EE671: VLSI Design
Spring 2024/25

Laxmeesha Somappa
Department of Electrical Engineering
IIT Bombay
laxmeesha@ee.iitb.ac.in

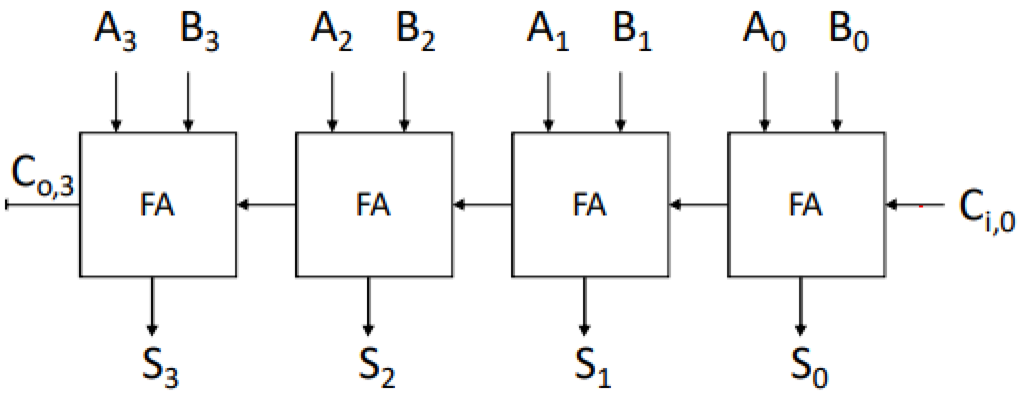
LECTURE – 15
ARITHMETIC IP: ADDERS

laxmeesha@ee.iitb.ac.in

THE CARRY RECURRENCE

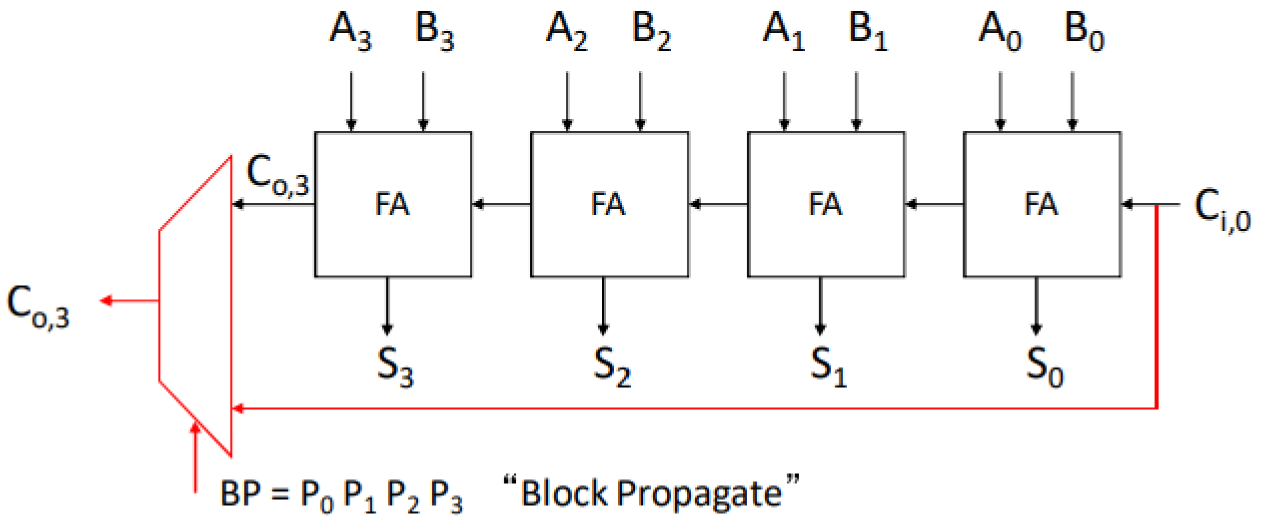
Recurrence: $C_{i+1}=G_i+P_iC_i$

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0\,C_0$$
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0\,C_0$$
$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0\,C_0$$



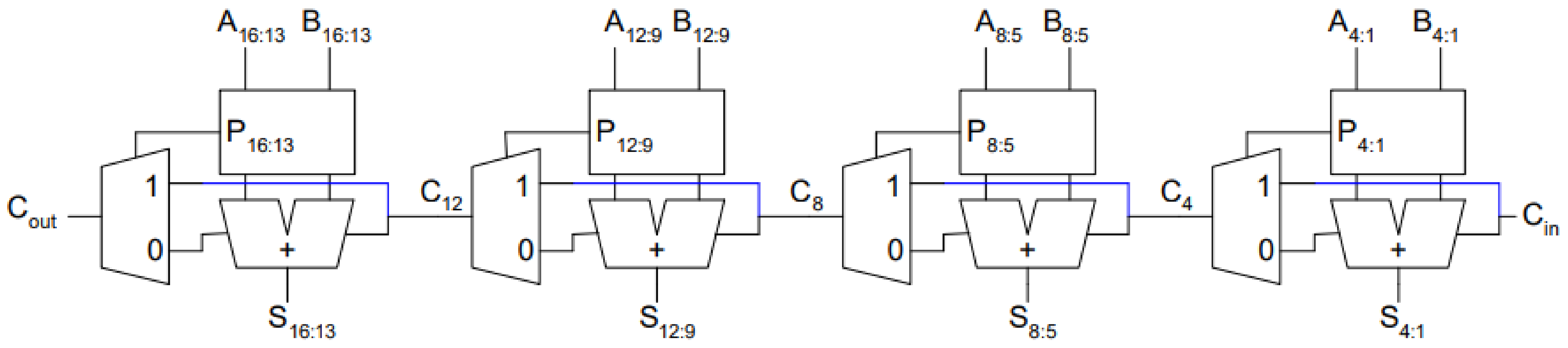
- Consider a 4 bit adder.
- The final Cout (C_4 in this case) can be directly generated from C_0 (neglecting the hardware complexity)
- Practically: to generate C_4 we need high fan-in CMOS gates (max 5 input AND)
- Generating C4 will have the same order of delay as RCA.

CARRY SKIP (BYPASS) ADDER



- Compute P and G for each bit parallelly
- If $P_0.P_1.P_2.P_3 = 1$, 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

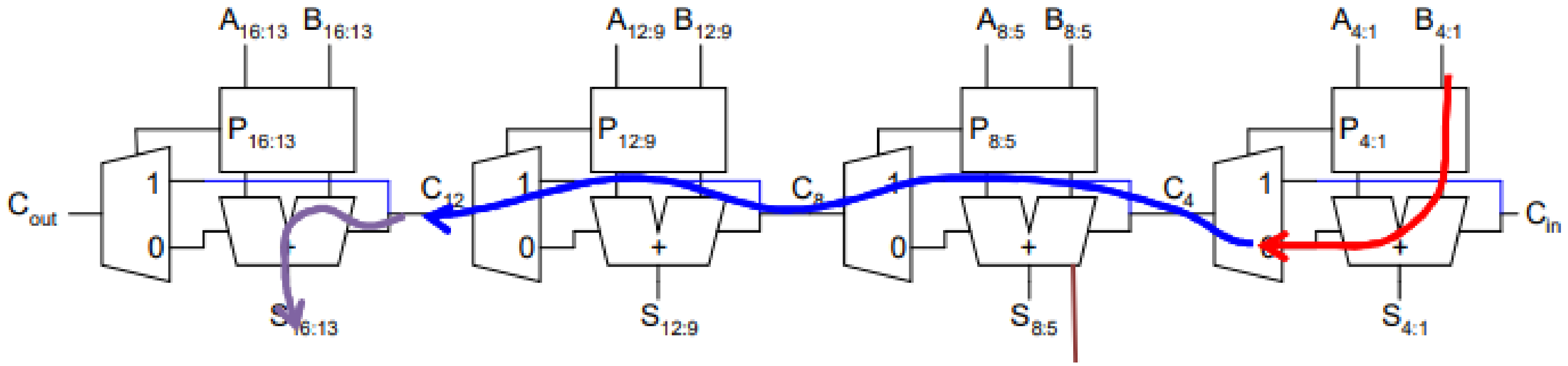
Carry Skip (Bypass) Adder



$$\boxed{S = P \oplus C \text{ and } C = G + P.Cin}$$

- 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

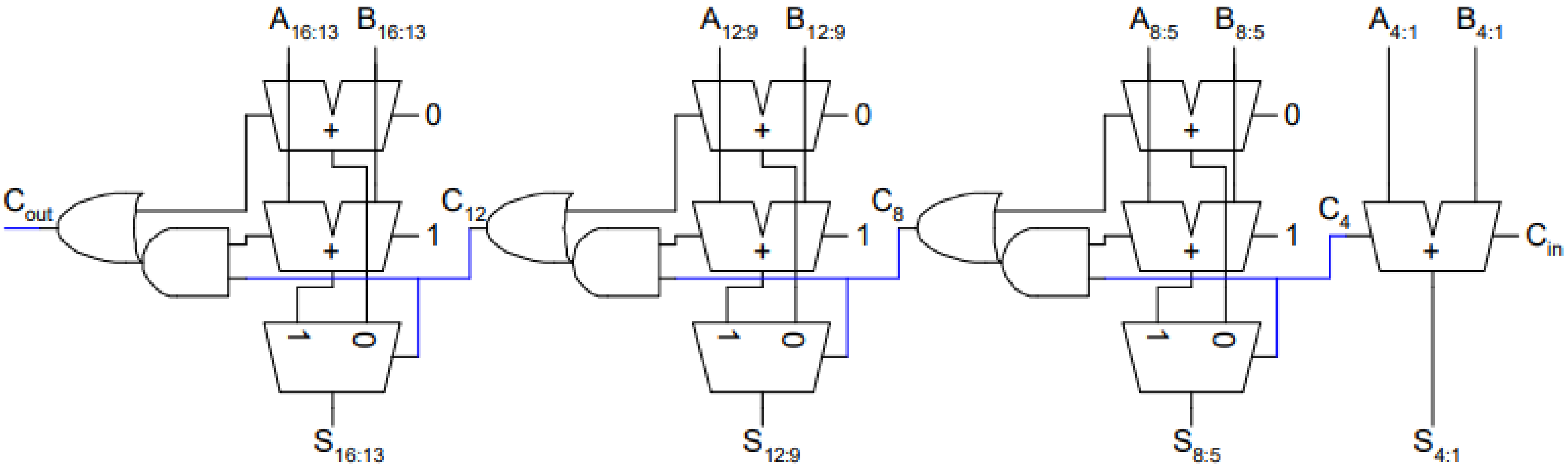
CARRY SKIP (BYPASS) ADDER



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

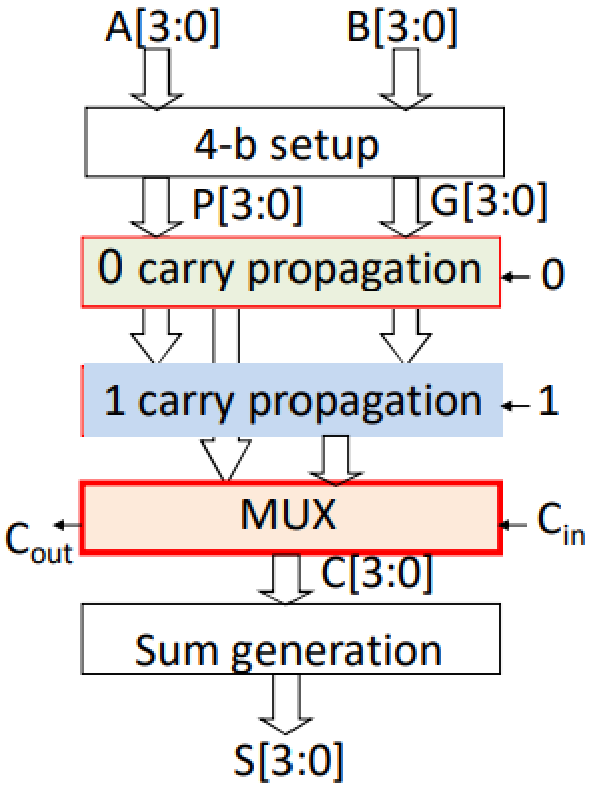
Carry Select Adder (CSA)

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

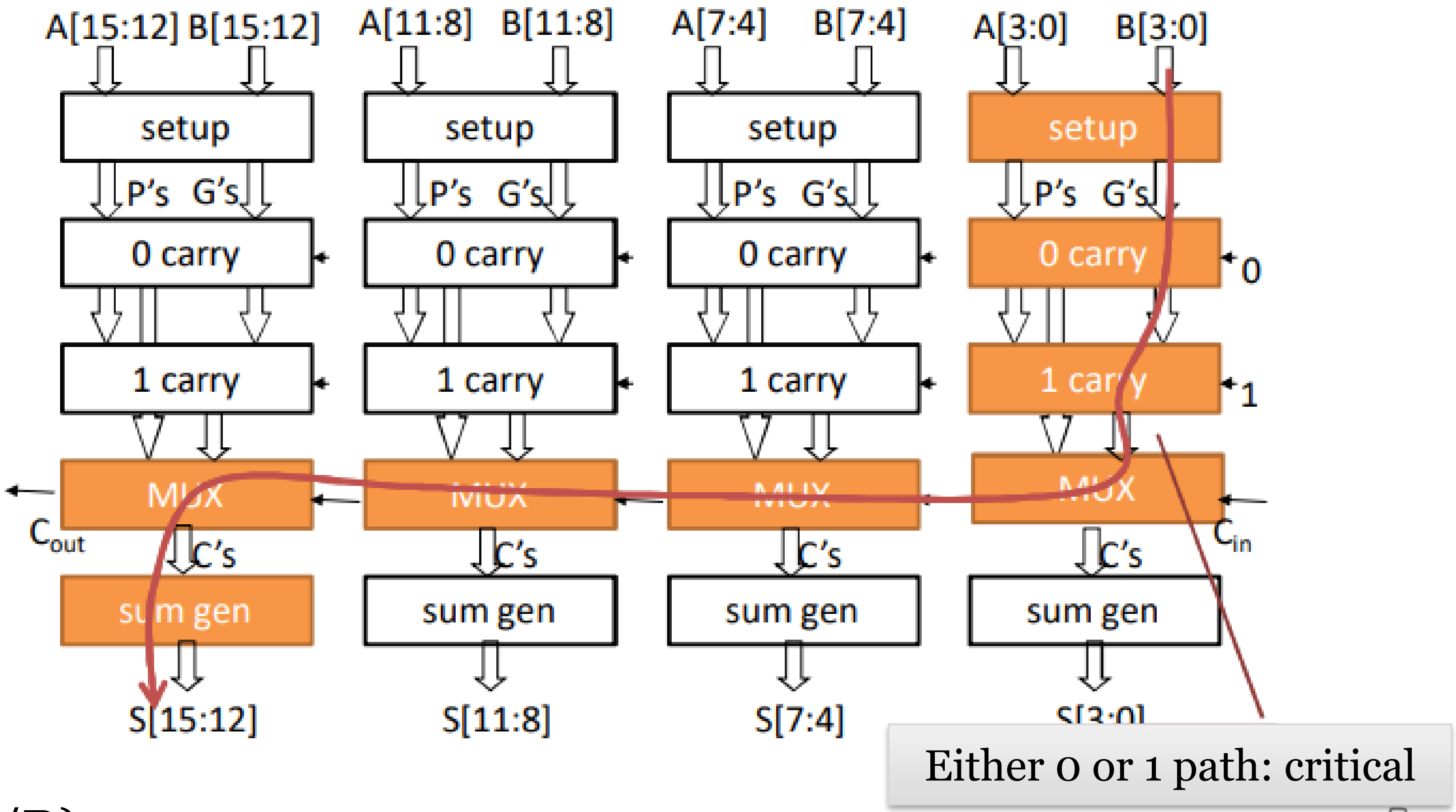


laxmeesha@ee.iitb.ac.in

CARRY SELECT ADDER (CSA): PRACTICAL IMPLEMENTATION



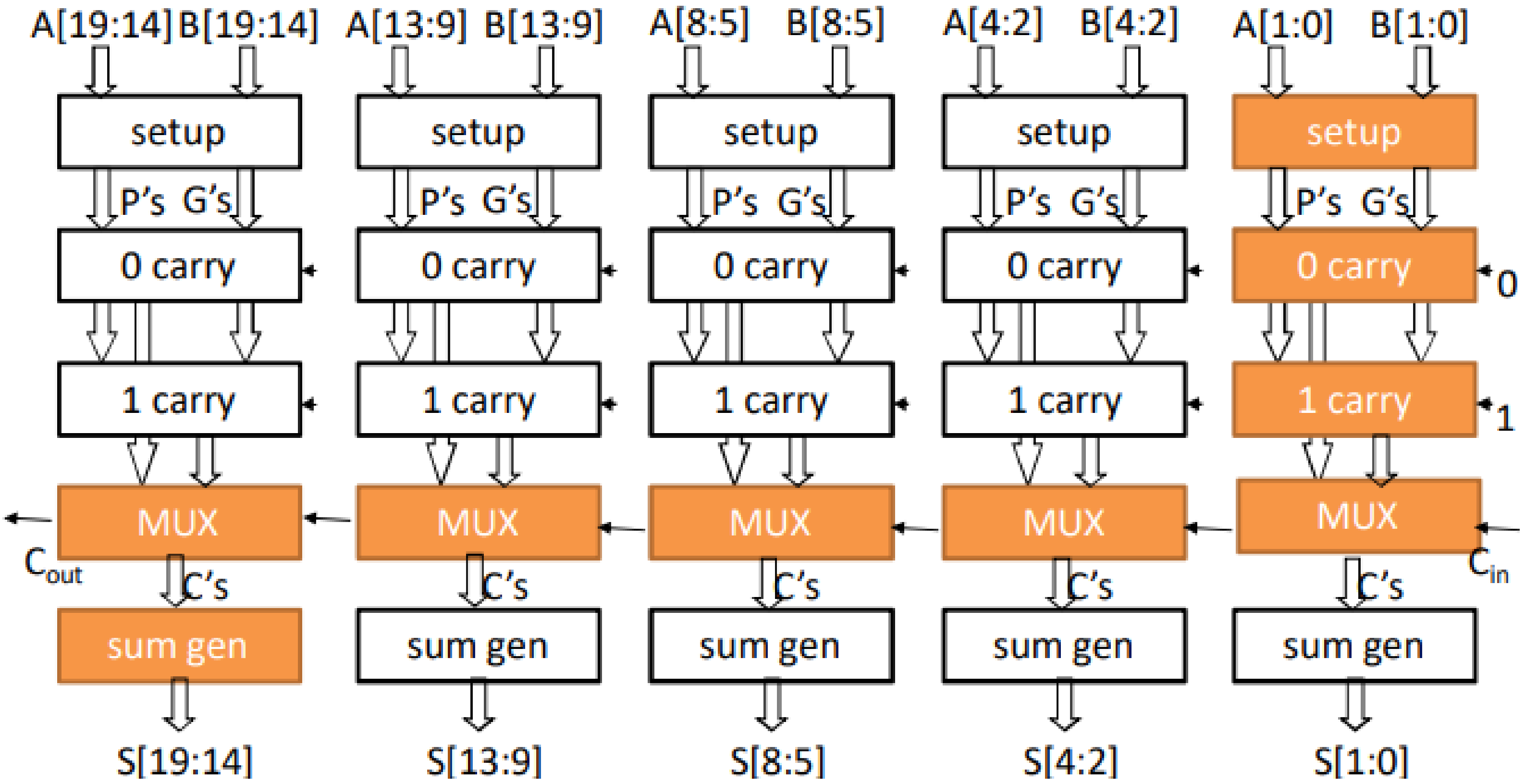
- Precompute P,G, Cout for both possible carry conditions
- Hardware doubles !



- $t_{adder\text{-}wc} = t_{pg} + B.t_{carry} + (N/B).\ t_{mux} + t_{sum}$

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)



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

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 $P_{i:j}$

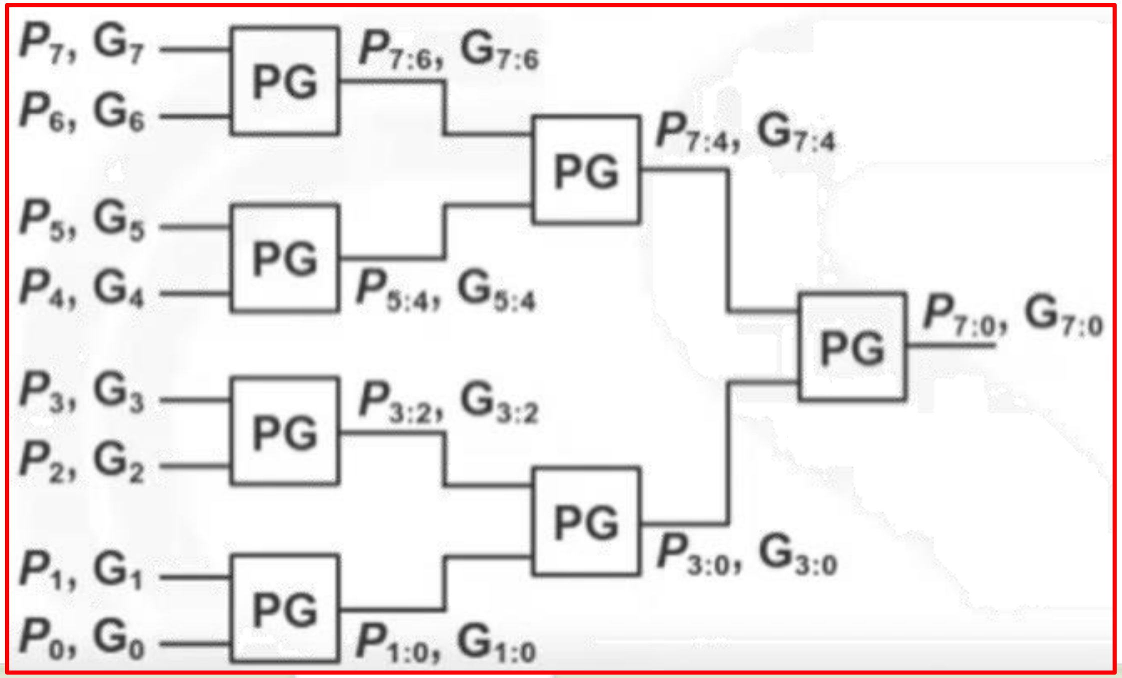
- Nomenclature: let's call the carry from present stage "*i*" as $C_{out,i}$ and the main adder carry as $C_{in,o}$

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

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

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

TREE ADDERS: BASIC IDEA



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

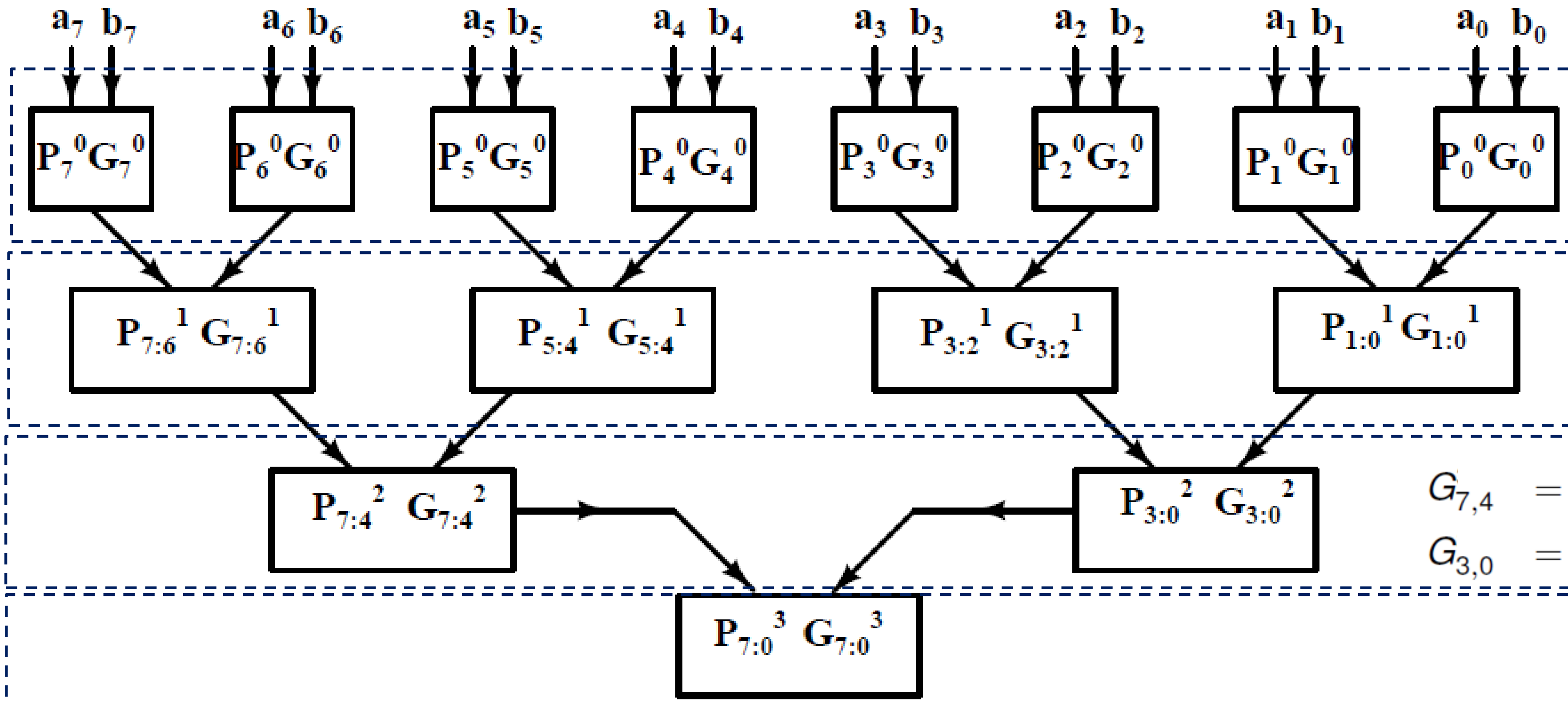
$$P_{3:2} = P_3 \cdot P_2 \quad G_{3:2} = G_3 + P_3 \cdot G_2$$
$$C_{out,3} = G_{3:2} + P_{3:2} C_{in,2}$$

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

- The final C_{out} which is in the critical path can be computed using C_{in} using 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



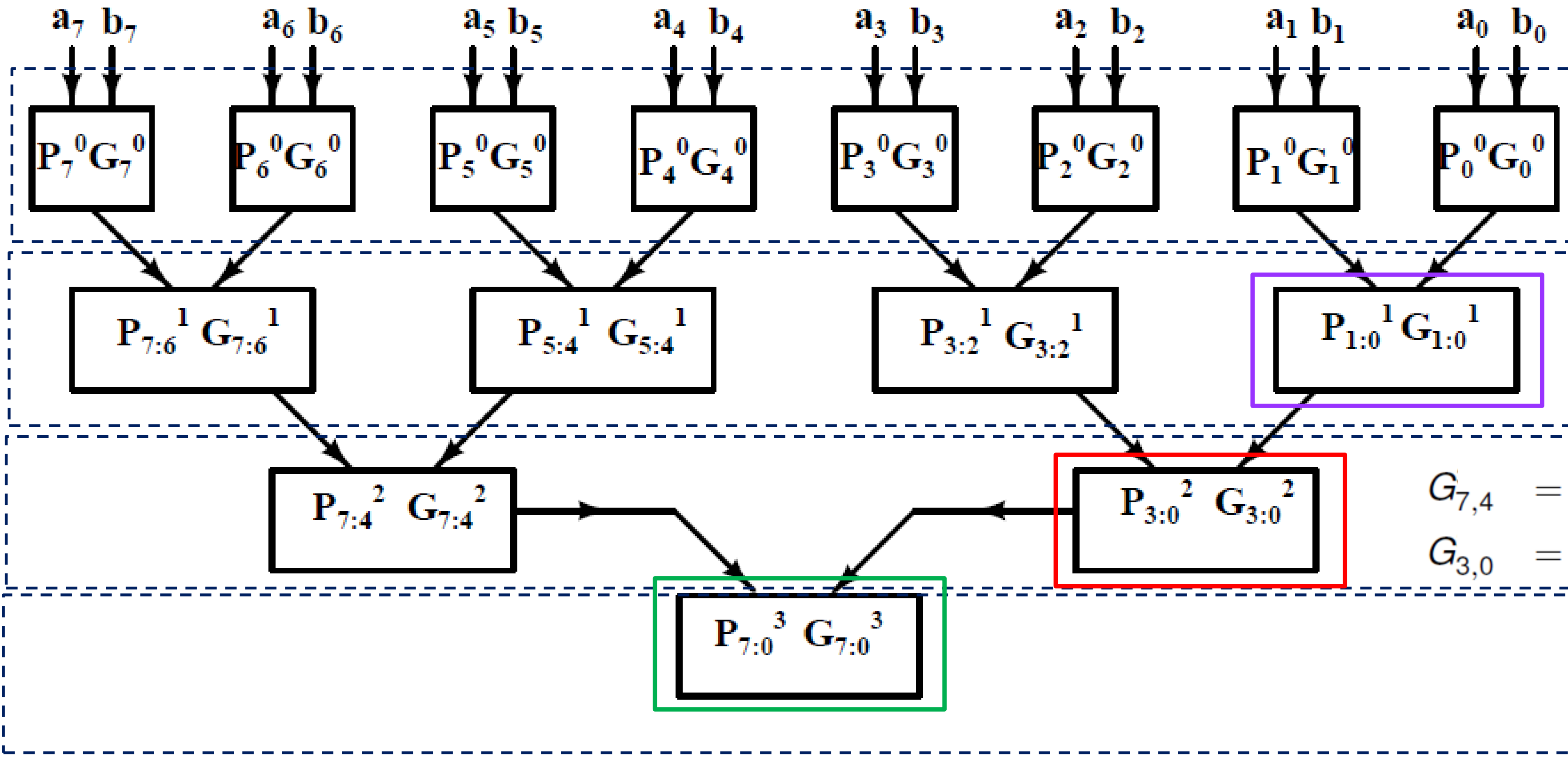
$$G_i = A_i \cdot B_i, \quad P_i = A_i \oplus B_i$$

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

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

$$\begin{aligned} G_{7,0} &= G_{7,4} + P_{7,4} \cdot G_{3,0} \\ P_{7,0} &= P_{7,4} \cdot P_{3,0} \end{aligned}$$

Tree Adders: Brent Kung Adder



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

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

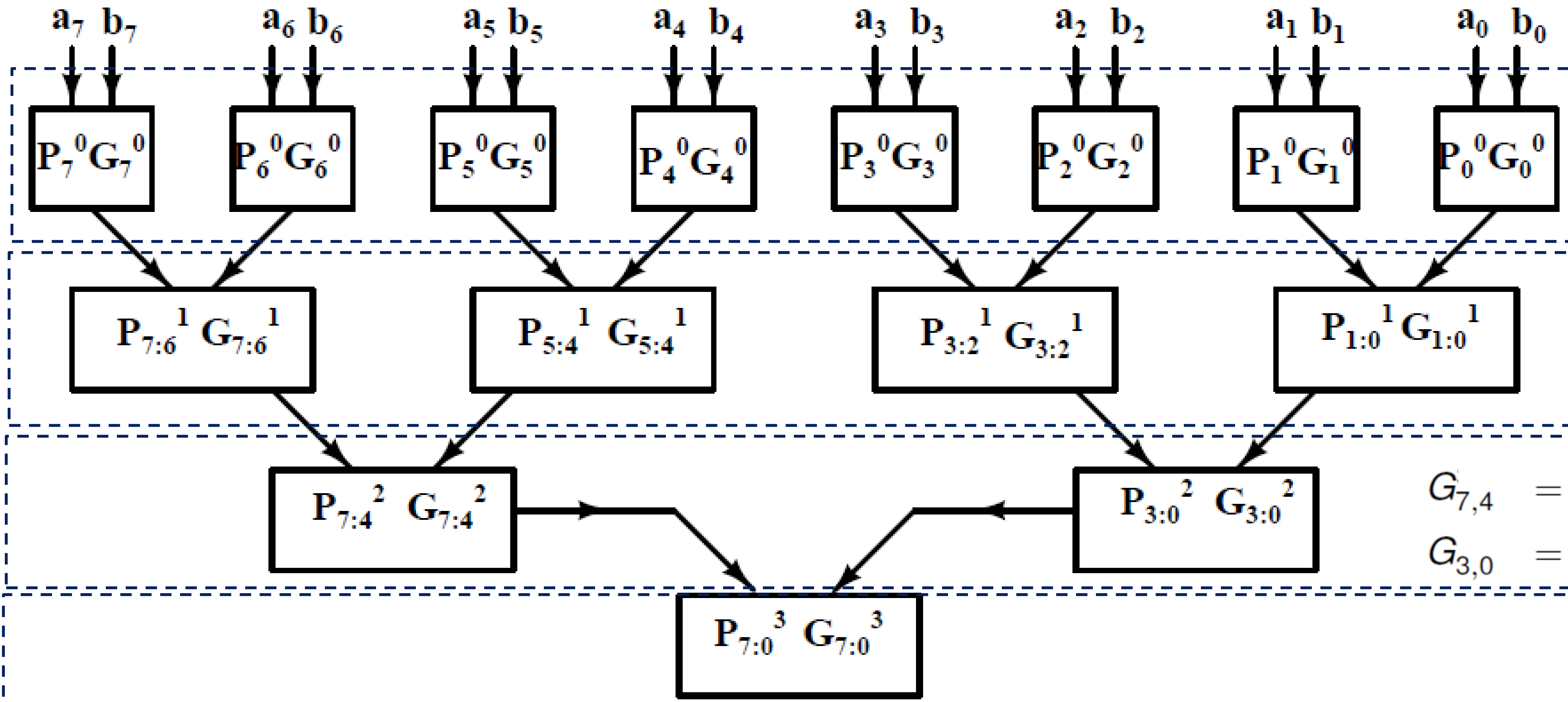
$$G_{7,4} = G_{7,6} + P_{7,6} \cdot G_{5,4}, \quad P_{7,4} = P_{7,6} \cdot P_{5,4}$$
$$G_{3,0} = G_{3,2} + P_{3,2} \cdot G_{1,0}, \quad P_{3,0} = P_{3,2} \cdot P_{1,0}$$

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

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

- 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



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

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

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

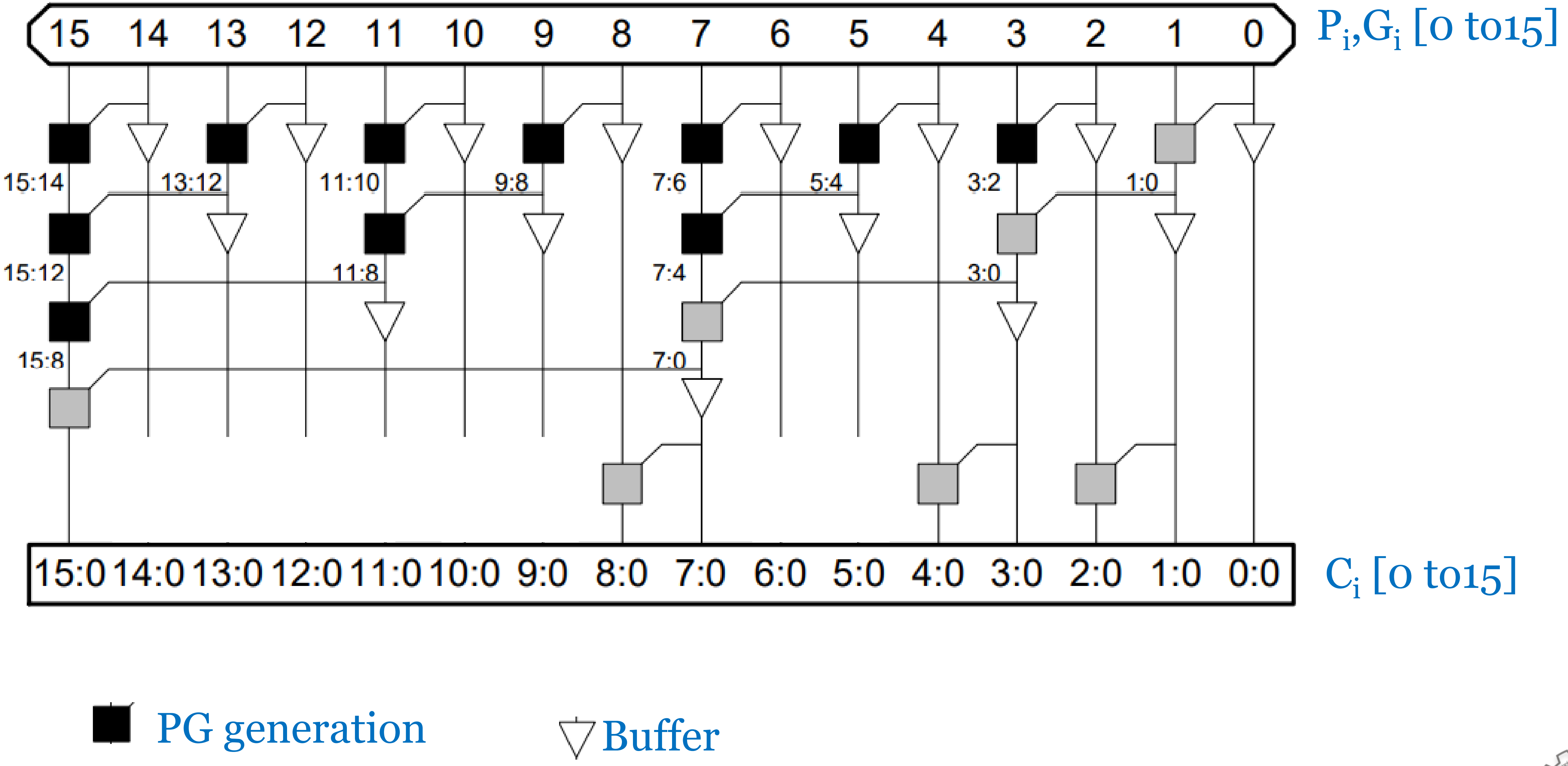
$$G_{7,0} = G_{7,4} + P_{7,4} \cdot G_{3,0}$$
$$P_{7,0} = P_{7,4} \cdot P_{3,0}$$

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

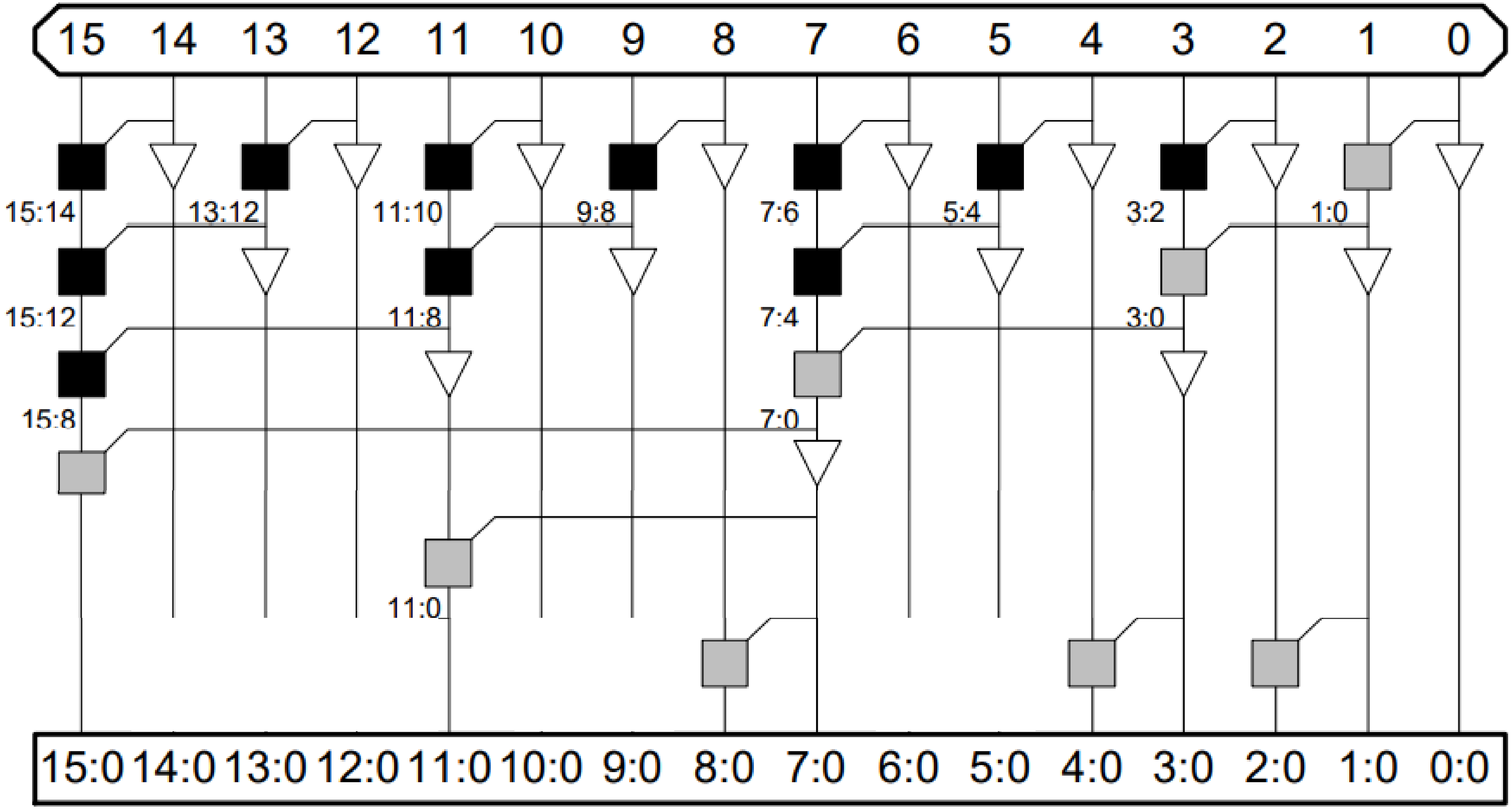
$C_0 = G_0 + P_0 \cdot C_{in}$ $\quad C_2 = G_2 + P_2 \cdot C_1,$ $\quad C_4 = G_4 + P_4 \cdot C_3$ and so on ....

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

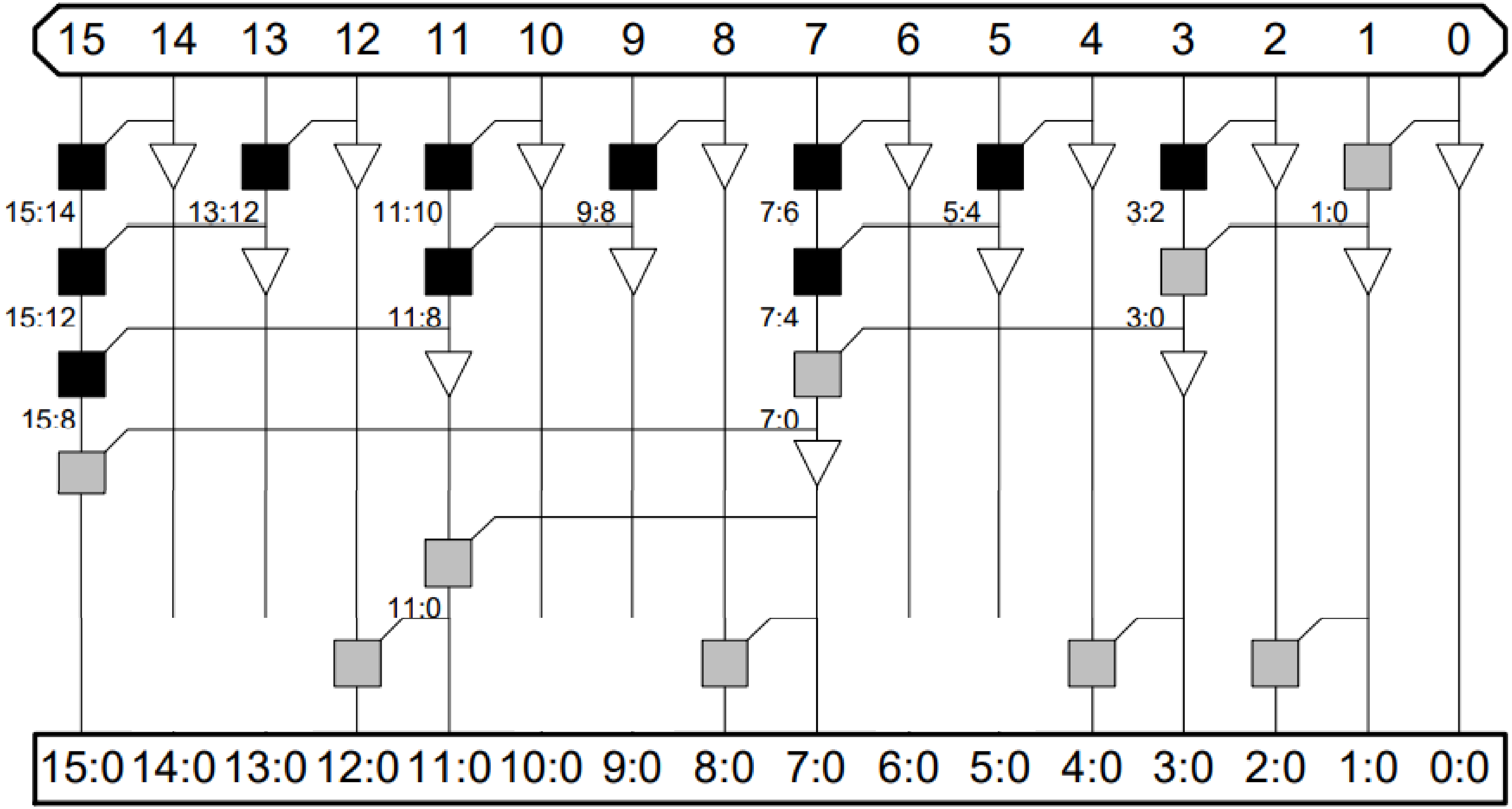
Tree Adders: 16 bit Brent Kung Adder



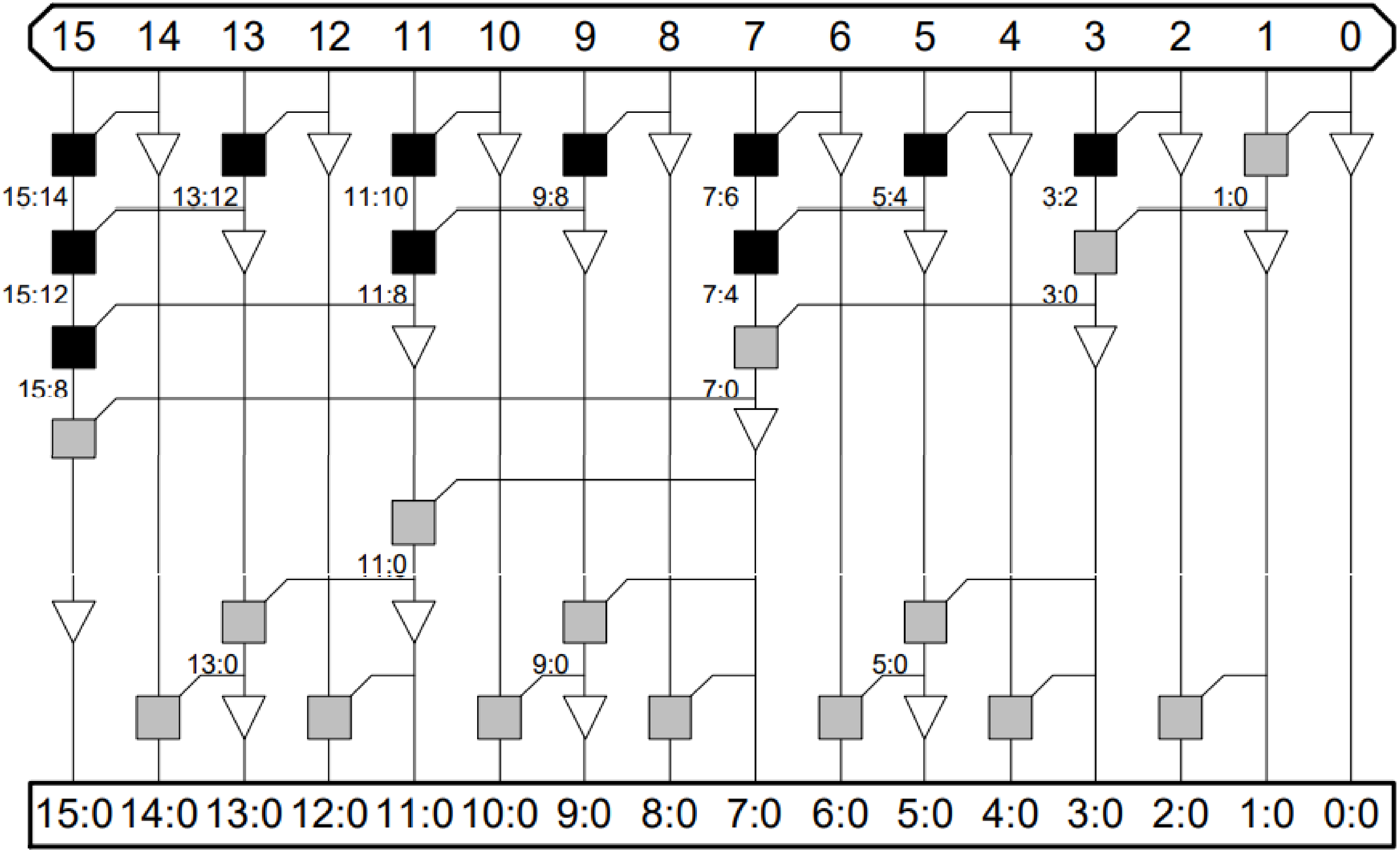
Tree Adders: 16 bit Brent Kung Adder



Tree Adders: 16 bit Brent Kung Adder



Tree Adders: 16 bit Brent Kung Adder



Tree Adders: 16 bit Kogge-Stone Adder