EE671: VLSI Design
Spring 2024/25

Laxmeesha Somappa

Department of Electrical Engineering

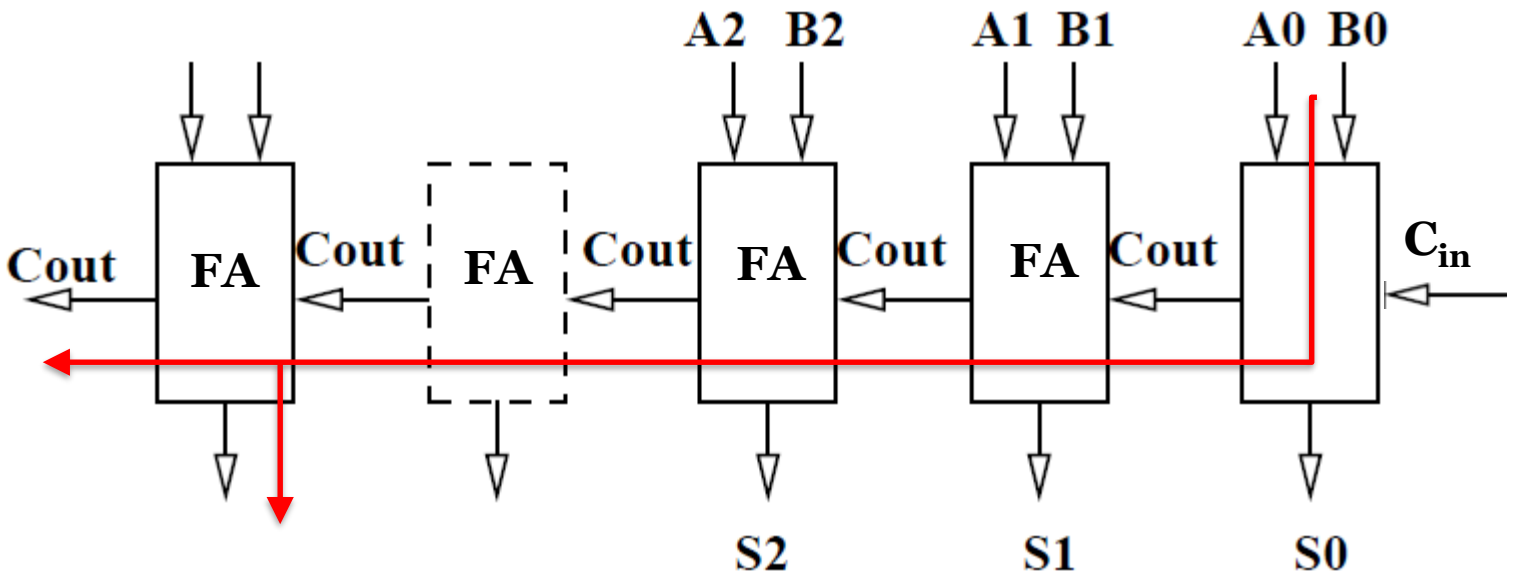
IIT Bombay

laxmeesha@ee.iitb.ac.in

Lecture – 13
Arithmetic IP: Adders

laxmeesha@ee.iitb.ac.in

N-BIT ADDERS: RIPPLE CARRY ADDER (RCA)

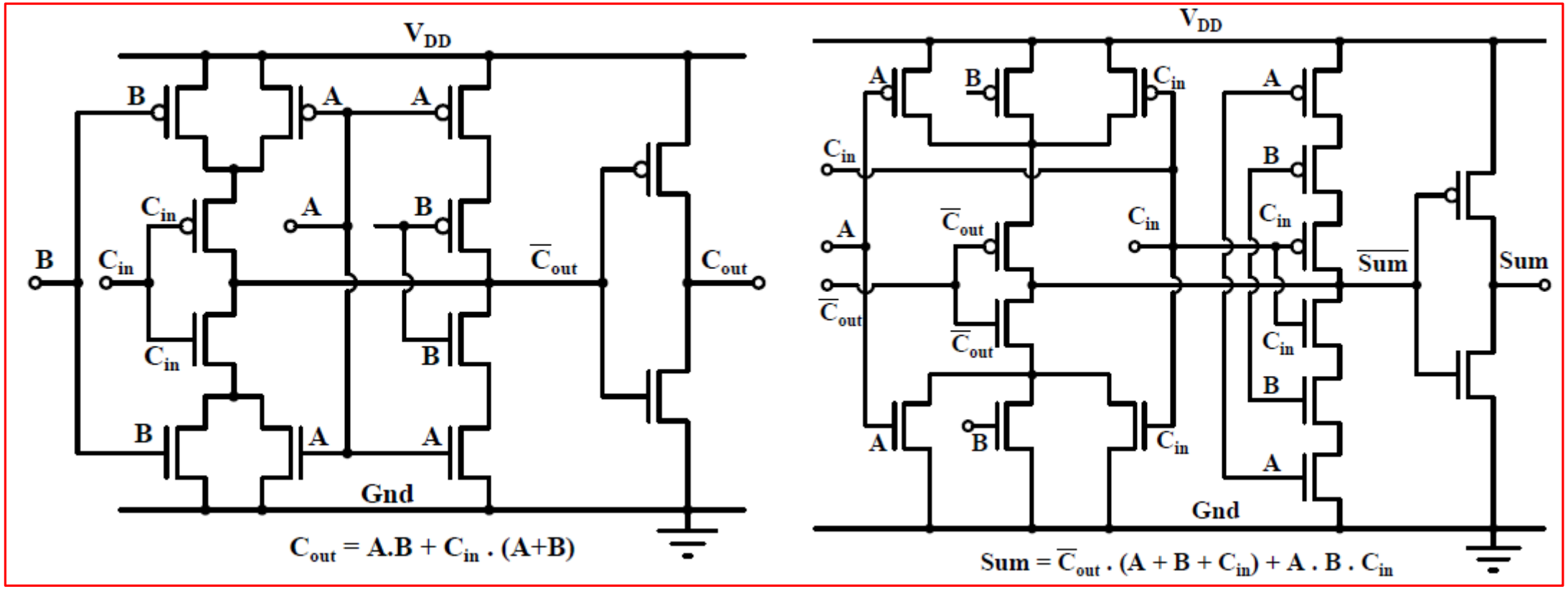


- RCA: Cascade of FAs
- Worst case delay: Carry path (carry propagates from C_{in} to C_{out})
- $T_{adder\_wc} = (N-1)\ t_{carry} + t_{sum}$
- Speeding up the adder: reduce carry hardware delay
- Alternately: Sum is not in the critical path !!

RCA: Optimization

- Sum is not in the critical path
  - Can generate sum slowly → if it can reduce MOSFETs

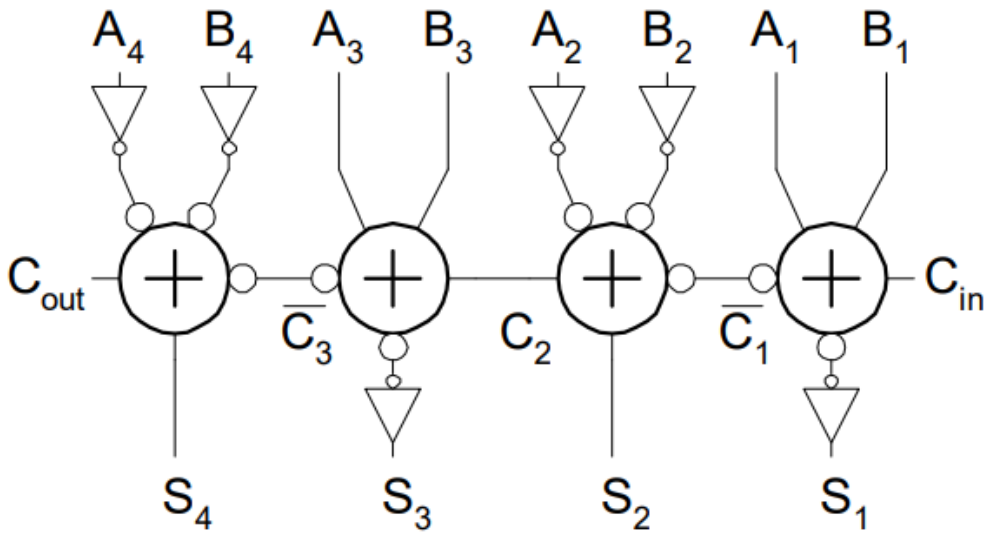
$$sum = \overline{C_{out}} \cdot (A + B + C_{in}) + A \cdot B \cdot C_{in}$$

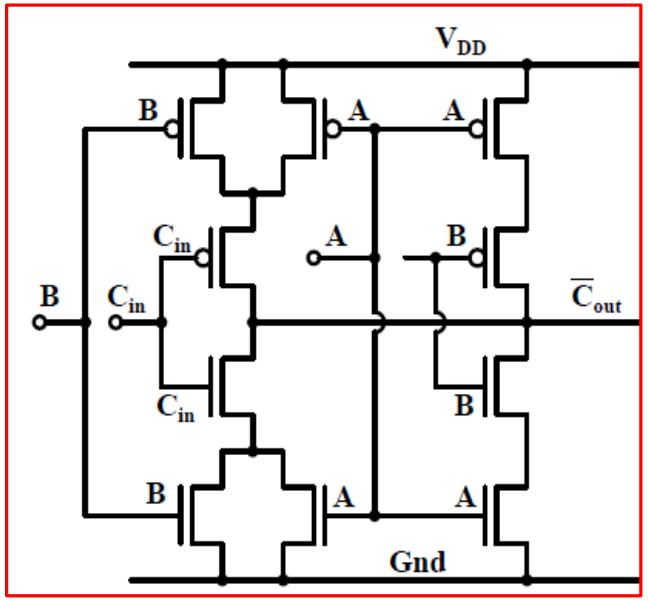


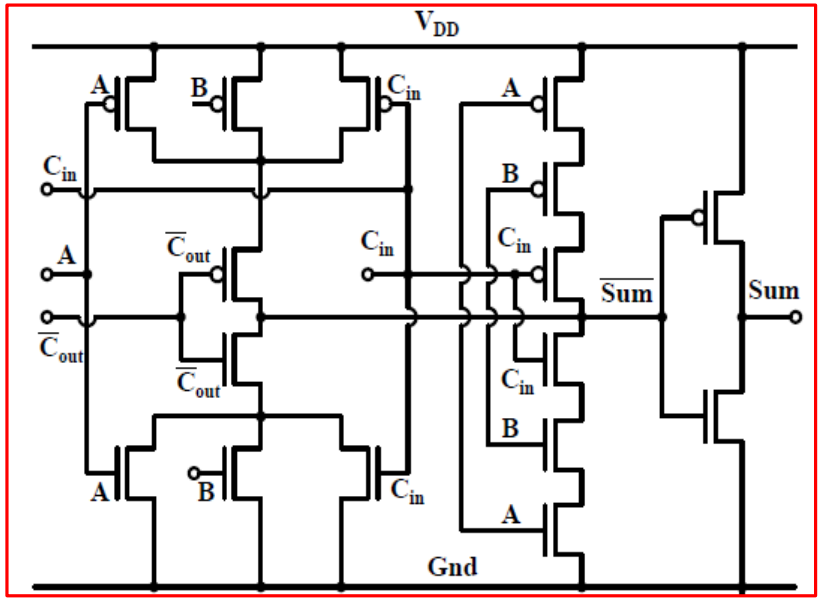
A total of 28 MOSFETs to generate Sum and Carry (Property-1)

RCA: Further Optimization?

- Use property-2 !! → Odd FAs: complement input







- Using only property-1:
  - [28*4] = 112 MOSFETs, delay: ($N.t_{carry}$)
- Now:
  - [10+16]*2 + [10+14+4]*2 = 108 MOSFETs, delay: ($N.t_{carry\_new}$)
  - t_{carry_new}: one-inverter delay lesser than t_{carry} → i.e., ($N.\ t_{inv}$) faster

Few Observations before we proceed

A	B	C	C_{out}	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

When both inputs are 0, irrespective of C_{in}, C_{out} is always 0.
Kill the Cout. i.e., when,

$$K \equiv \overline{A} \cdot \overline{B} \ , \ C_{out} = 0$$

Other input cases (only one input is 0/1), $C_{out} = C_{in}$.
Propagate Cout. i.e., when,

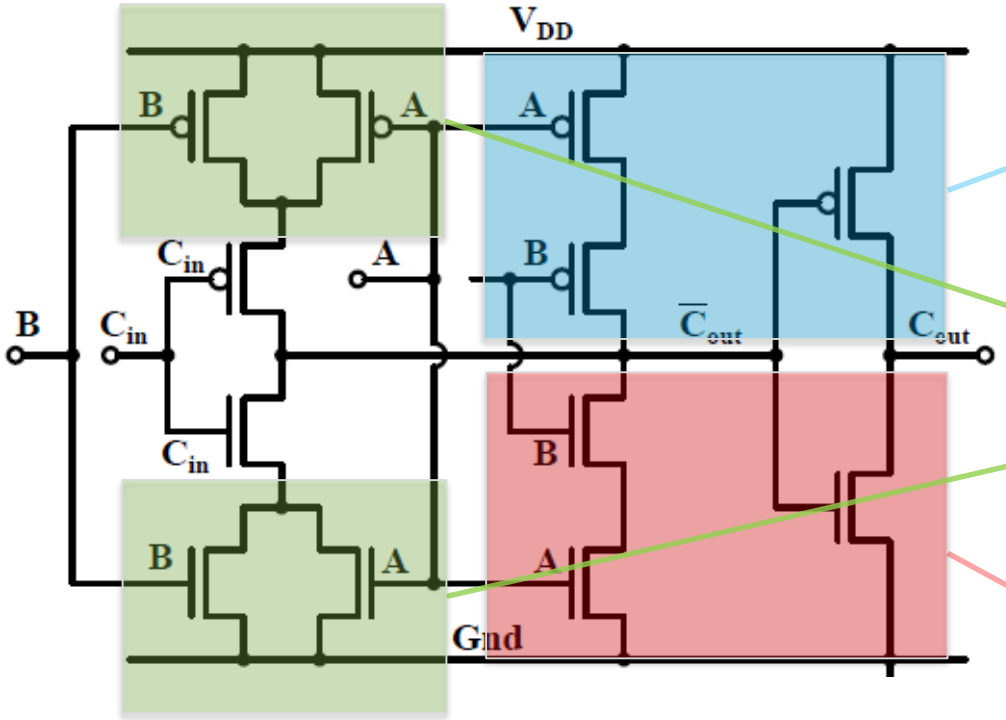
$$P \equiv A \oplus B \ , \ C_{out} = C_{in}$$

When both inputs are 1, irrespective of Cin, Cout is always 1.
Generate Cout. i.e., when,

$$G \equiv A.B \ , \ C_{out} = 1$$

$$\text{Sum} = P \oplus C_{in} \text{ and } C_{out} = G + P.C_{in}$$

GENERATING P AND G



When both inputs are 0, irrespective of C_{in}, C_{out} is always 0.
Kill the C_{out}. i.e., when,

$$K \equiv \overline{A} \cdot \overline{B},\ C_{out} = 0$$

Other input cases (only one input is 0/1), $C_{out} = C_{in}$.
Propagate C_{out}. i.e., when,

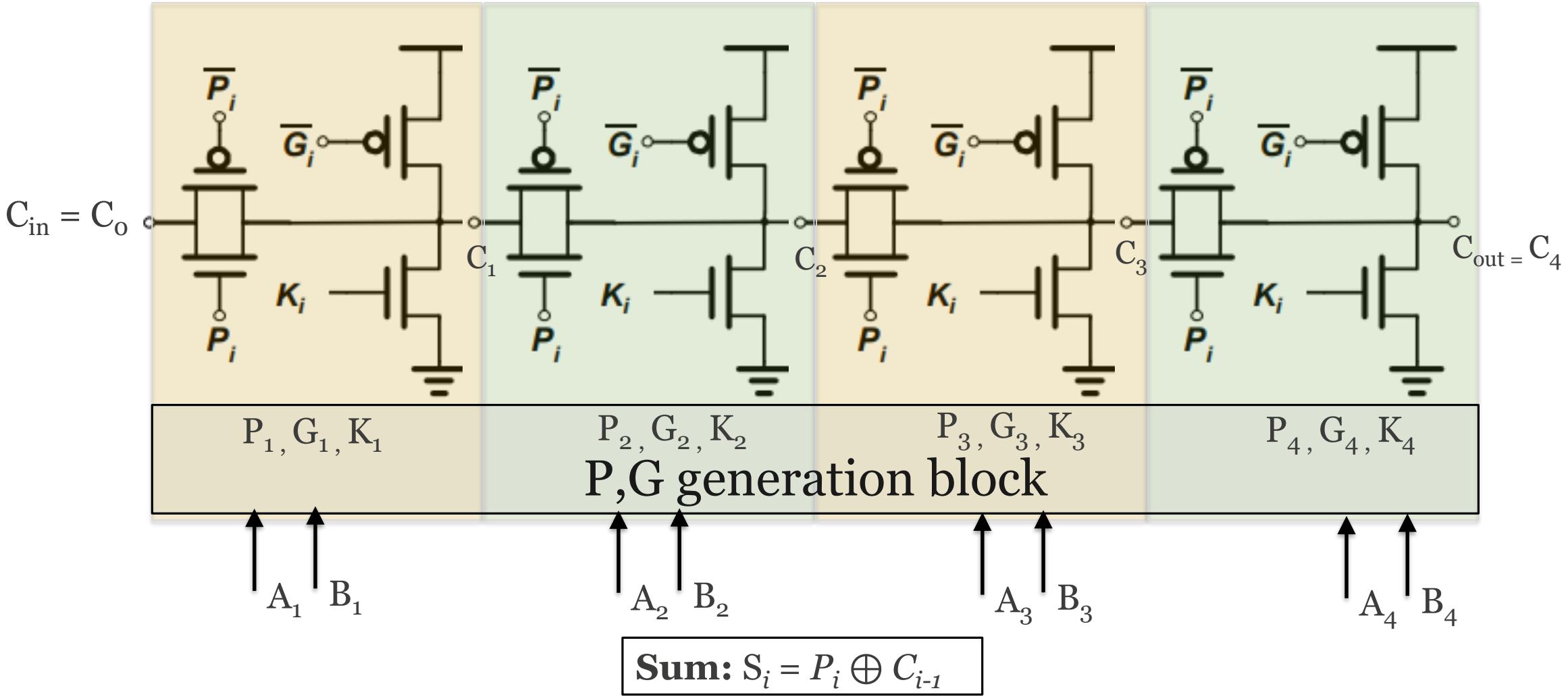
$$P \equiv A \oplus B,\ C_{out} = C_{in}$$

When both inputs are 1, irrespective of C_{in}, C_{out} is always 1.
Generate C_{out}. i.e., when,

$$G \equiv A.B,\ C_{out} = 1$$

laxmeesha@ee.iitb.ac.in

4-BIT MANCHESTER CARRY CHAIN



Sum: $S_i = P_i \oplus C_{i-1}$

- Worst case delay: All "P" are 1
- Series of RC network → delay high in the absence of static inverter

THE CARRY RECURRENCE

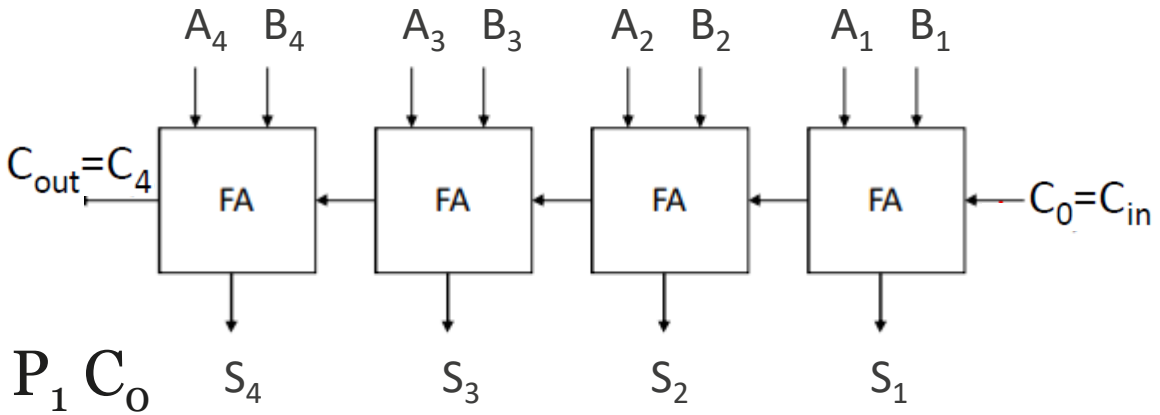
Recurrence: $C_i = G_i + P_i C_i$

$$C_1 = G_1 + P_1 C_0$$

$$C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

$$C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$$

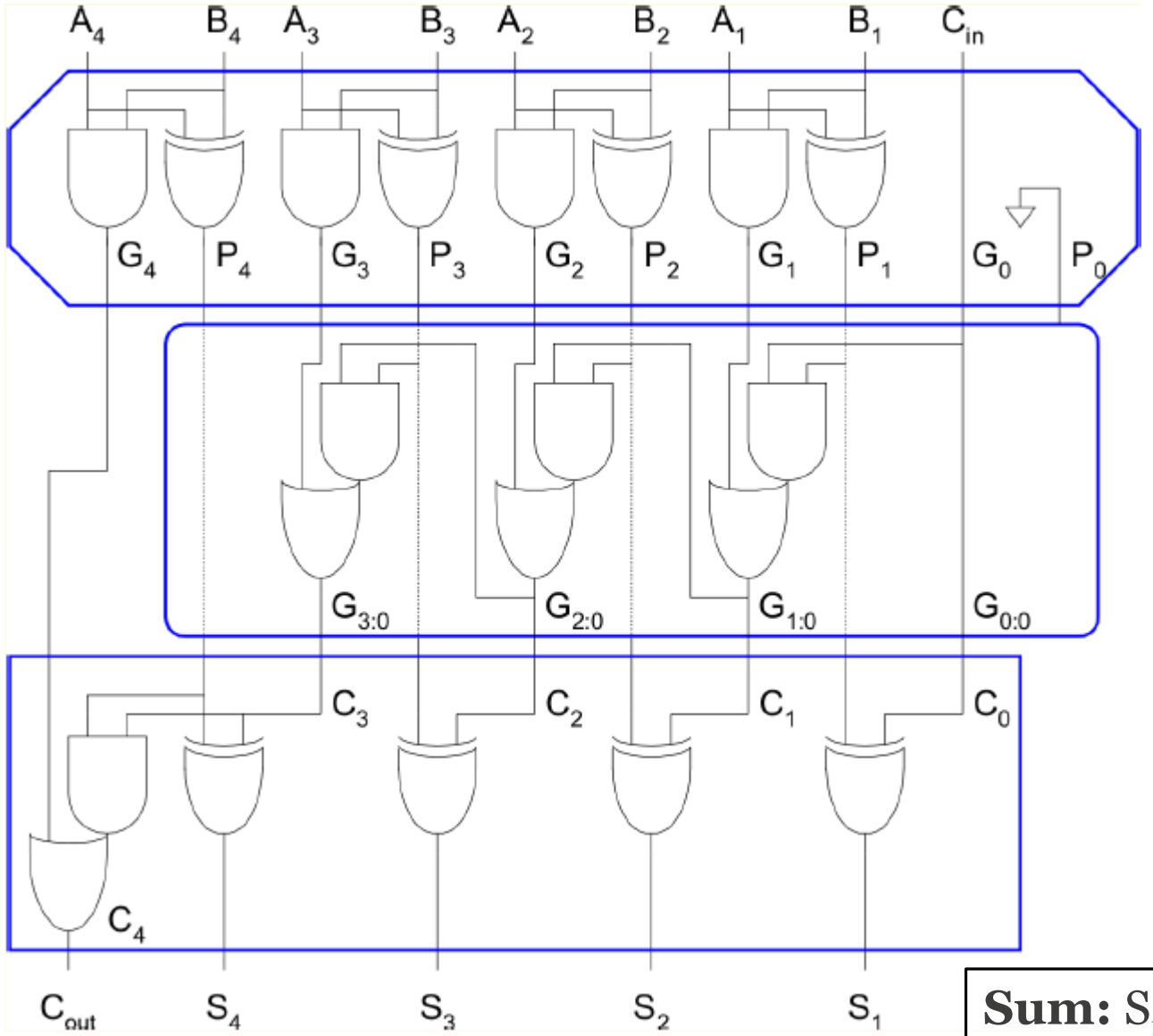


- Consider a 4 bit adder.
- The final C_{out} (C_4 in this case) can be directly generated from C_0 (neglecting the hardware complexity)
- Practically: to generate C_4 we need high fan-in CMOS gates (max 5 input AND)
- Generating C4 will have the same order of delay as RCA with FA !!!
- Remember: fan-in increases with increased adder size → limitation → might be okay for 4-bits

RCA USING PG

$$C_1 = G_1 + P_1 C_0$$
$$C_2 = G_2 + P_2 C_1$$
$$C_3 = G_3 + P_3 C_2$$

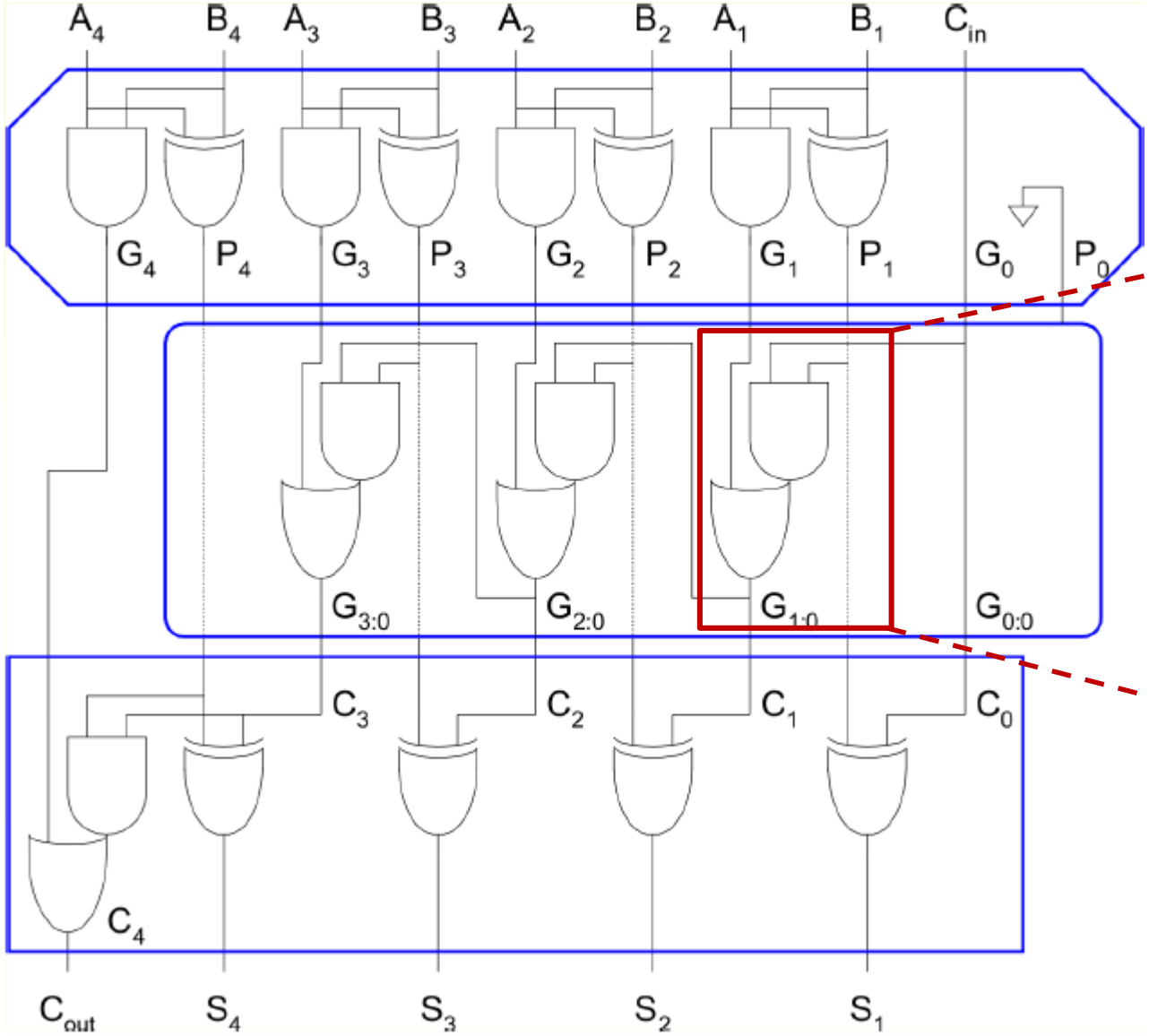
$$C_4 = G_4 + P_4 C_3$$

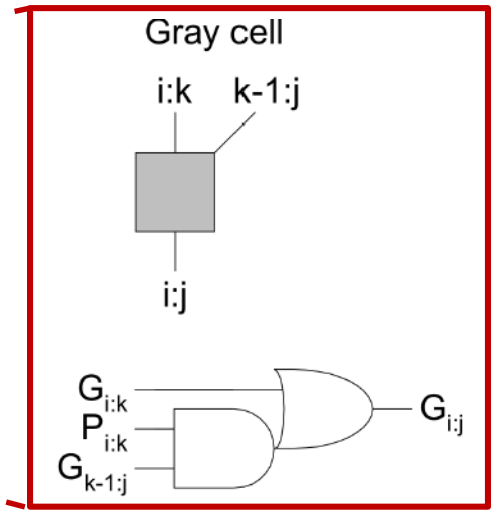


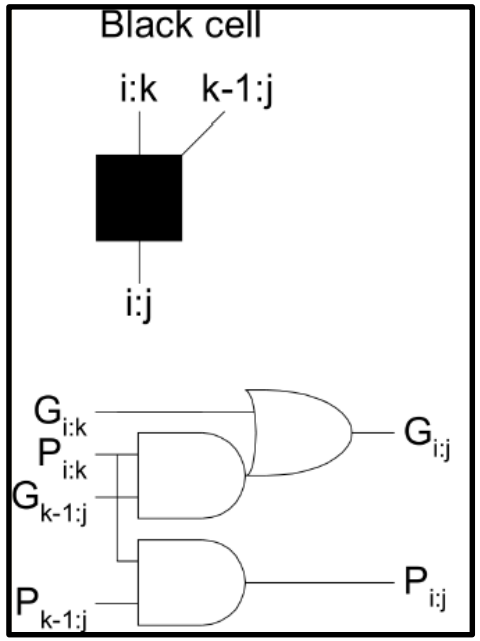
Sum: $S_i = P_i \oplus C_{i-1}$

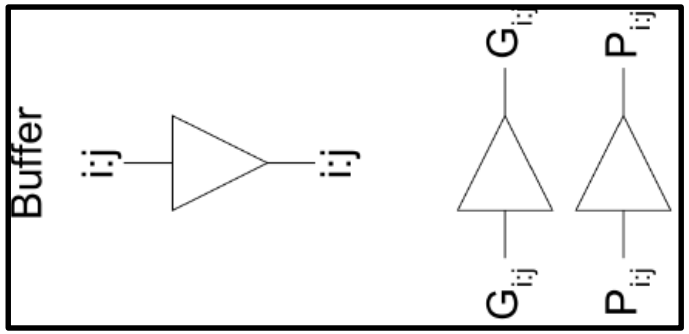
laxmeesha@ee.iitb.ac.in

PG DIAGRAM NOTATION

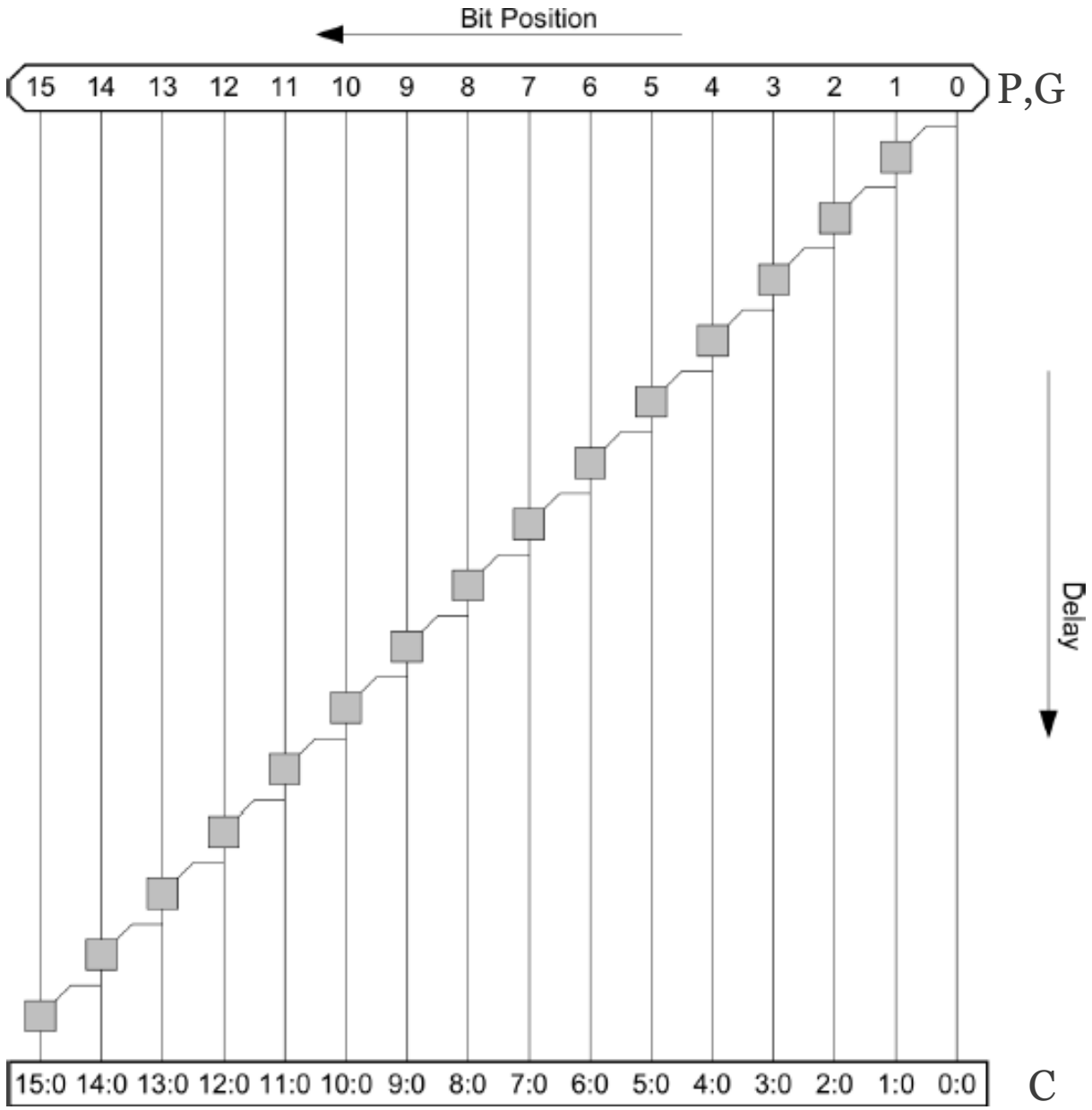


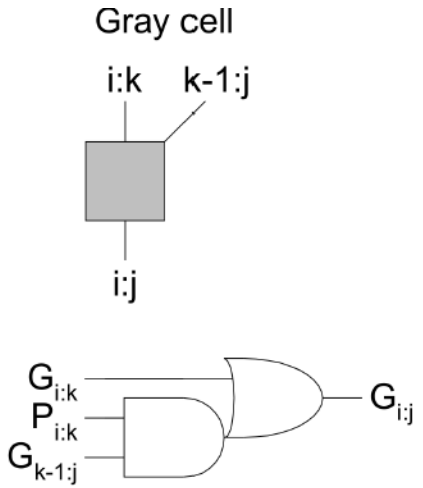






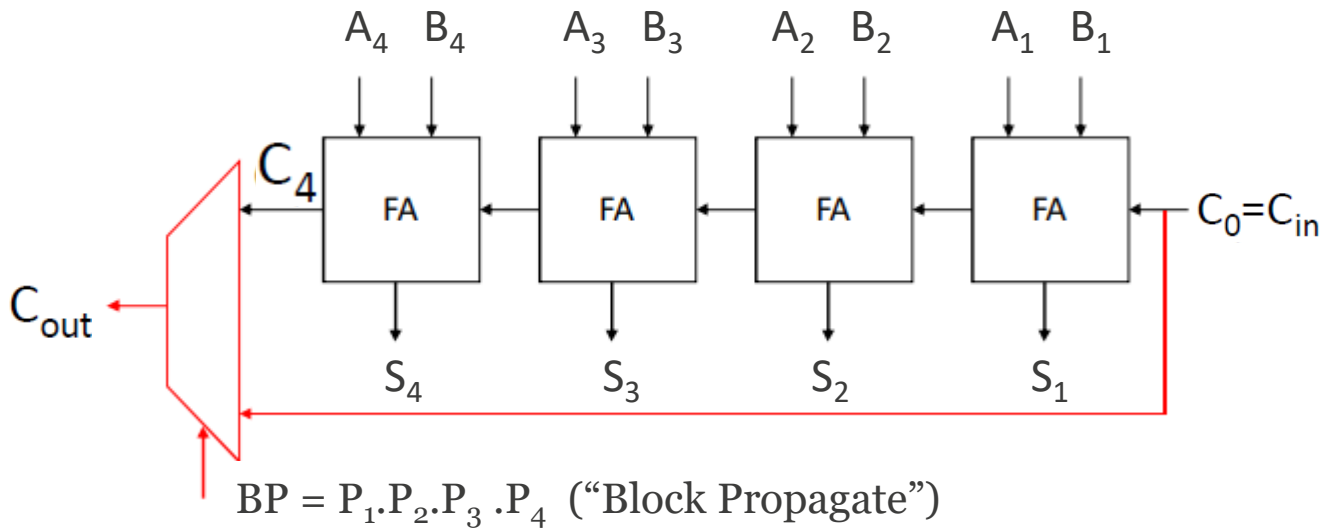
PG DIAGRAM NOTATION (16-BIT ADDER)





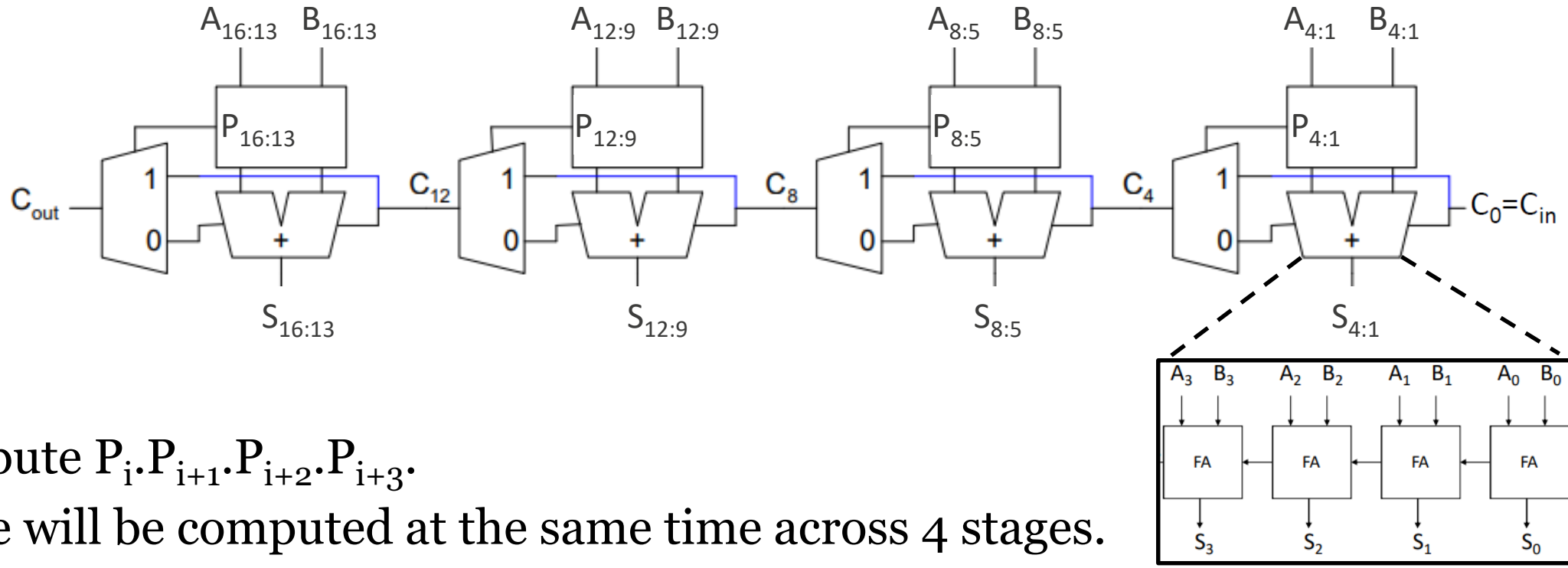
- $t_{add\_wc} = t_{pg} + (N-1)\, t_{AO} + t_{sum}$

CARRY SKIP (BYPASS) ADDER USING FA-RCA



- Compute P and G for each bit parallelly
- If $P_1.P_2.P_3.P_4 = 1$, then carry is propagated (critical path delay)
- If a G is generated in any FA: no propagation → not critical path
- Hence, bypass the critical path through mux !
- No use in its standalone mode. But consider the next case

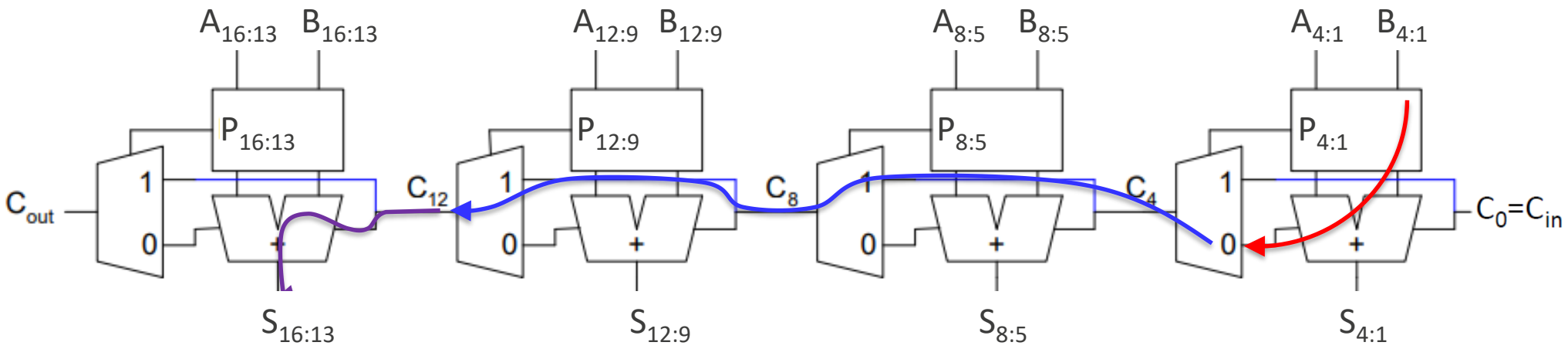
CARRY SKIP (BYPASS) ADDER USING FA-RCA



FA-RCA

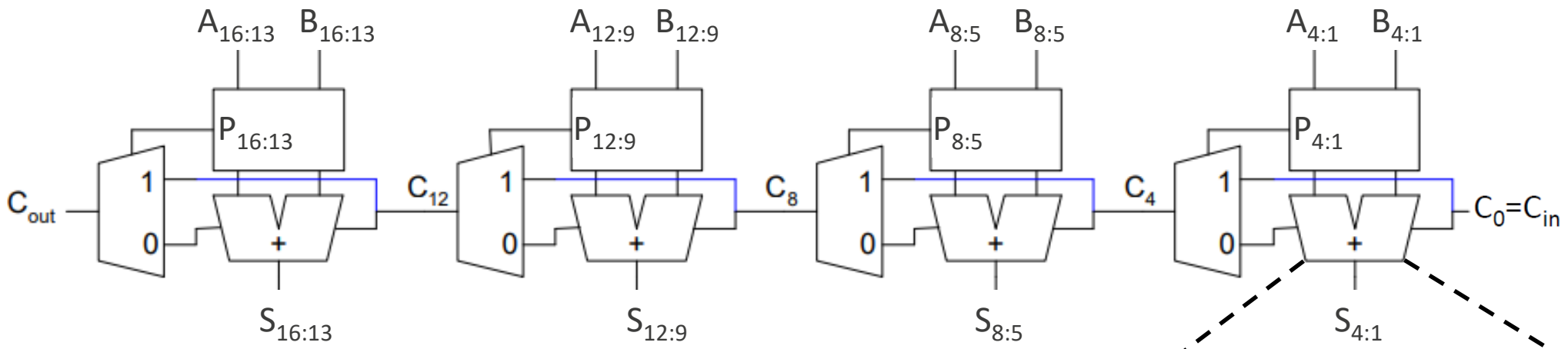
- Compute $P_i.P_{i+1}.P_{i+2}.P_{i+3}$.
- These will be computed at the same time across 4 stages.
- Propagate signal are pre-computed for stage-2 onwards
- Critical path can be bypassed from stage-2 onwards

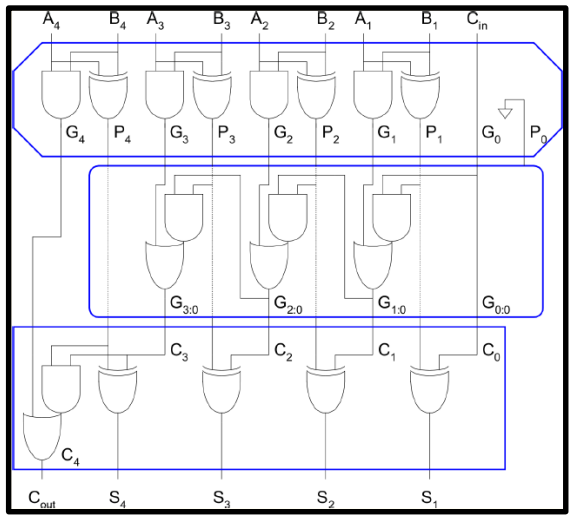
CARRY SKIP (BYPASS) ADDER USING FA-RCA



- Consider N bit adder with B bits in each stage
- $t_{add\_wc} = (B\text{-}1).t_{carry} + (N/B - 1).t_{mux} + (B\text{-}1).t_{carry} + t_{sum}$ [Assuming $t_{pg} < (B\text{-}1)t_{carry}$]
- Recall, $t_{add,rca\_wc} = (N\text{-}1)\ t_{carry} + t_{sum}$
- For carry skip adder: optimal B = sqrt(N/2) [Solve this !!]

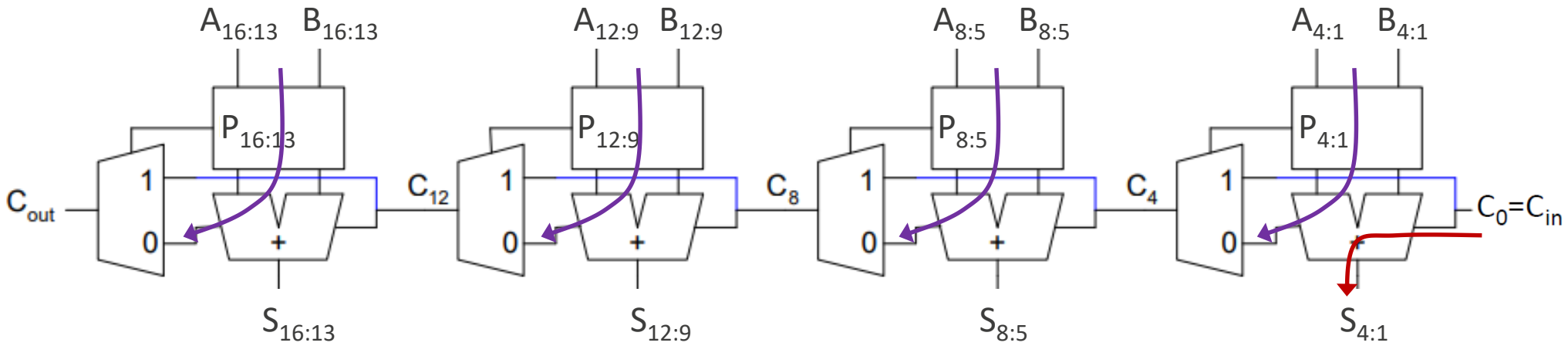
CARRY SKIP (BYPASS) ADDER USING PG-RCA

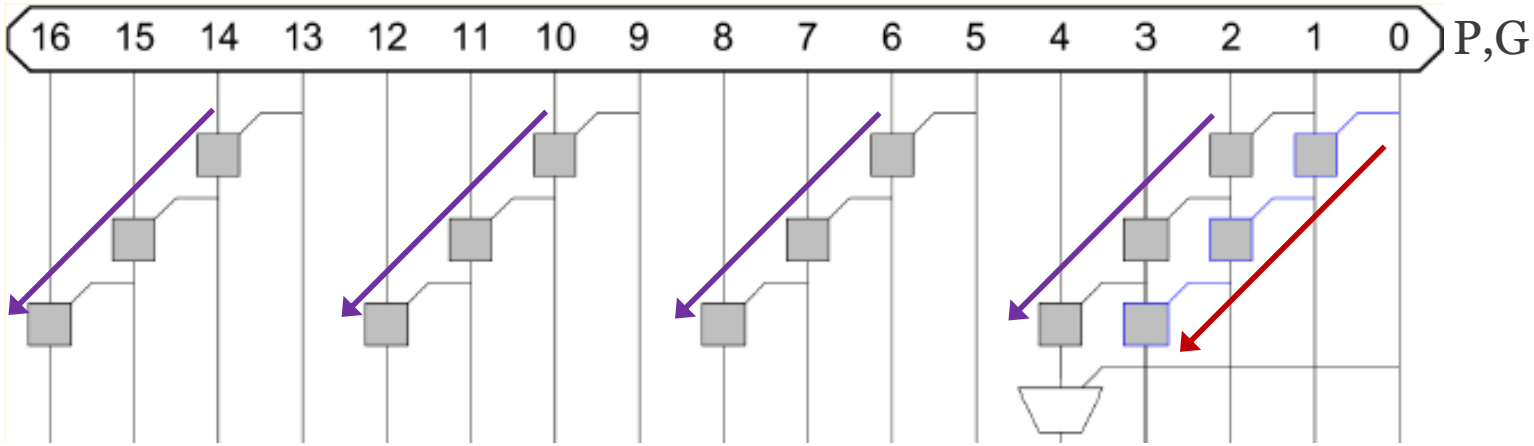




- Compute $P_i.P_{i+1}.P_{i+2}.P_{i+3}$.
- These will be computed at the same time across 4 stages.
- Propagate signal are pre-computed for stage-2 onwards
- Critical path can be bypassed from stage-2 onwards

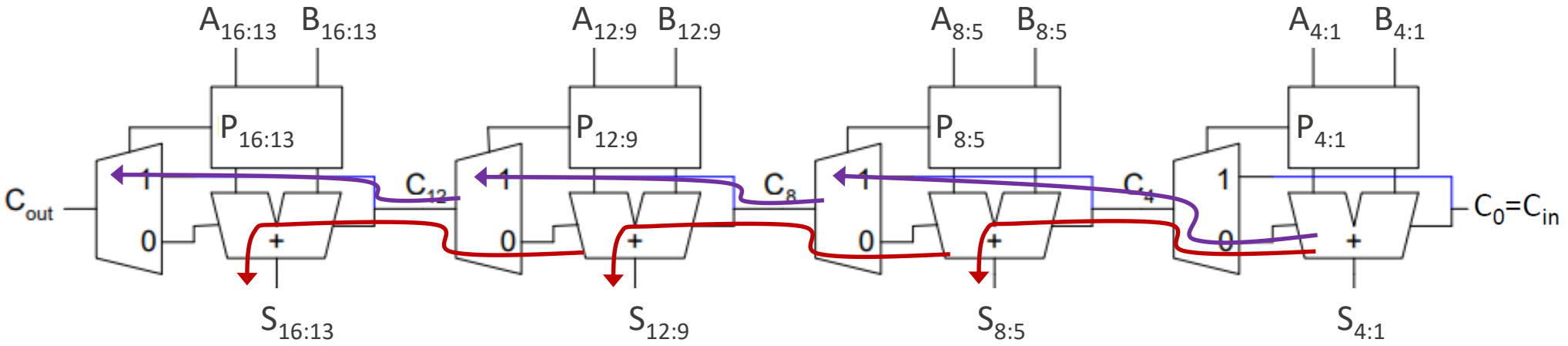
CARRY SKIP (BYPASS) ADDER USING PG-RCA

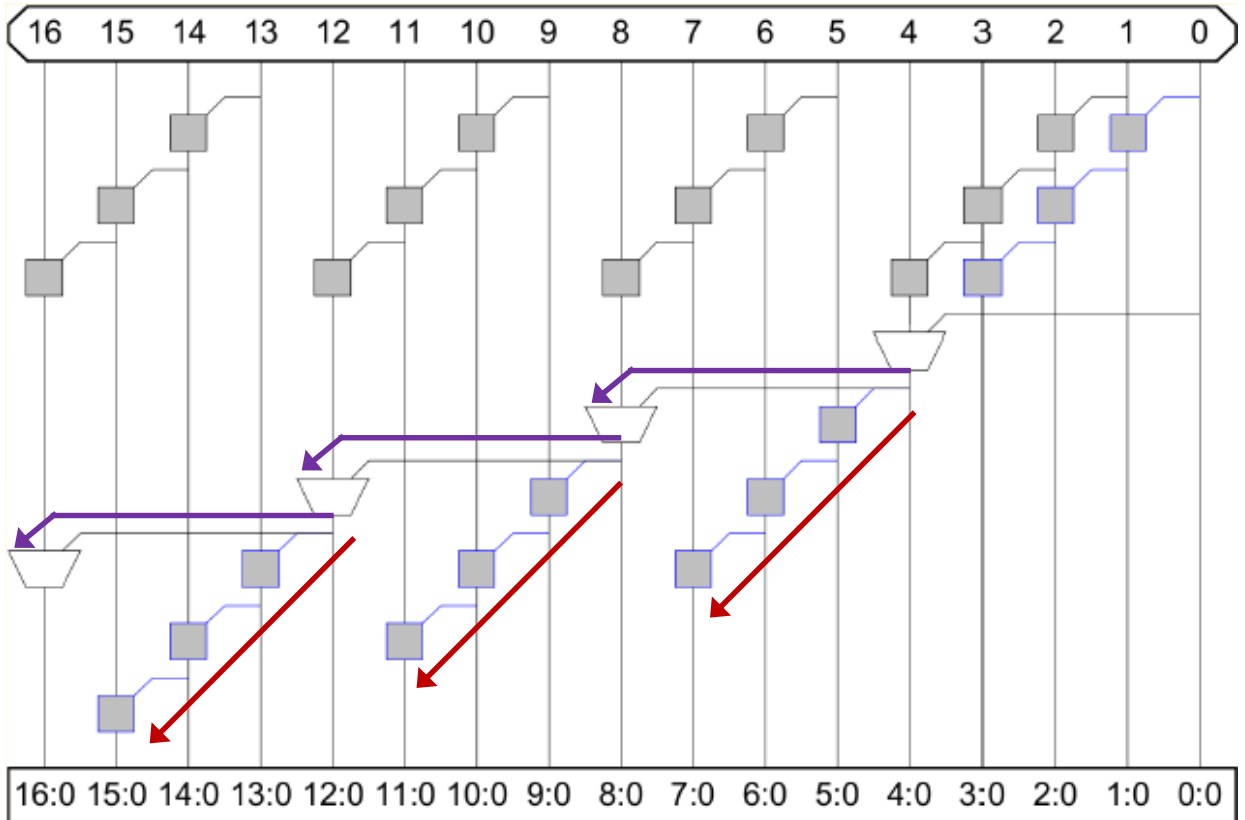




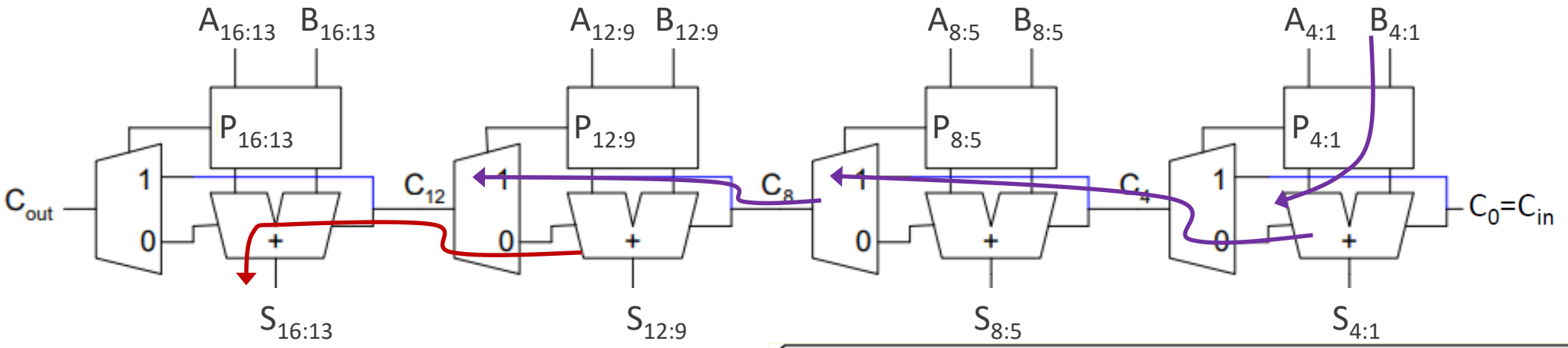
- First block of 4-bits → worst-case delay → no propagate (no bypass) → ripple through 3 bits

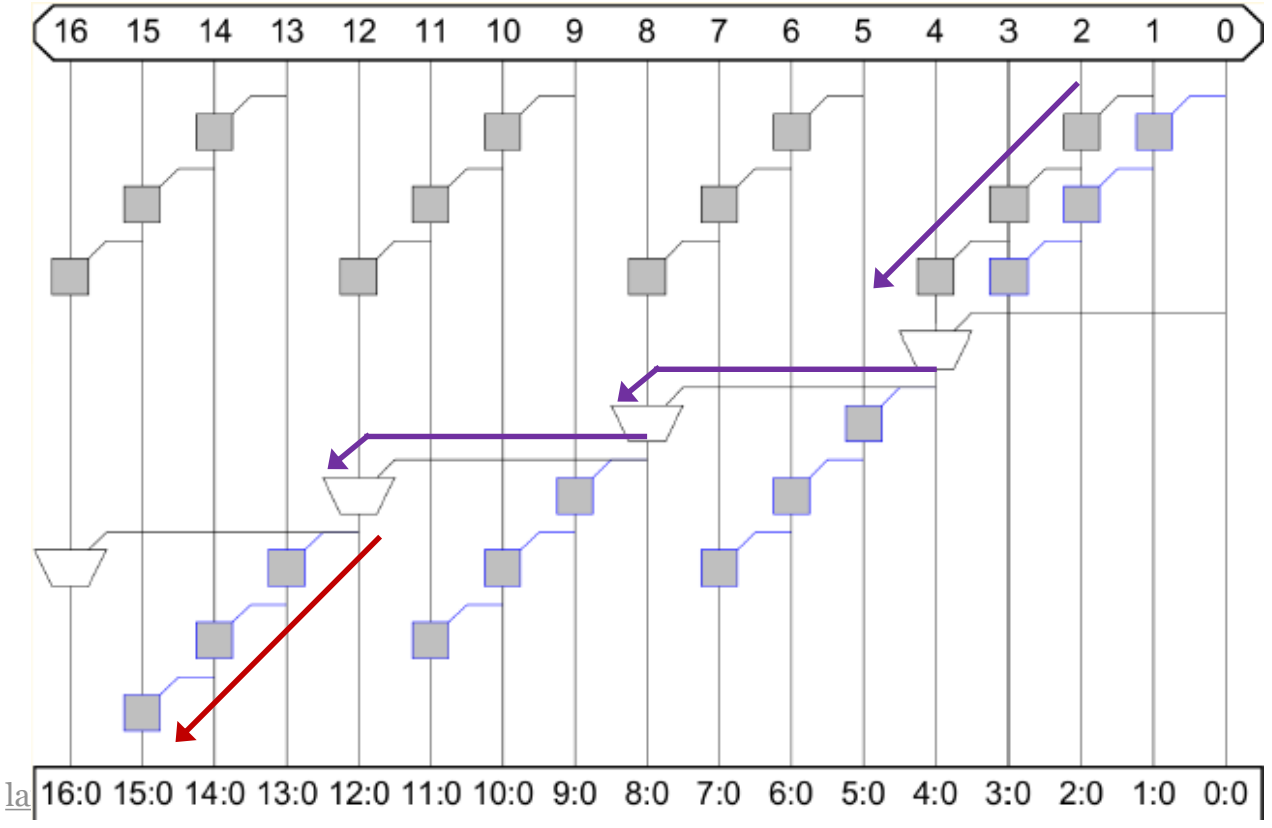
CARRY SKIP (BYPASS) ADDER USING PG-RCA





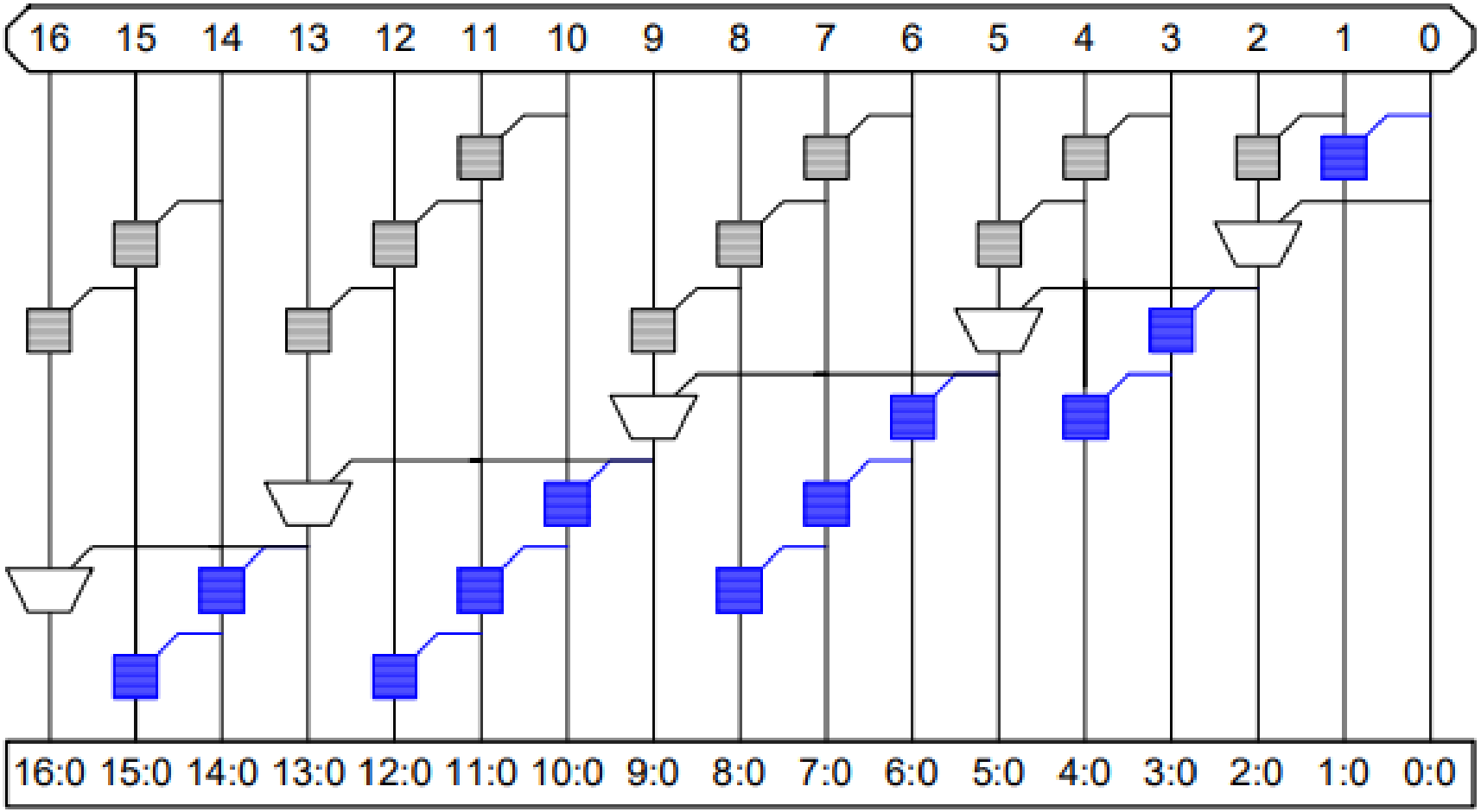
CARRY SKIP (BYPASS) ADDER USING PG-RCA





- $T_{add\_wc} = t_{pg} + (B\text{-}1).t_{AO} + (N/B - 1).t_{mux} + (B\text{-}1).t_{AO} + t_{sum}$
- For carry skip adder: optimal B = sqrt(N/2) [Solve this !!]

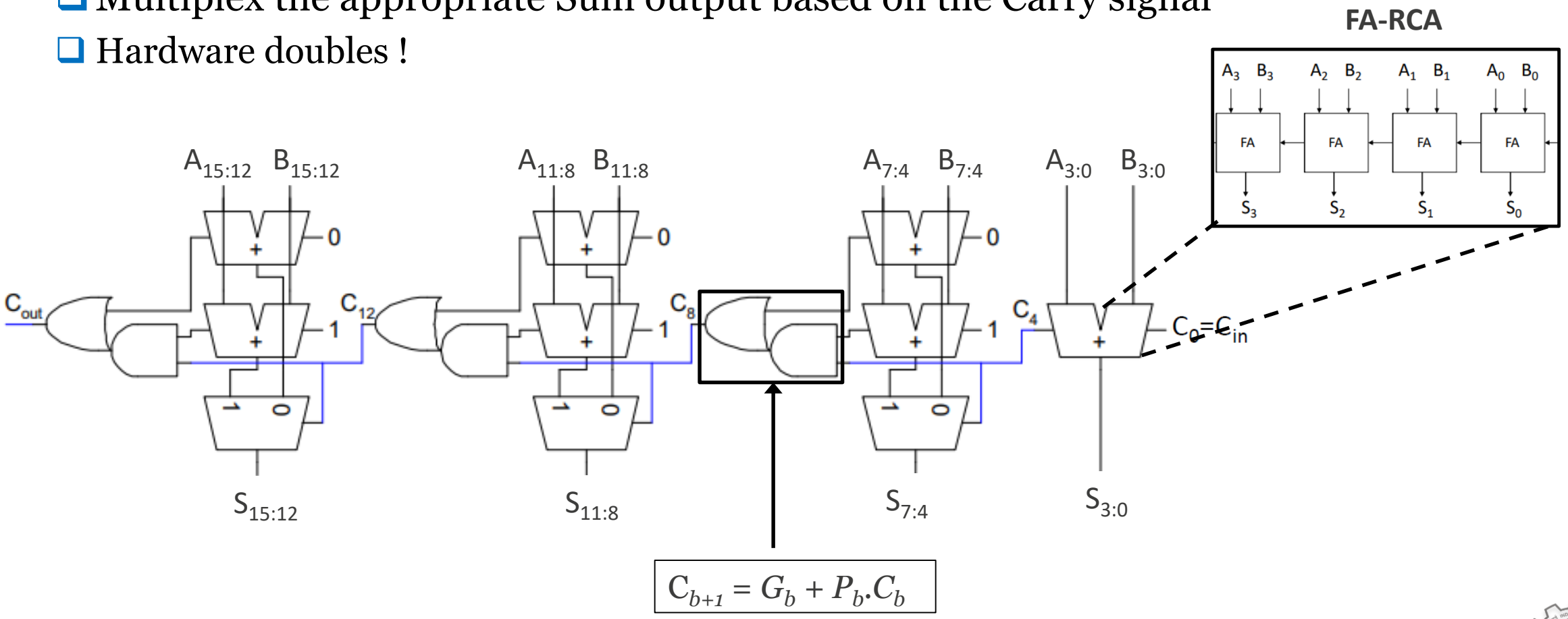
Carry Skip (Bypass) Adder using PG-RCA: Variable size



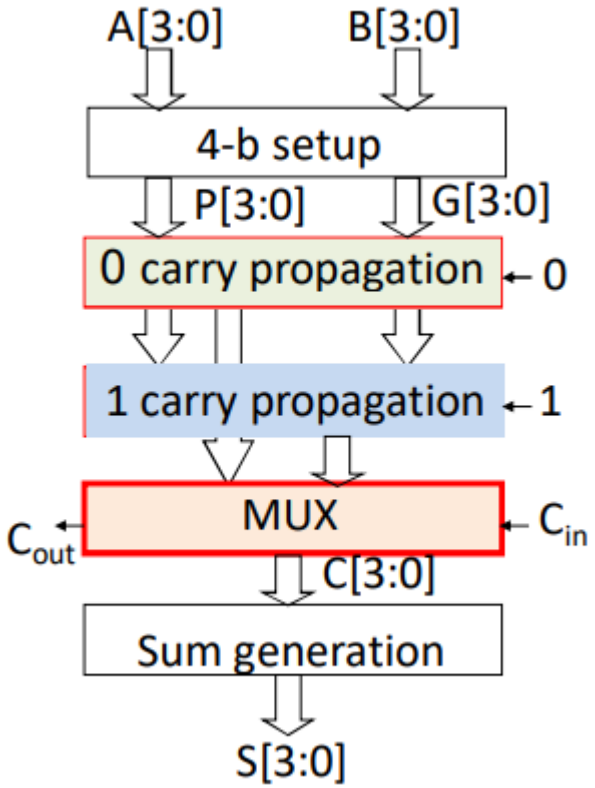
- Adder with 2,3,4,4,3 bit groups (16-bit adder)
- T_{add_wc} = ?

CARRY SELECT ADDER (CSA) USING FA-RCA

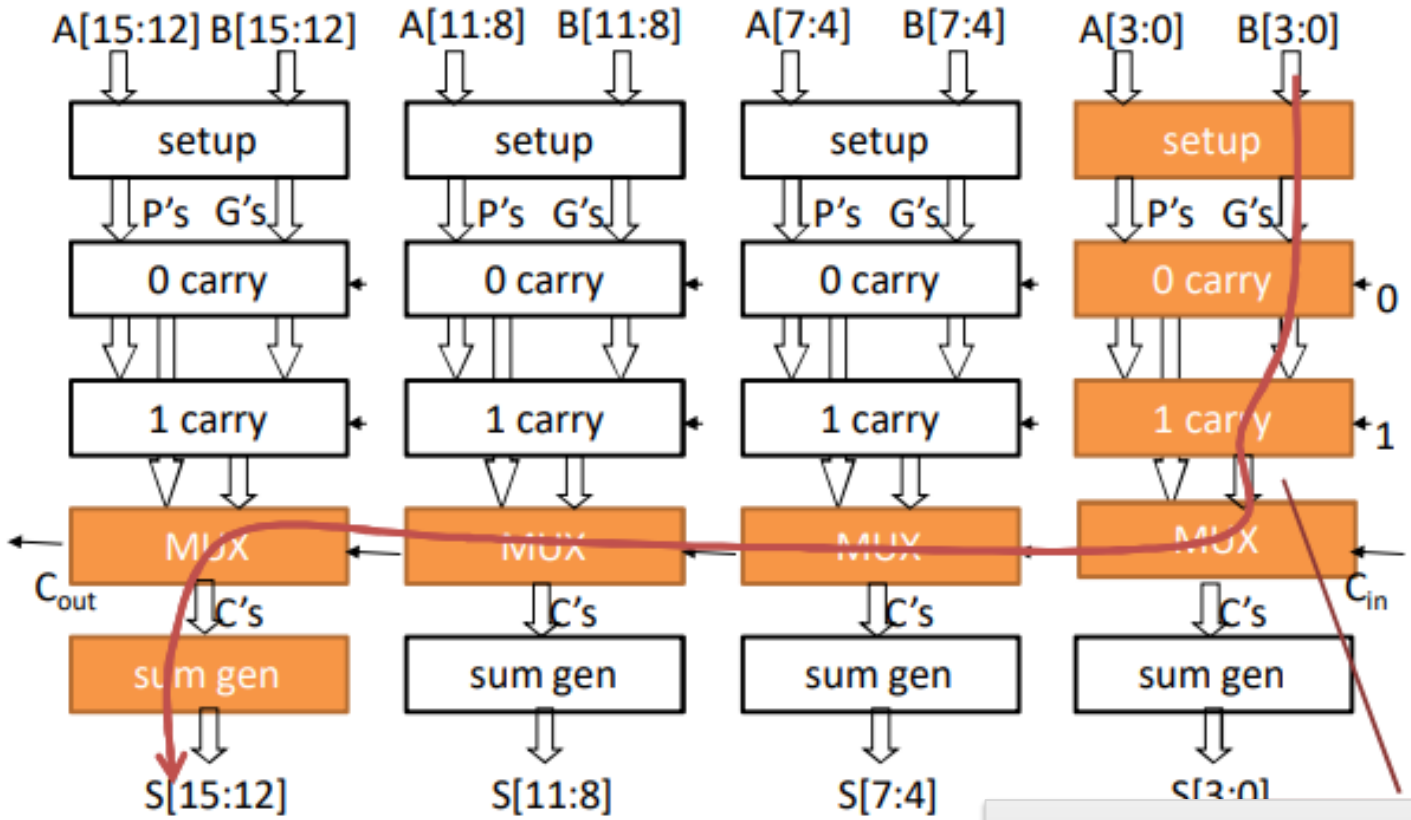
- CSA: Simple implementation version
  - Precompute Sum for both possible carry options
  - Multiplex the appropriate Sum output based on the Carry signal
  - Hardware doubles !



CARRY SELECT ADDER (CSA): USING PG-RCA



- Precompute P,G, C_{out} for both possible carry conditions
- Hardware doubles !

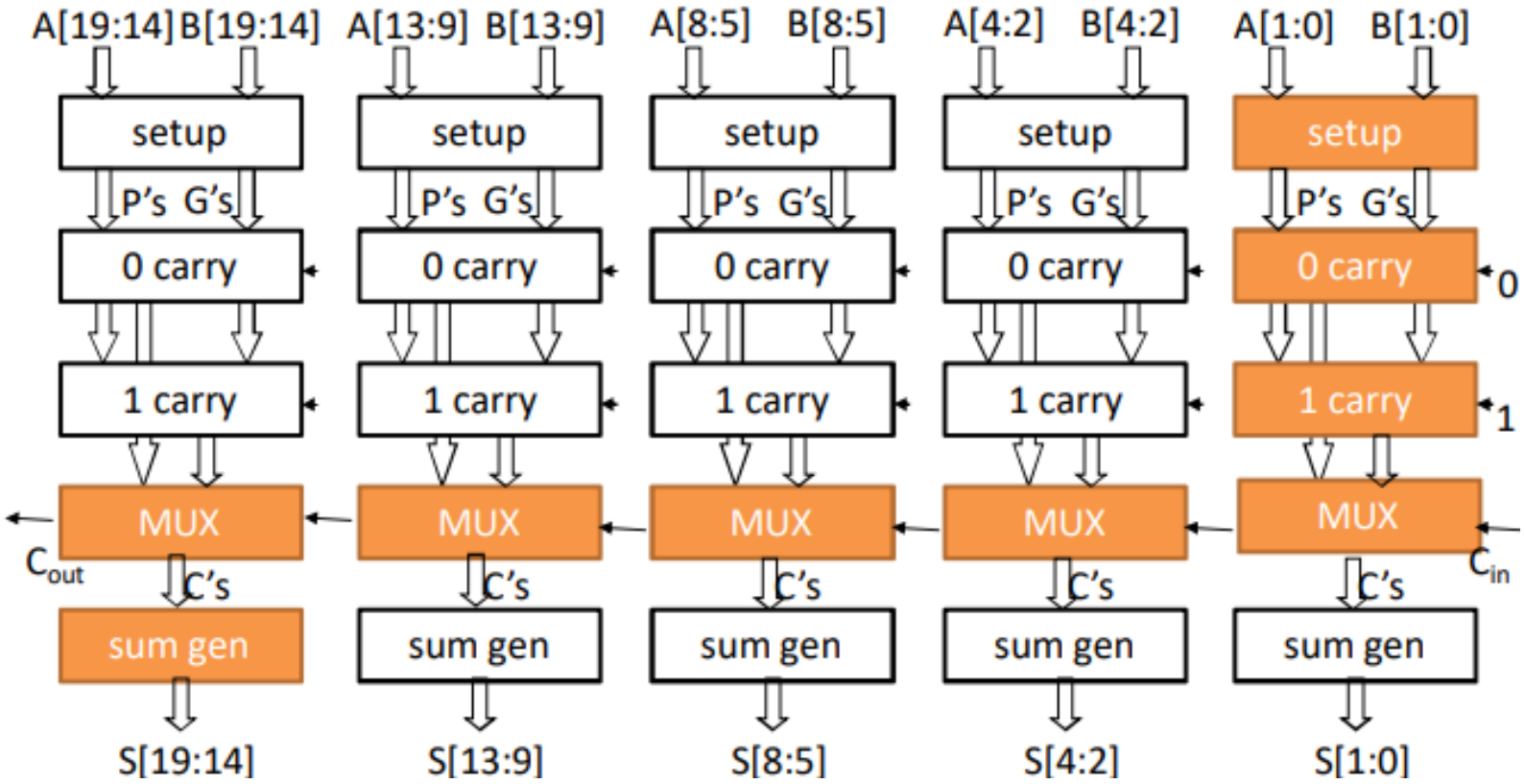


Either 0 or 1 path: critical

- $T_{adder\_wc} = t_{pg} + B.t_{AO} + (N/B).\ t_{mux} + t_{sum}$

SQUARE ROOT CSA

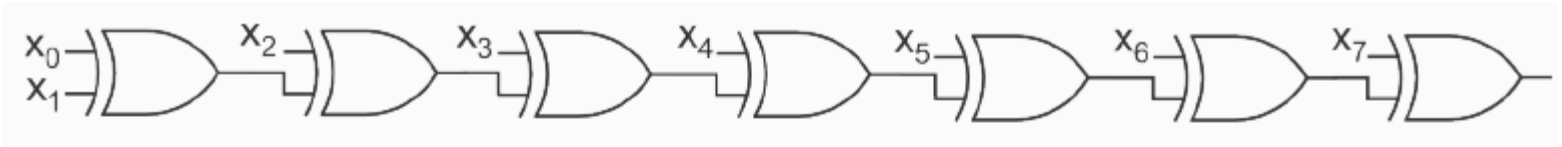
- All subsequent stages are ready with the result and waiting for the previous stage
- Instead of idle time, spend this wait time on computing additional bit addition!!
- Instead of fixed B, use increasing bit group (example: increase by 1)

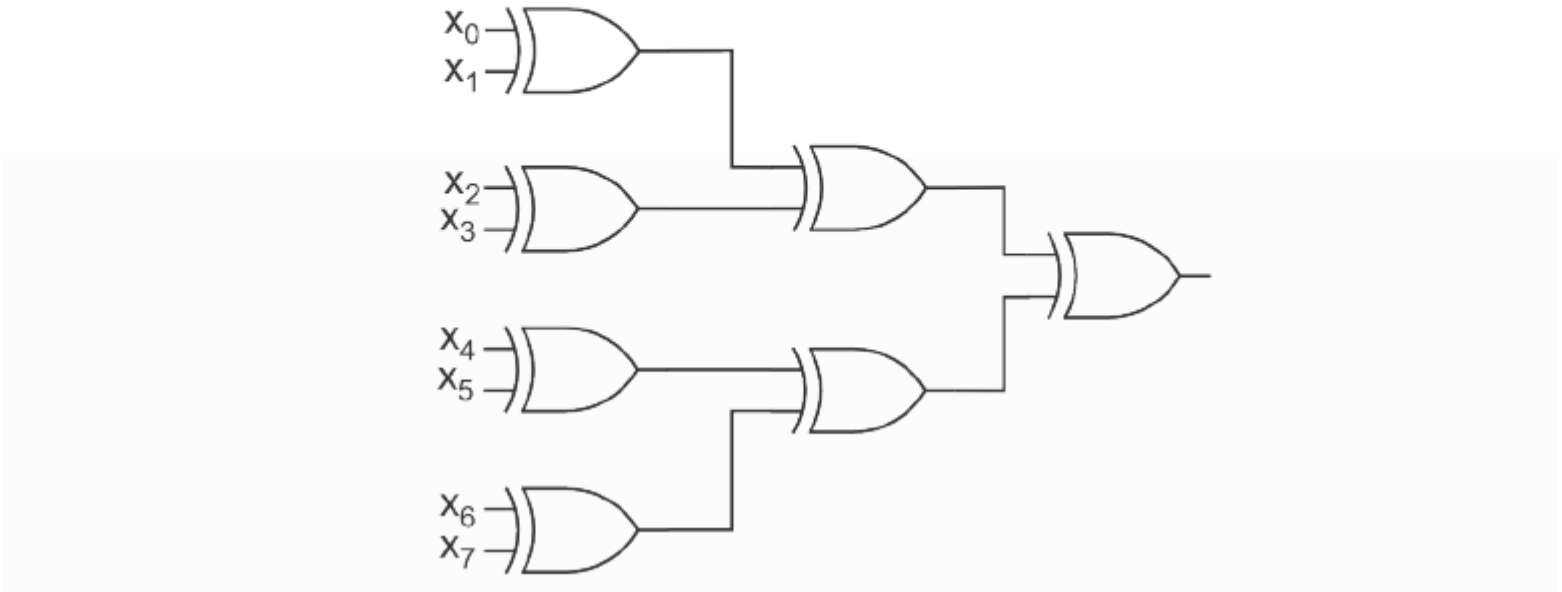


- $t_{adder\text{-}wc} = t_{pg} + 2.t_{AO} + sqrt(N).\ t_{mux} + t_{sum}$

Tree Adders: Idea

- Speed up computation: tree structure
- Similar to example below:





- Combine neighbours to produce immediate intermediate output groups

TREE ADDERS: BASICS

- Recall the carry recurrence

Recurrence: $C_{i+1}=G_i+P_iC_i$

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0\,C_0$$
$$C_2 = \boxed{(G_1 + P_1G_0)} + \boxed{(P_1P_0)}\,C_0$$

Group generate $G_{i:j}$

Group propagate $P_{i:j}$

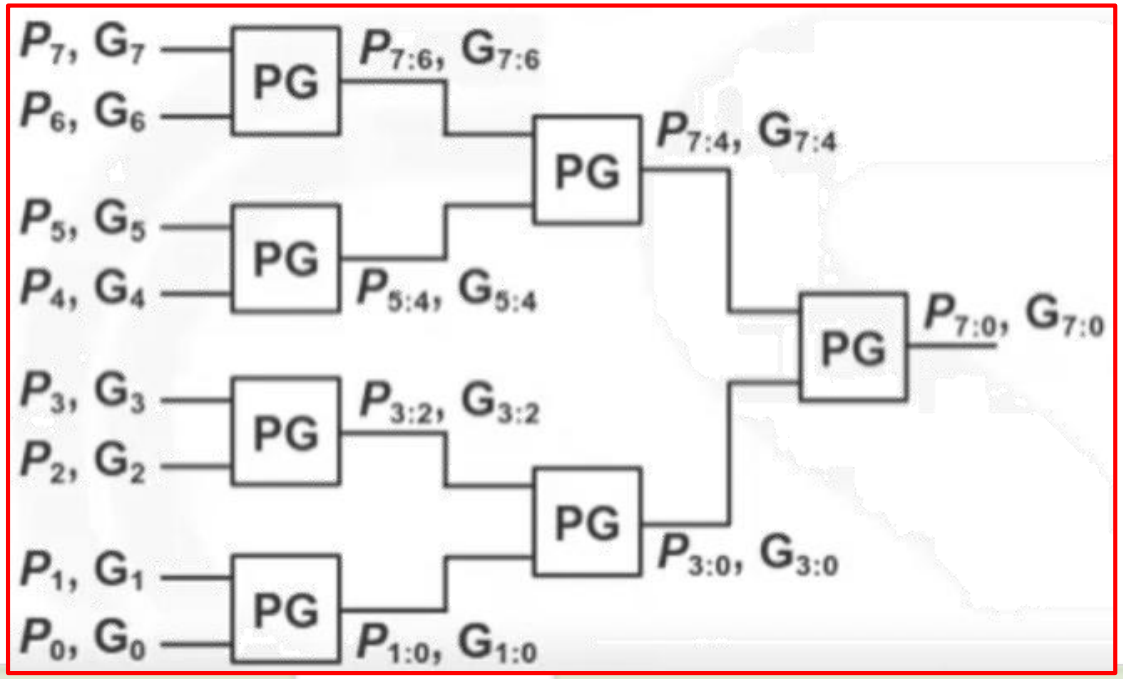
- Nomenclature: let's call the carry from present stage "*i*" as $C_{out,i}$ and the main adder carry as $C_{in,o}$

$$P_{1:0} = P_1 \cdot P_0 \quad G_{1:0} = G_1 + P_1 \cdot G_0$$
$$C_{\text{out},1} = G_{1:0} + P_{1:0}C_{\text{in},0}$$

$$P_{3:2} = P_3 \cdot P_2 \quad G_{3:2} = G_3 + P_3 \cdot G_2$$
$$C_{\text{out},3} = G_{3:2} + P_{3:2}C_{in,2}$$

$$P_{3:0} = P_{3:2} \cdot P_{1:0} \quad G_{3:0} = G_{3:2} + P_{3:2} \cdot G_{1:0}$$
$$C_{\text{out},3} = G_{3:0} + P_{3:0}C_{\text{in},0}$$

TREE ADDERS: BASIC IDEA



$$P_{1:0} = P_1 \cdot P_0 \quad G_{1:0} = G_1 + P_1 \cdot G_0$$

$$C_{\text{out},1} = G_{1:0} + P_{1:0} C_{\text{in},0}$$

$$P_{3:2} = P_3 \cdot P_2 \quad G_{3:2} = G_3 + P_3 \cdot G_2$$

$$C_{\text{out},3} = G_{3:2} + P_{3:2} C_{in,2}$$

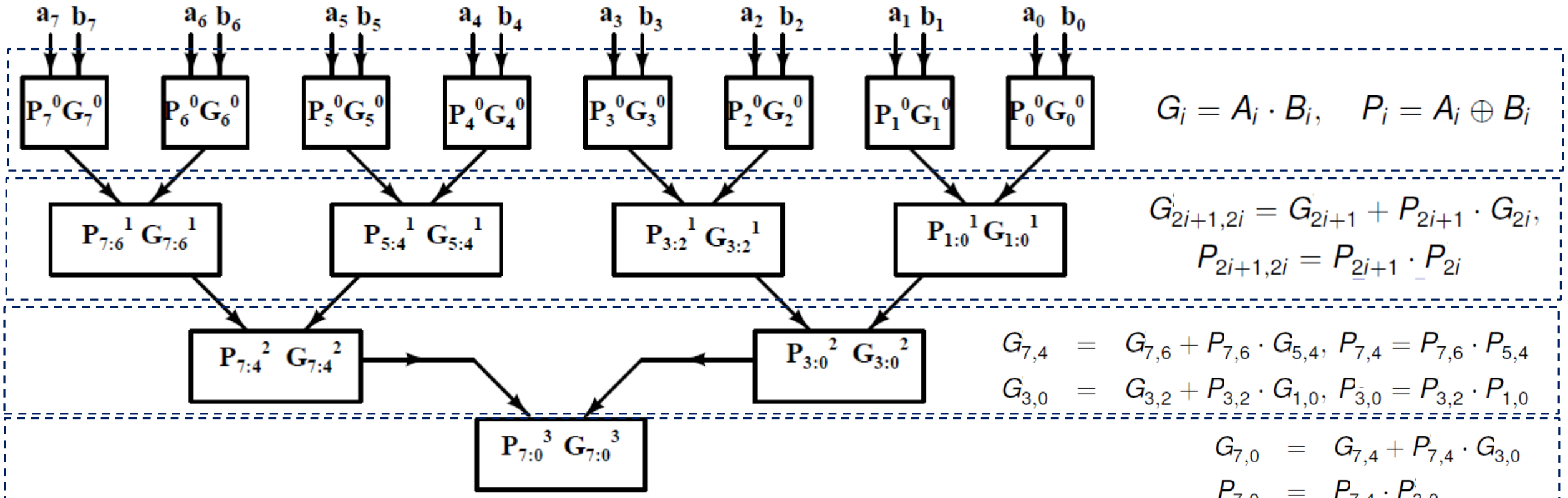
$$P_{3:0} = P_{3:2} \cdot P_{1:0} \quad G_{3:0} = G_{3:2} + P_{3:2} \cdot G_{1:0}$$

$$C_{\text{out},3} = G_{3:0} + P_{3:0} C_{\text{in},0}$$

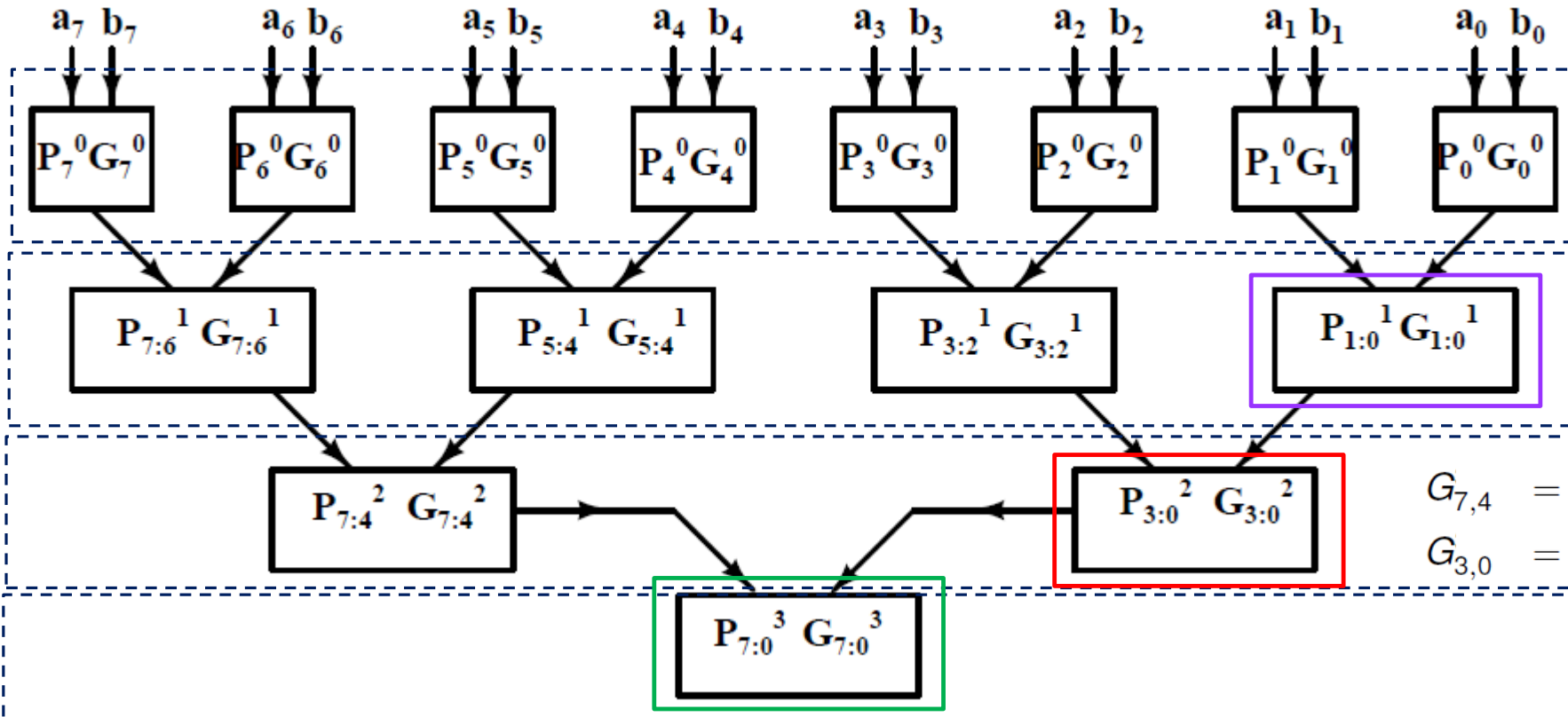
- The final C_{out} which is in the critical path can be computed using C_{in} with a tree like structure without relying on previous-stage carry !

TREE ADDERS: BRENT KUNG ADDER

- Brent Kung adder: logarithmic adder
  - P, G computed over 1,2,4,…. bits in a tree structure



TREE ADDERS: BRENT KUNG ADDER



$$G_i = A_i \cdot B_i, \quad P_i = A_i \oplus B_i$$

$$G_{2i+1,2i} = G_{2i+1} + P_{2i+1} \cdot G_{2i},$$
$$P_{2i+1,2i} = P_{2i+1} \cdot P_{2i}$$

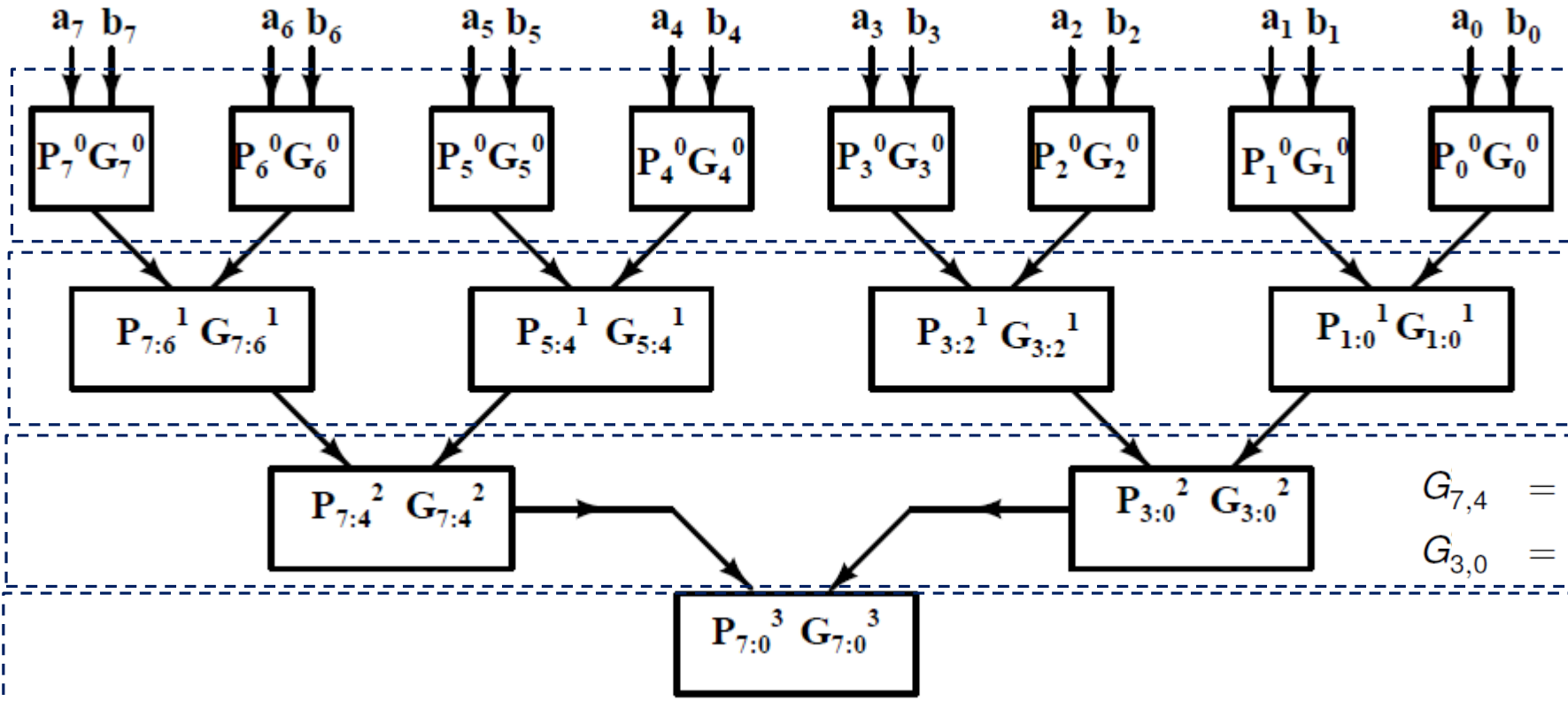
$$G_{7,4} = G_{7,6} + P_{7,6} \cdot G_{5,4}, \; P_{7,4} = P_{7,6} \cdot P_{5,4}$$
$$G_{3,0} = G_{3,2} + P_{3,2} \cdot G_{1,0}, \; P_{3,0} = P_{3,2} \cdot P_{1,0}$$

$$G_{7,0} = G_{7,4} + P_{7,4} \cdot G_{3,0}$$
$$P_{7,0} = P_{7,4} \cdot P_{3,0}$$

$$C_1 = G_{1,0} + P_{1,0} \cdot C_{in} \qquad C_3 = G_{3,0} + P_{3,0} \cdot C_{in} \qquad C_7 = G_{7,0} + P_{7,0} \cdot C_{in}$$

- Carry C_1 C_3 C_7 : generated from input carry C_{in} in a tree fashion in log-time

Tree Adders: Brent Kung Adder



$$G_i = A_i \cdot B_i, \quad P_i = A_i \oplus B_i$$

$$G_{2i+1,2i} = G_{2i+1} + P_{2i+1} \cdot G_{2i},$$
$$P_{2i+1,2i} = P_{2i+1} \cdot P_{2i}$$

$$G_{7,4} = G_{7,6} + P_{7,6} \cdot G_{5,4}, \; P_{7,4} = P_{7,6} \cdot P_{5,4}$$
$$G_{3,0} = G_{3,2} + P_{3,2} \cdot G_{1,0}, \; P_{3,0} = P_{3,2} \cdot P_{1,0}$$

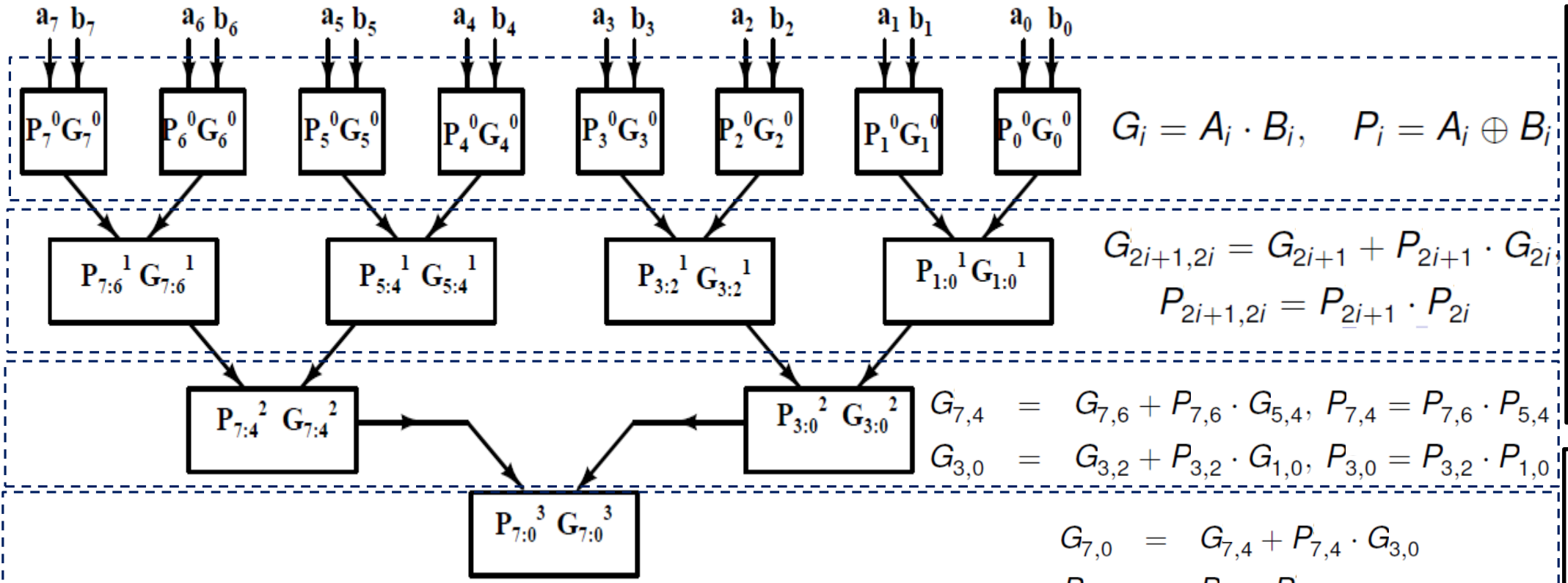
$$G_{7,0} = G_{7,4} + P_{7,4} \cdot G_{3,0}$$
$$P_{7,0} = P_{7,4} \cdot P_{3,0}$$

$$C_1 = G_{1,0} + P_{1,0} \cdot C_{in} \qquad C_3 = G_{3,0} + P_{3,0} \cdot C_{in} \qquad C_7 = G_{7,0} + P_{7,0} \cdot C_{in}$$

$$C_0 = G_0 + P_0 \cdot C_{in} \qquad C_2 = G_2 + P_2 \cdot C_1, \qquad C_4 = G_4 + P_4 \cdot C_3 \quad \text{and so on ....}$$

$$\text{Sum}_i = P_i \oplus C_i$$

TREE ADDERS: BRENT KUNG ADDER

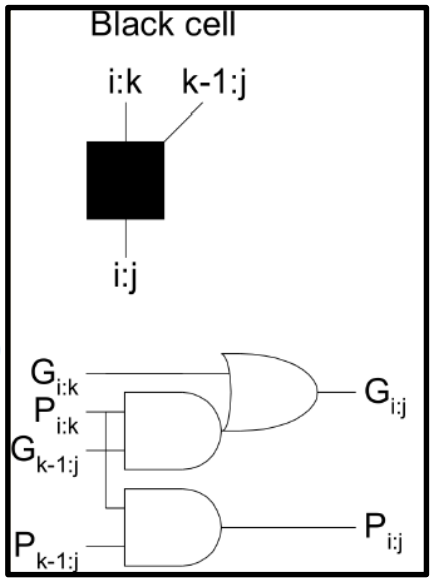


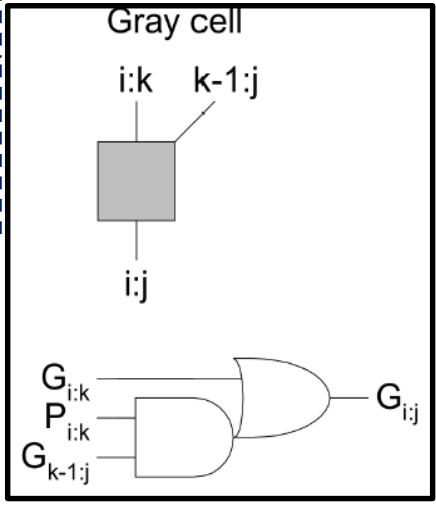
$$G_i = A_i \cdot B_i, \quad P_i = A_i \oplus B_i$$

$$G_{2i+1,2i} = G_{2i+1} + P_{2i+1} \cdot G_{2i}$$
$$P_{2i+1,2i} = P_{2i+1} \cdot P_{2i}$$

$$G_{7,4} = G_{7,6} + P_{7,6} \cdot G_{5,4}, \; P_{7,4} = P_{7,6} \cdot P_{5,4}$$
$$G_{3,0} = G_{3,2} + P_{3,2} \cdot G_{1,0}, \; P_{3,0} = P_{3,2} \cdot P_{1,0}$$

$$G_{7,0} = G_{7,4} + P_{7,4} \cdot G_{3,0}$$
$$P_{7,0} = P_{7,4} \cdot P_{3,0}$$





$$C_1 = G_{1,0} + P_{1,0} \cdot C_{in} \qquad C_3 = G_{3,0} + P_{3,0} \cdot C_{in} \qquad C_7 = G_{7,0} + P_{7,0} \cdot C_{in}$$

$$C_0 = G_0 + P_0 \cdot C_{in} \qquad C_2 = G_2 + P_2 \cdot C_1, \qquad C_4 = G_4 + P_4 \cdot C_3 \quad \text{and so on ....}$$

$$\text{Sum}_i = P_i \oplus C_i$$