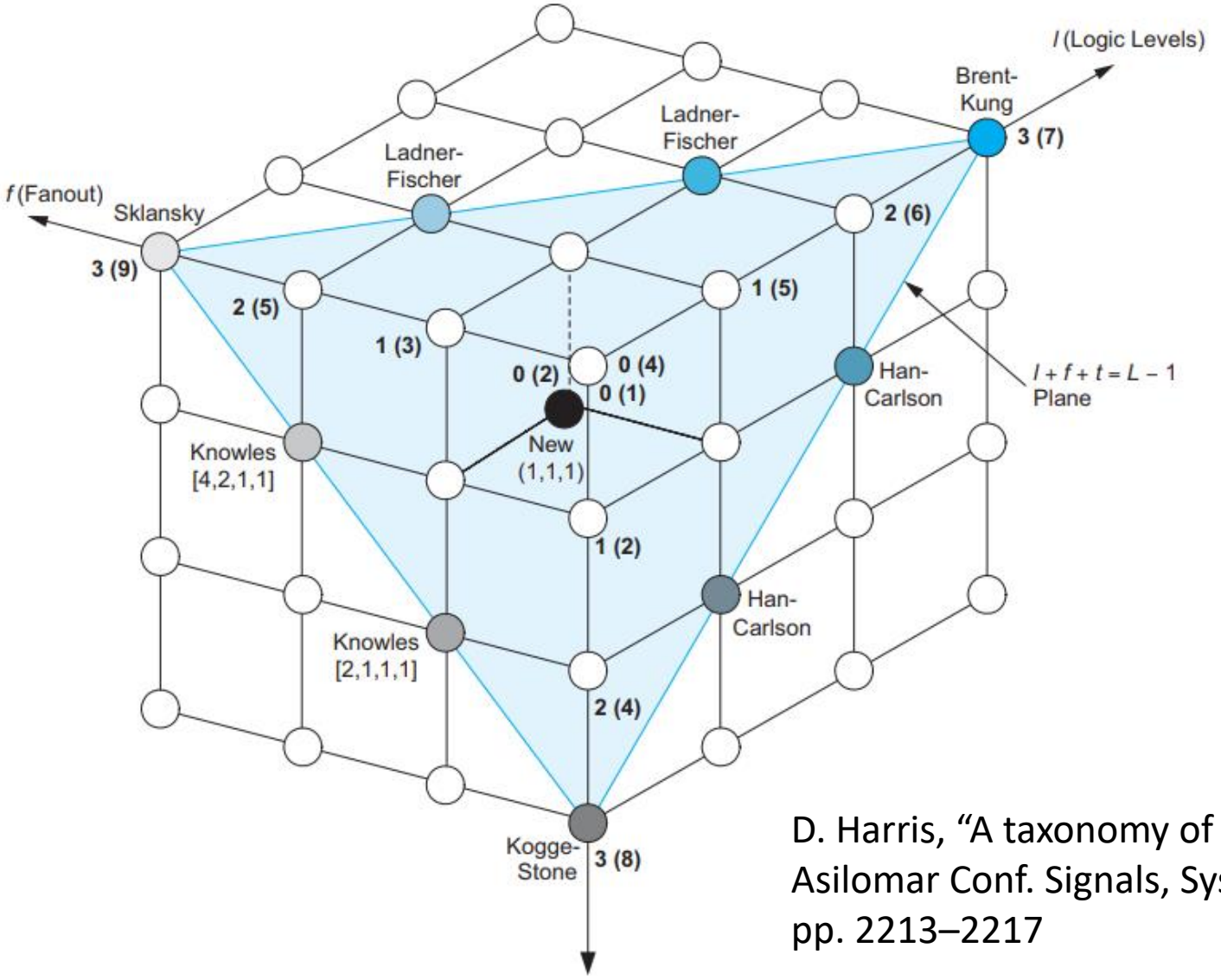
15-Bit Knowles Adder (1999)



Lander-Fisher



Three Respects of Tree Adder



D. Harris, "A taxonomy of prefix networks," Proc. 37th Asilomar Conf. Signals, Systems, and Computers, 2003, pp. 2213–2217

Checkpoints

- ✓Various structures of tree adders
- ✓Three performance respects of tree adders
- ✓Coming up next: Adder other than tree structures