Digital Integrated Circuits

Arithmetic Circuits

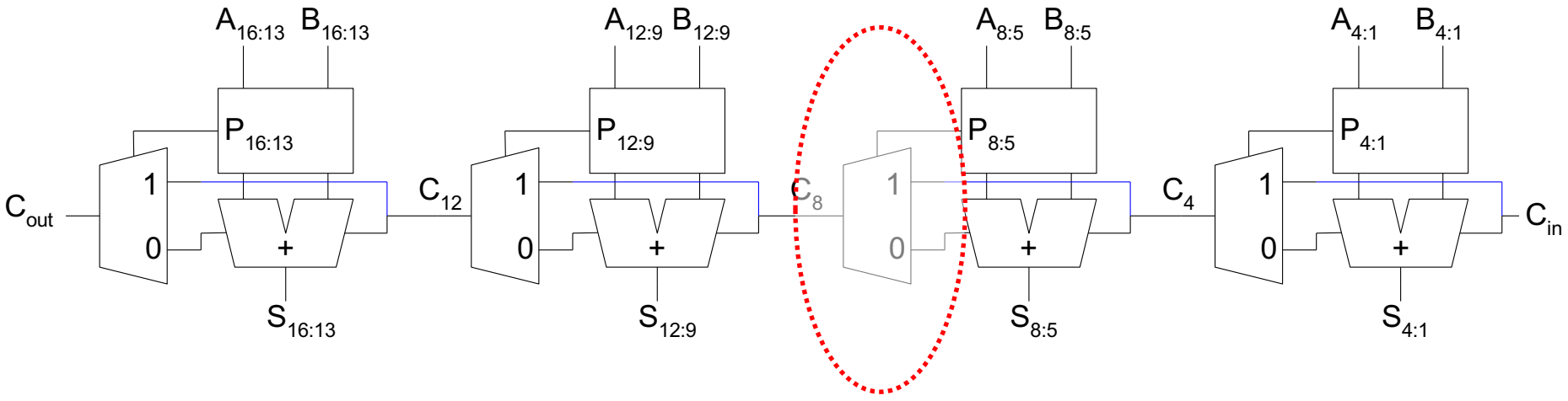
Fuyuzhuo

Outline

- Single-bit Addition
- Carry-Ripple Adder
- ***Carry-Skip Adder***
- Carry-Lookahead Adder
- Carry-Select Adder
- Carry-Increment Adder
- Tree Adder

Carry-Skip Adder

- ***Carry-ripple is slow through all N stages***
- Carry-skip allows carry to skip over groups of n bits
  - Decision based on n-bit propagate signal



Multiplexer

$$C_o = G + PC_{in} = G\bar{P} + PC_{in} = A\bar{P} + PC_{in} = B\bar{P} + PC_{in}$$

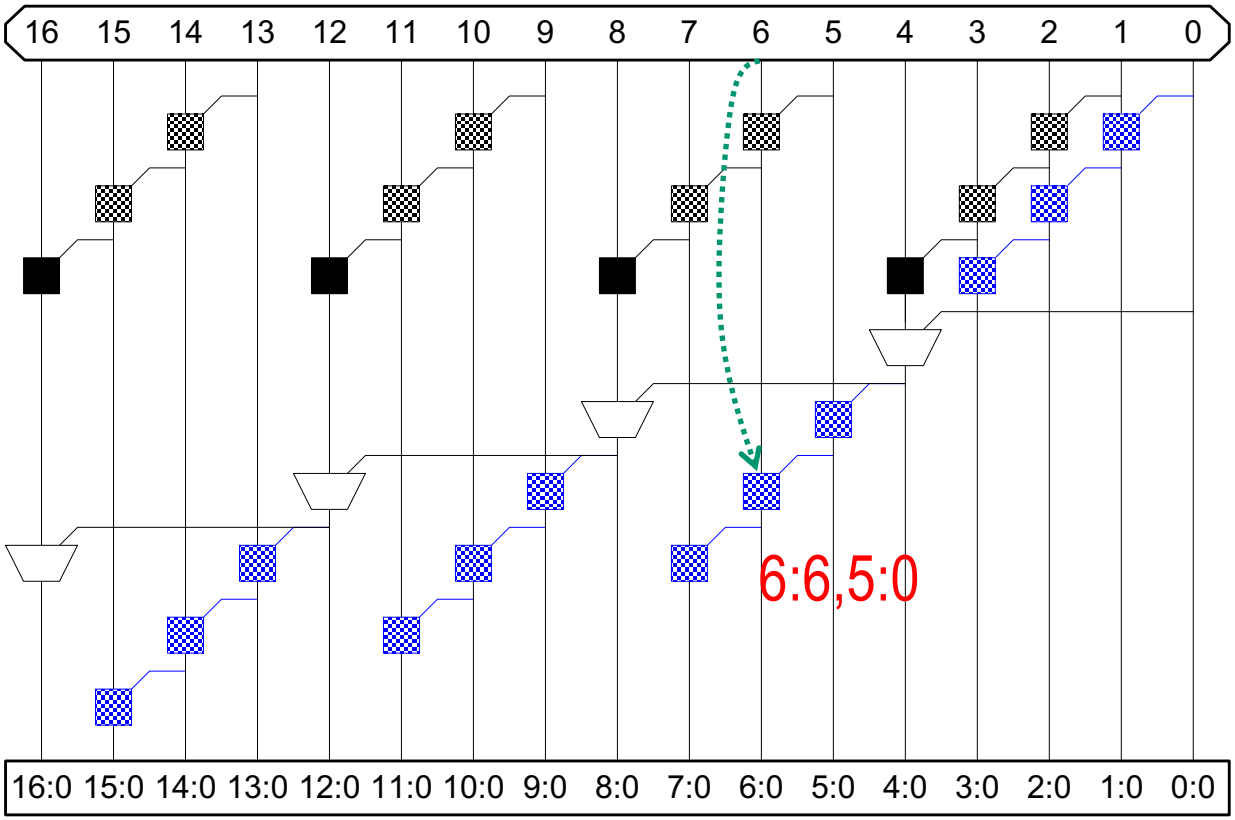
$$G_{i:j} = \underbrace{G_{i:k} + P_{i:k}G_{k-1:j}}_{\textbf{AND-OR}} = \underbrace{G_{i:k}\overline{P_{i:k}} + P_{i:k}G_{k-1:j}}_{\textbf{Multiplexer}}$$

$if\ P_{i:k} = 0 \qquad left = right$

$if\ P_{i:k} = 1 \qquad left = G_{i:k} + P_{i:k}G_{k-1:j} = P_{i:k}G_{k-1:j}$

$$right = P_{i:k}G_{k-1:j}$$

Carry-Skip PG Diagram



$$G_{4:0} = G_{4:1} + P_{4:1}G_{0:0}$$
$$G_{8:0} = G_{8:5} + P_{8:5}G_{4:0}$$
$$G_{12:0} = G_{12:9} + P_{12:9}G_{8:0}$$
$$G_{16:0} = G_{16:13} + P_{16:13}G_{12:0}$$

Note:P=A ⊕ B.

Why group?

$$C_i = g_i + p_i C_{i-1}$$

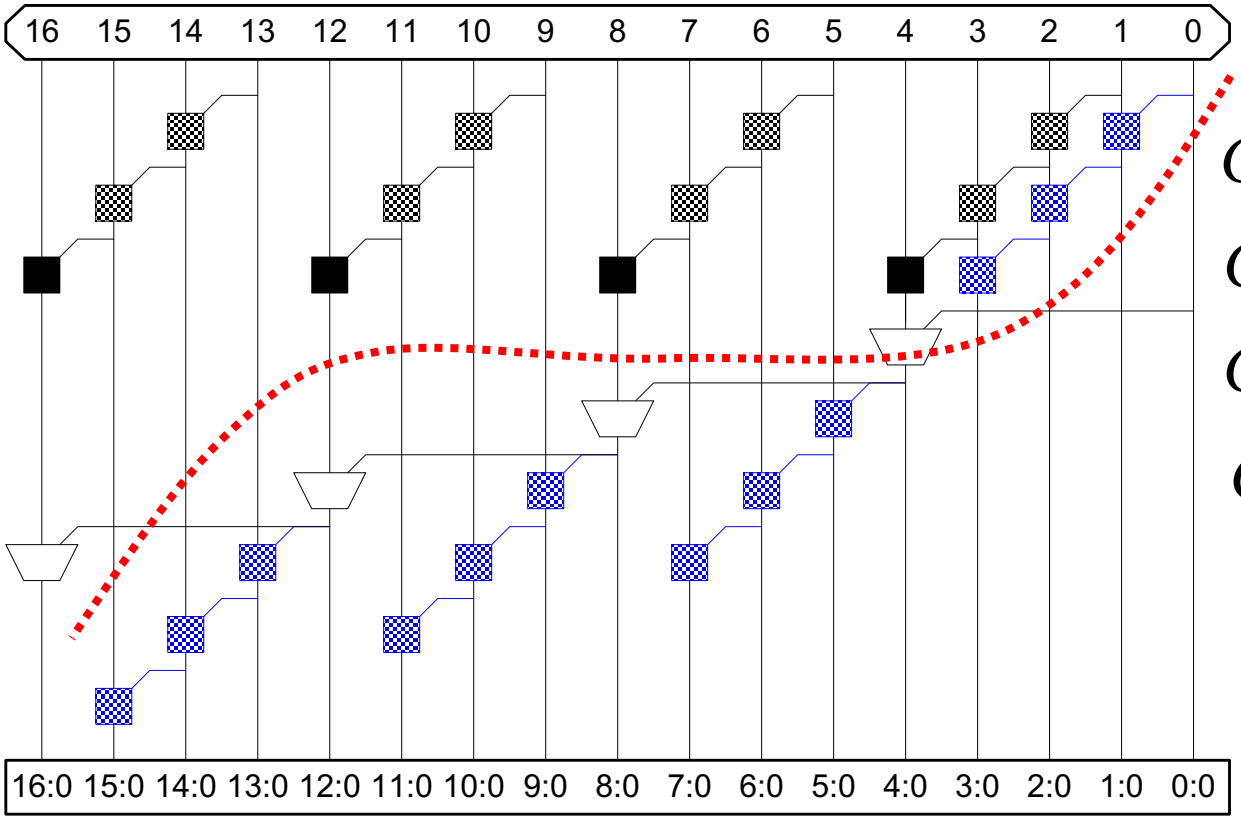
$$C_{o,4} = g_4 + p_4 C_3 = g_4 + p_4(g_3 + p_3 C_2) = g_4 + p_4(g_3 + p_3(g_2 + p_2\,(g_1 + p_1 C_{i,0}))) =$$

Independent with carry in *bypass*

$$g_4 + p_4 g_3 + p_4 p_3 g_2 + p_4 p_3 p_2 g_1 + p_4 p_3 p_2 p_1 p_0 C_{i,0}$$

$$= g_4 + p_4(g_3 + p_3(g_2 + p_2\, g_1)) + p_3 p_2 p_1 p_0 C_{i,0}$$

Carry-Skip PG Diagram



$$G_{4:0} = G_{4:1} + P_{4:1}G_{0:0}$$

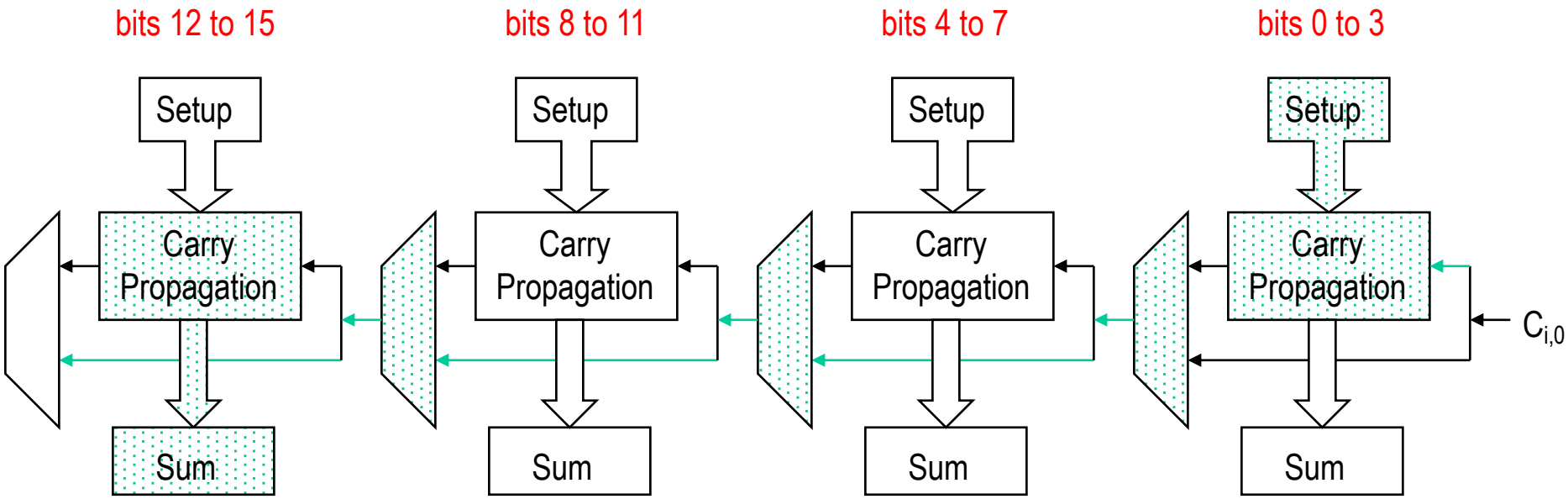
$$G_{8:0} = G_{8:5} + P_{8:5}G_{4:0}$$

$$G_{12:0} = G_{12:0} + P_{12:9}G_{8:0}$$

$$G_{16:0} = G_{16:13} + P_{16:13}G_{12:0}$$

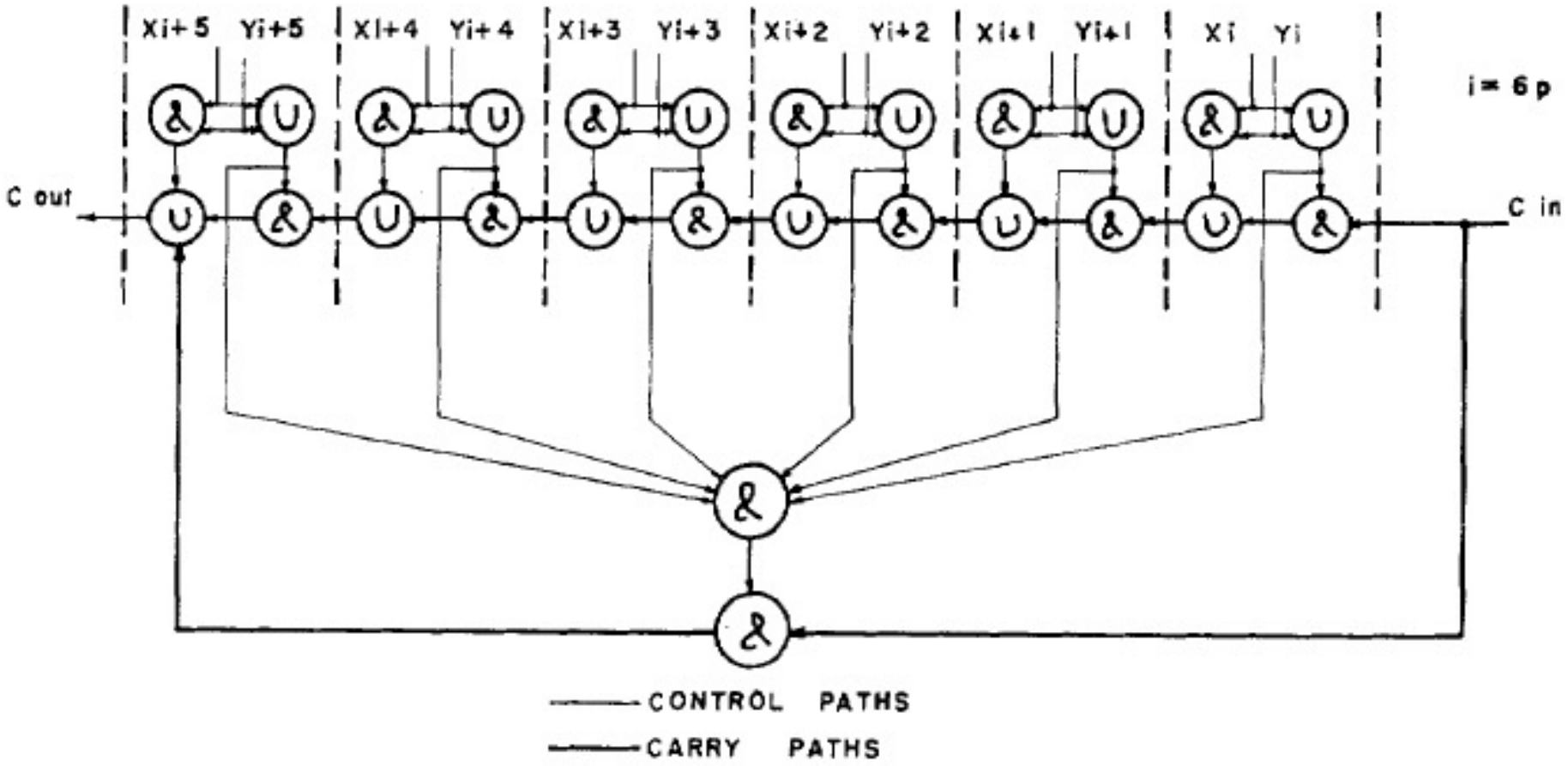
$$t_{\text{skip}} = t_{pg} + \left[2(n-1) + (k-1)\right] t_{AO} + t_{\text{xor}}$$

4/16 bit Block Carry Skip Adder



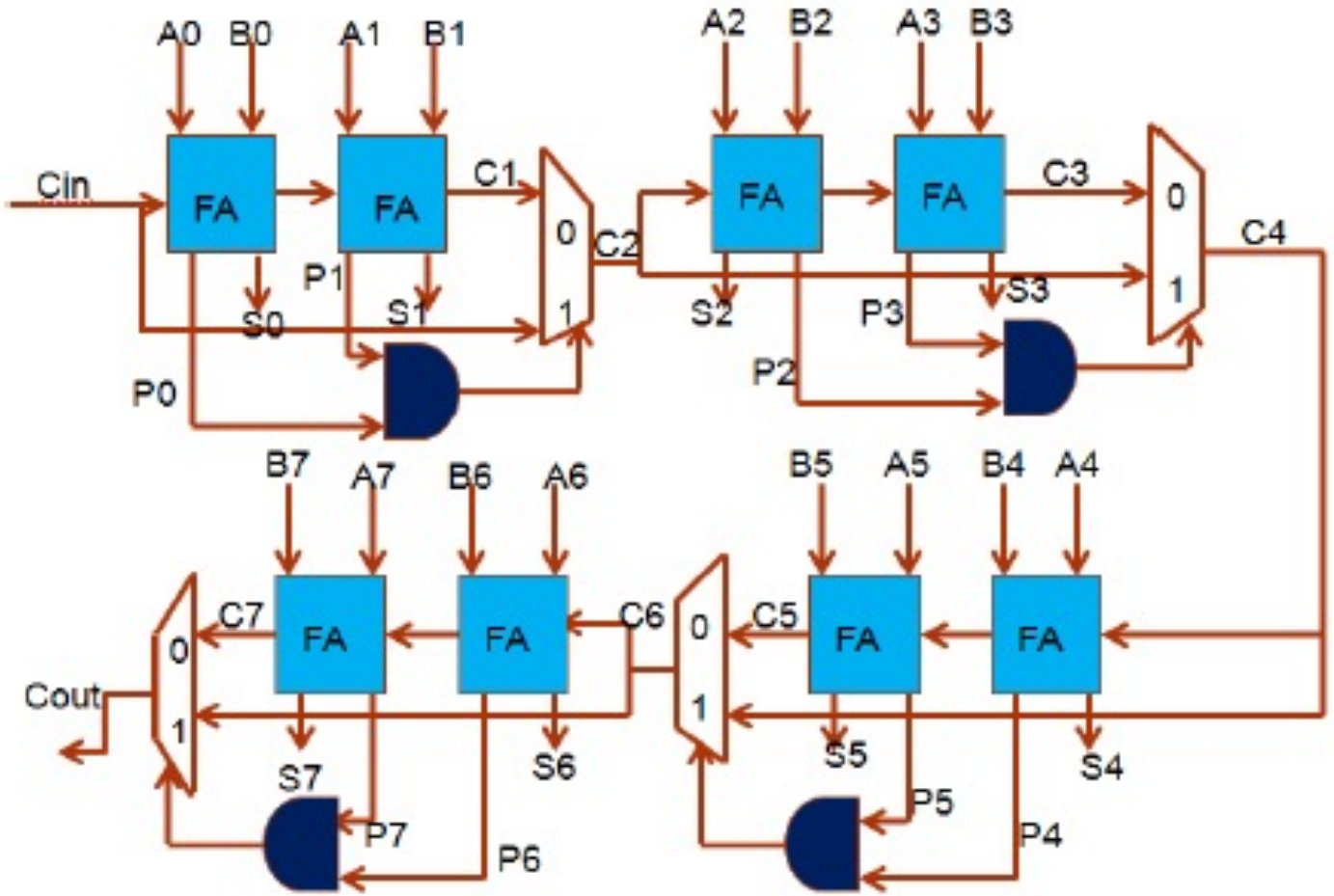
Worst-case delay → carry from bit 0 to bit 15 = carry generated in bit 0, ripples through bits 1, 2, and 3, skips the middle two groups (B is the group size in bits), ripples in the last group from bit 12 to bit 15

Another prove



M. LEHMAN AND N. BURLA Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units IRE TRANSACTIONS ON ELECTRONIC COMPUTERS 1961

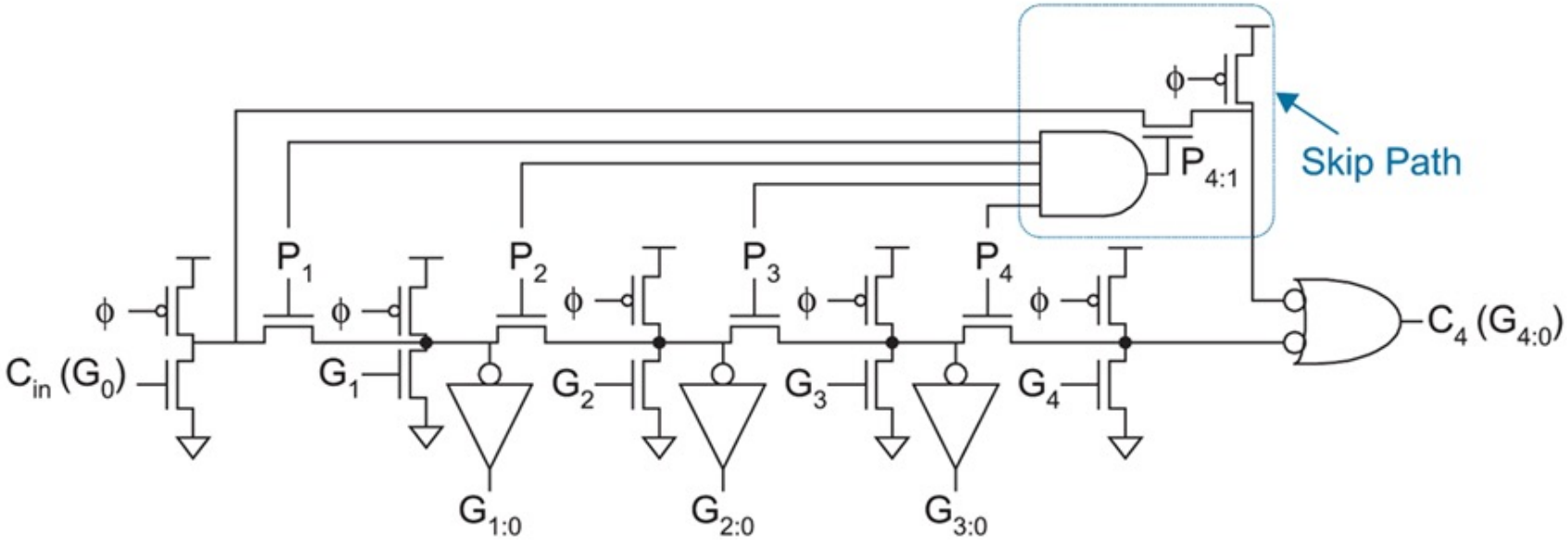
Another carry skip adder



Santanu Maity, Bishnu Prasad De, Aditya Kr. Singh ,*Design and Implementation of Low-Power High-Performance Carry Skip Adder*, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-4, April 2012

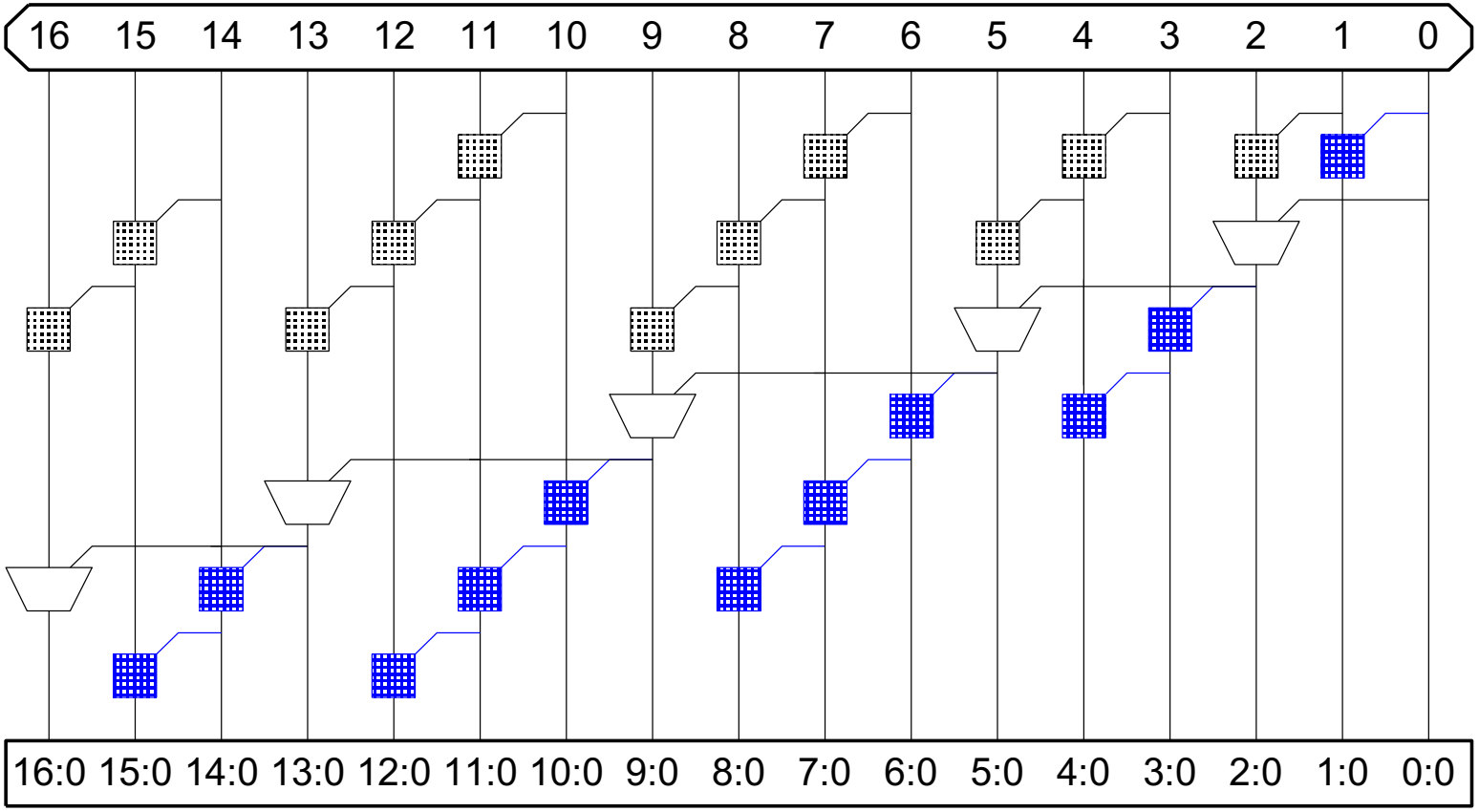
Carry-Skip using manchester

P.CHAN and M.Schlag,"Analysis and design of CMOS Manchester adders with variable carry-skip"
IEEE Trans.Computers,Vol.39,No.8,Aug.1990,pp.983-992



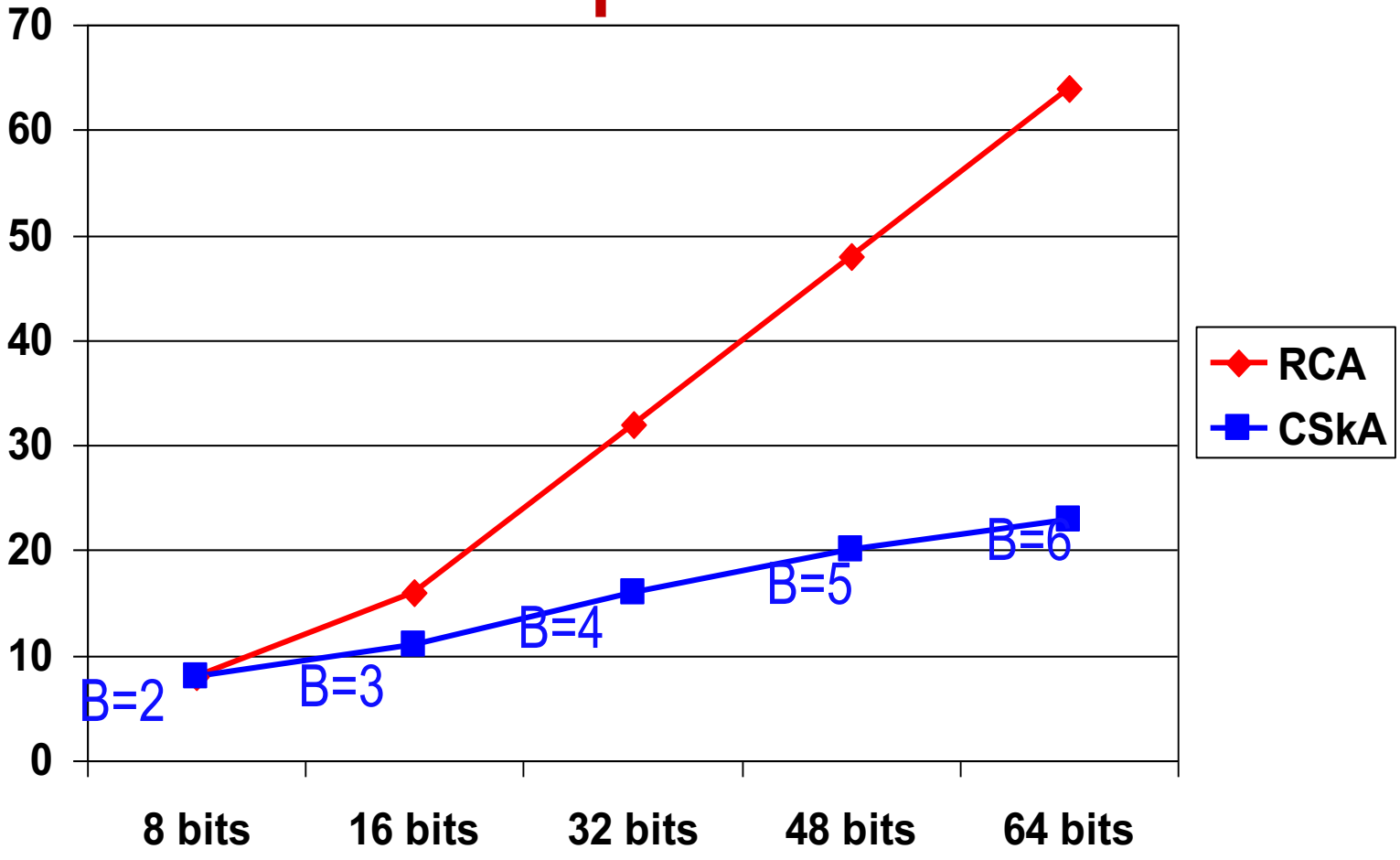
$$G_{4,0} = G_{4,1} + P_{4,1}G_{0,0}$$

Variable Group Size

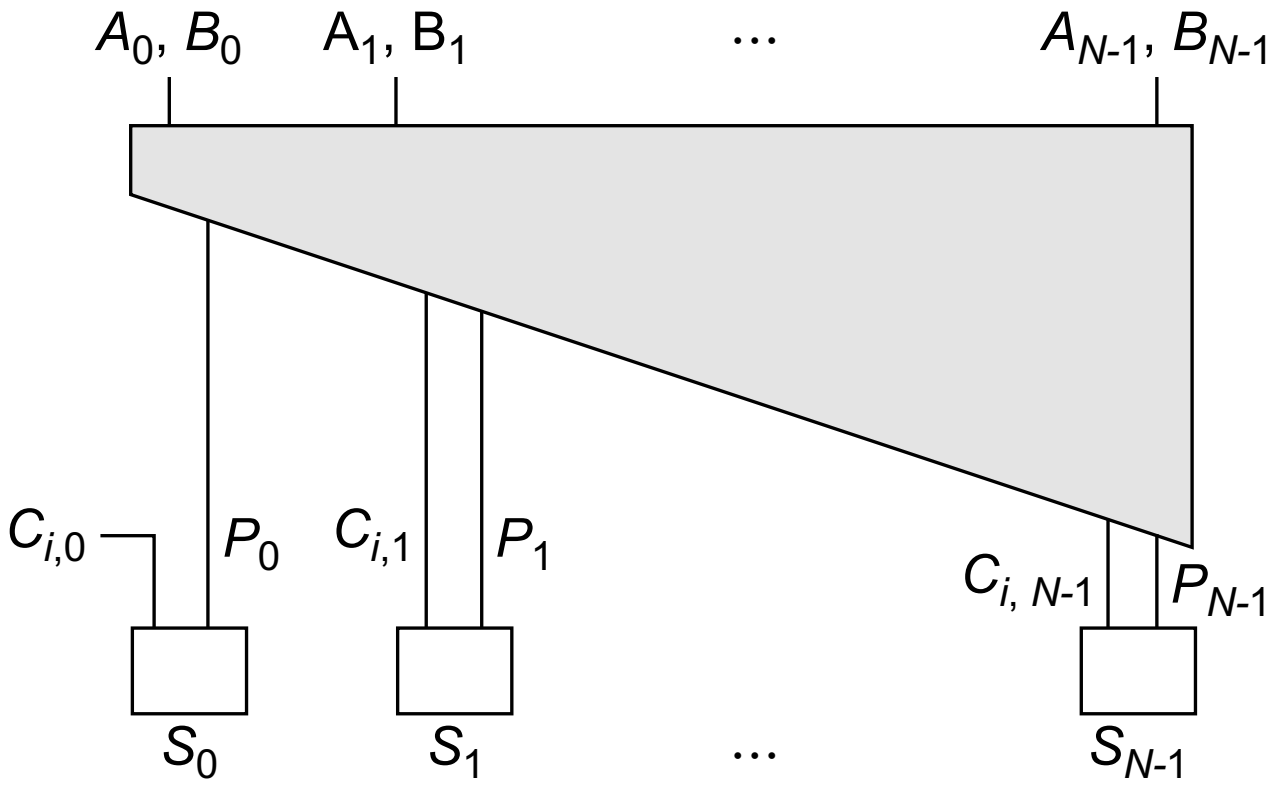


Delay grows as O(sqrt(N))

RCA, Carry Skip Adder Comparison



LookAhead - Basic Idea



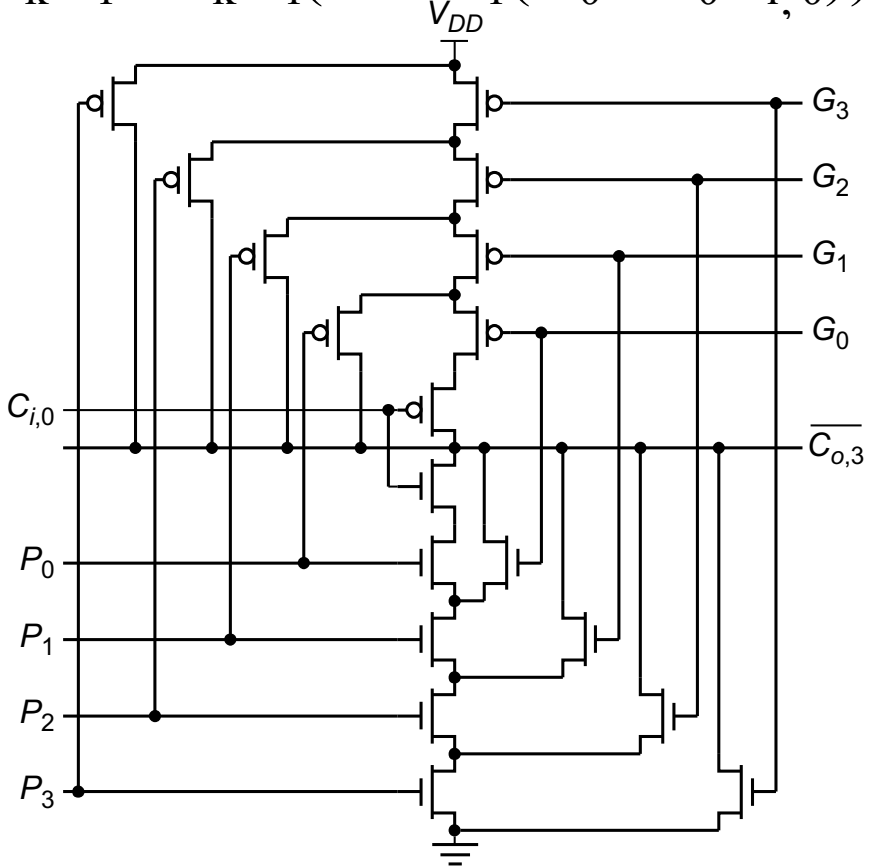
$$C_{o,\,k} = f(A_k, B_k, C_{o,\,k-1}) = G_k + P_k C_{o,\,k-1}$$

Look-Ahead: Topology

Expanding Lookahead equations:

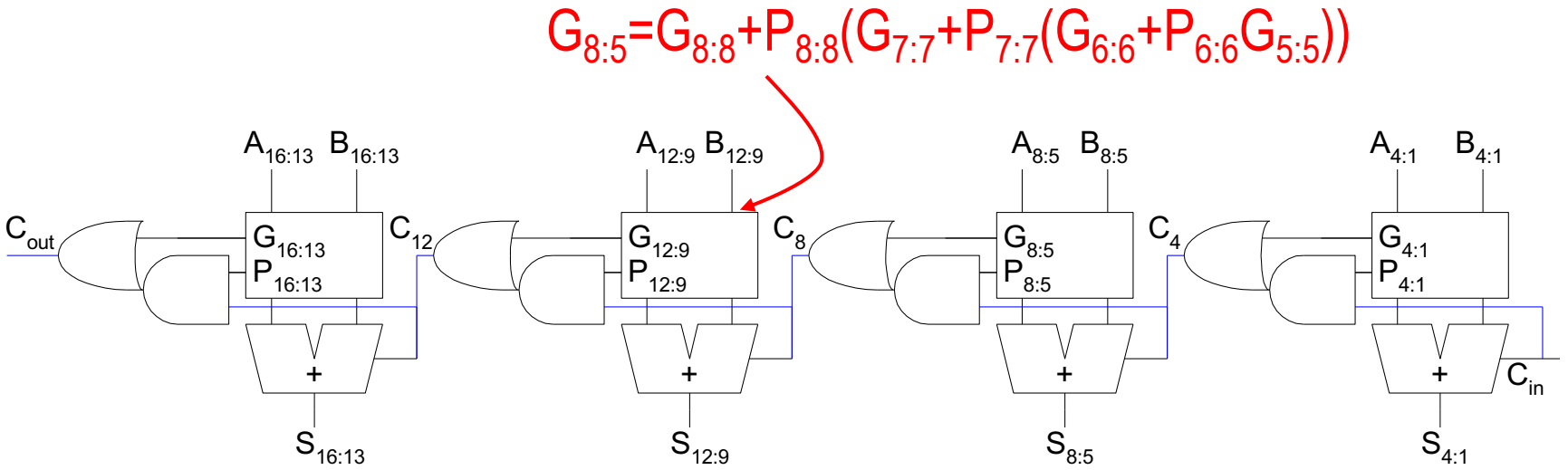
$$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}C_{o,k-2})$$

All the way: $$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}(\ldots + P_1(G_0 + P_0 C_{i,0})))$$



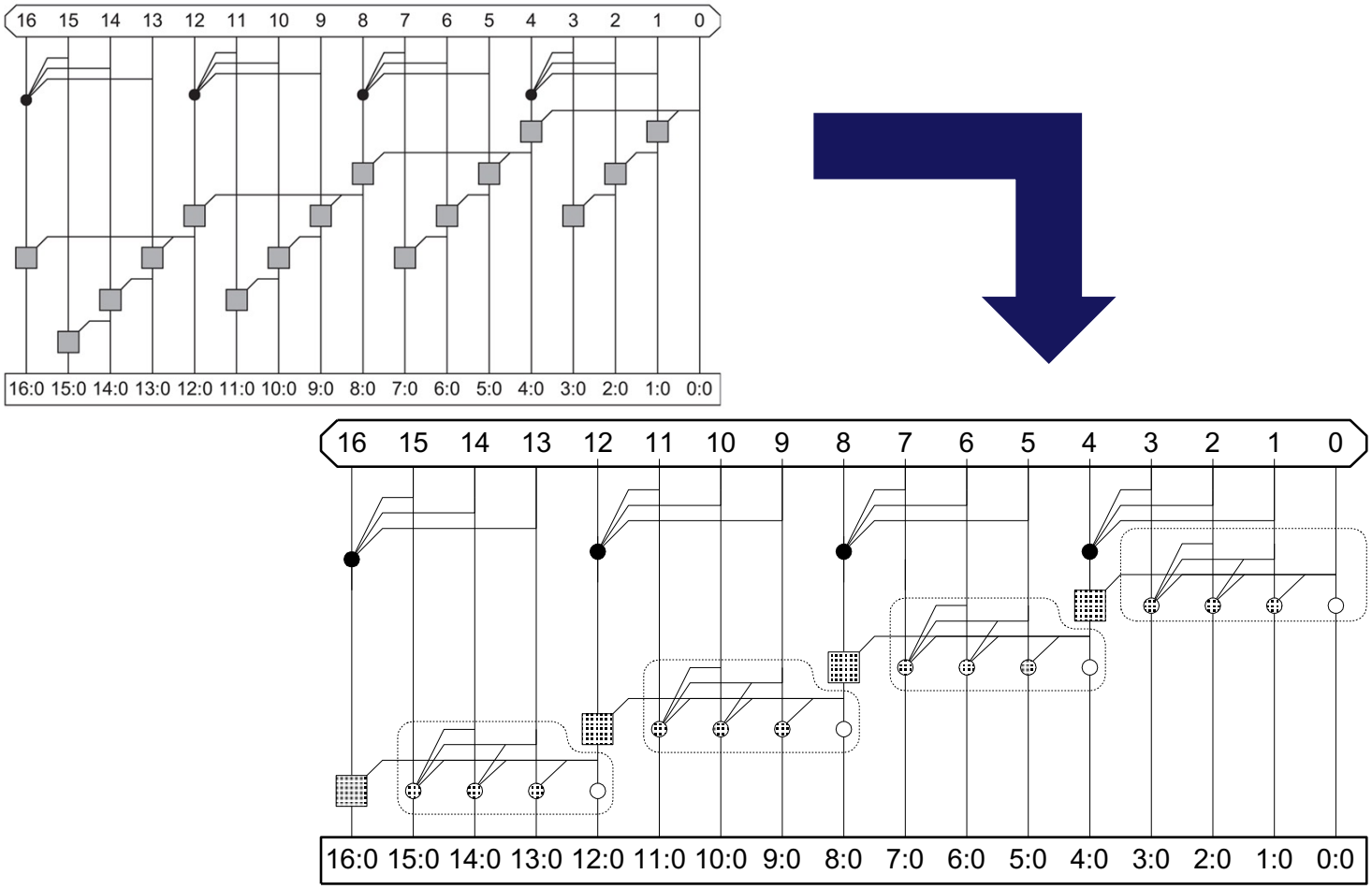
Carry-Lookahead Adder

- Carry-lookahead adder computes $G_{i:0}$ for many bits in parallel.
- Uses higher-valency cells with more than two inputs.



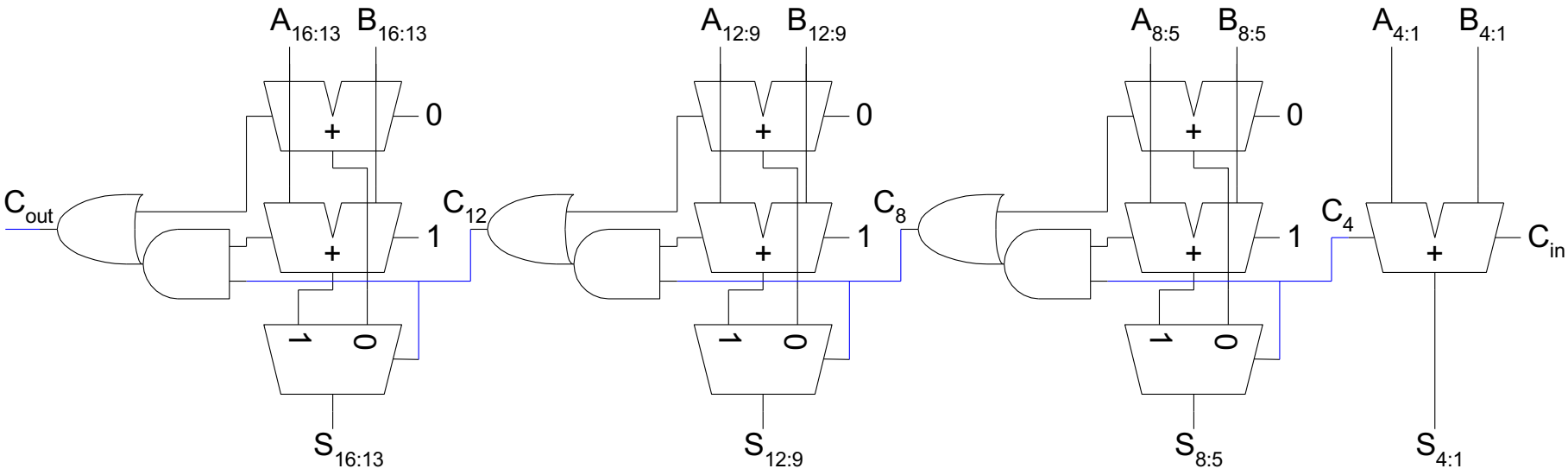
$$t_{cla} = t_{pg} + t_{pg(n)} + [(n-1)+(k-1)]t_{AO} + t_{xor}$$

CLA PG Diagram



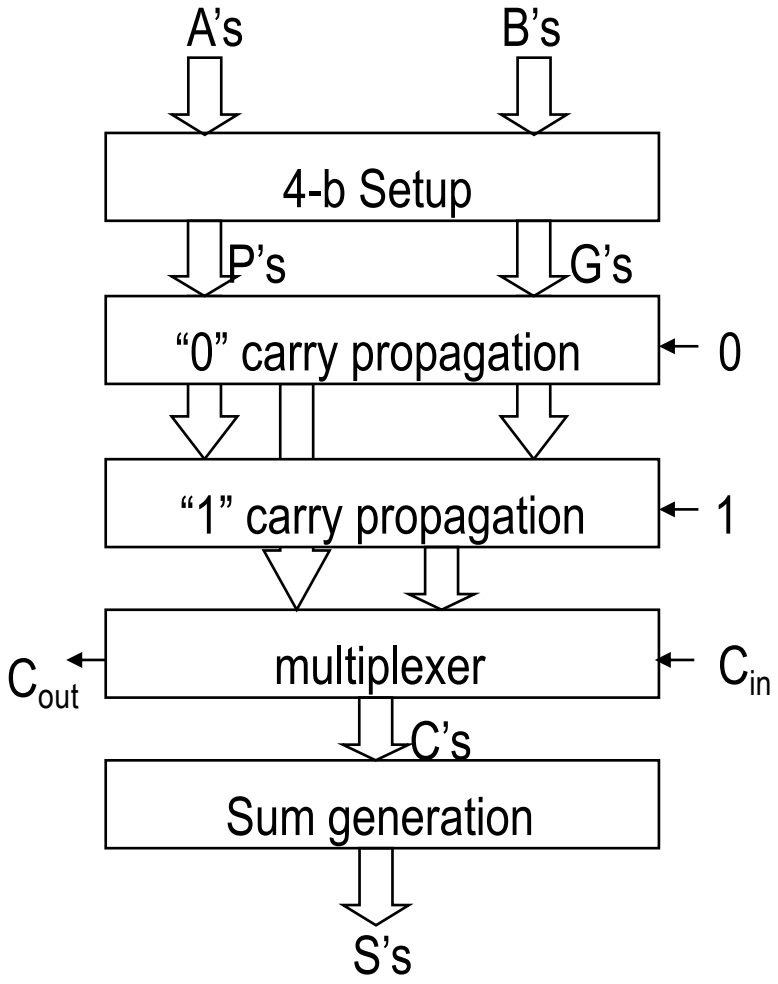
Carry-Select Adder

- Trick for critical paths dependent on late input X
  - Precompute two possible outputs for X = 0, 1
  - Select proper output when X arrives
- Carry-select adder precomputes n-bit sums
  - For both possible carries into n-bit group

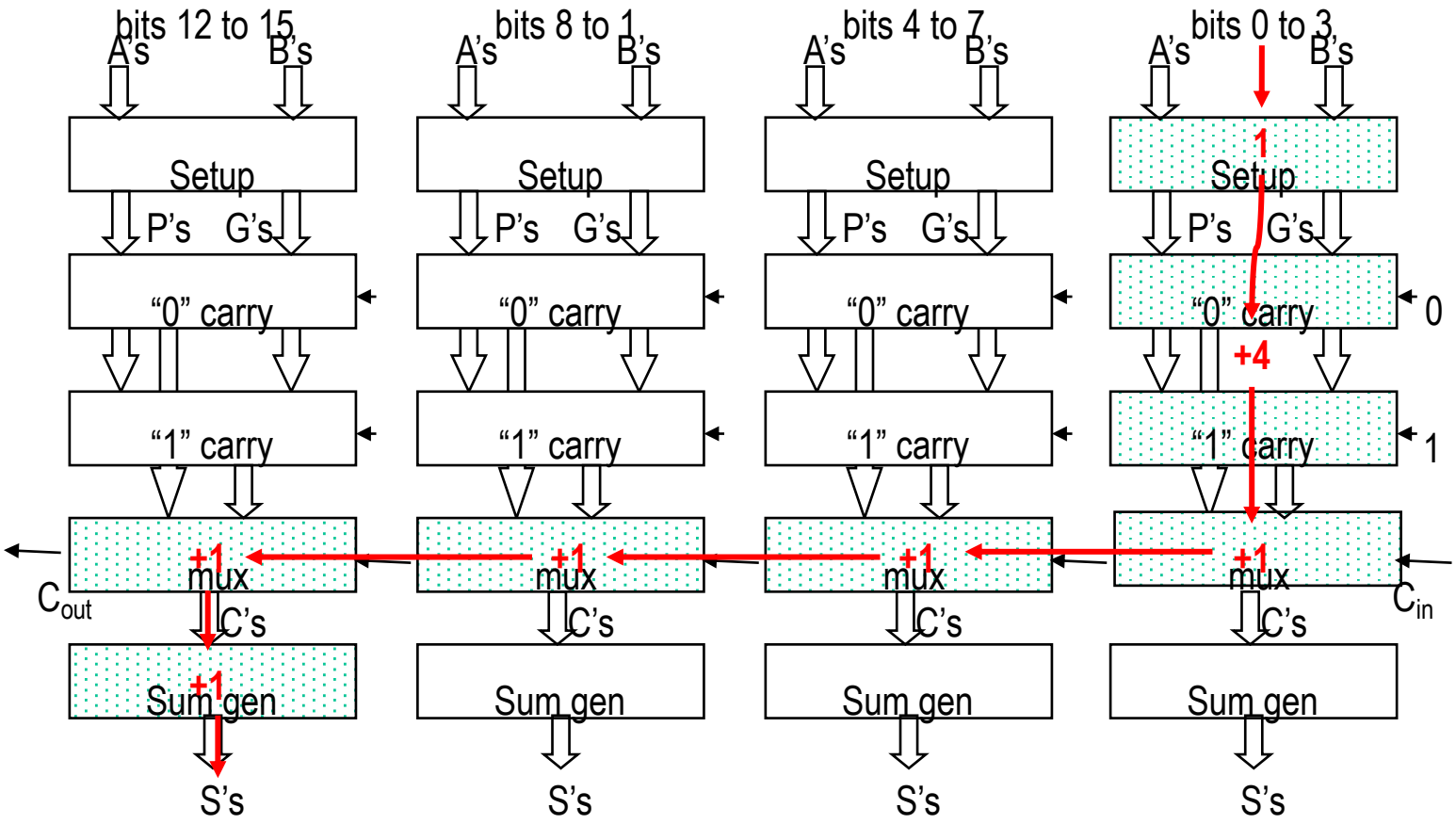


Carry Select Adder

Precompute the carry out of each block for both carry_in = 0 and carry_in = 1 (can be done for all blocks in parallel) and then select the correct one

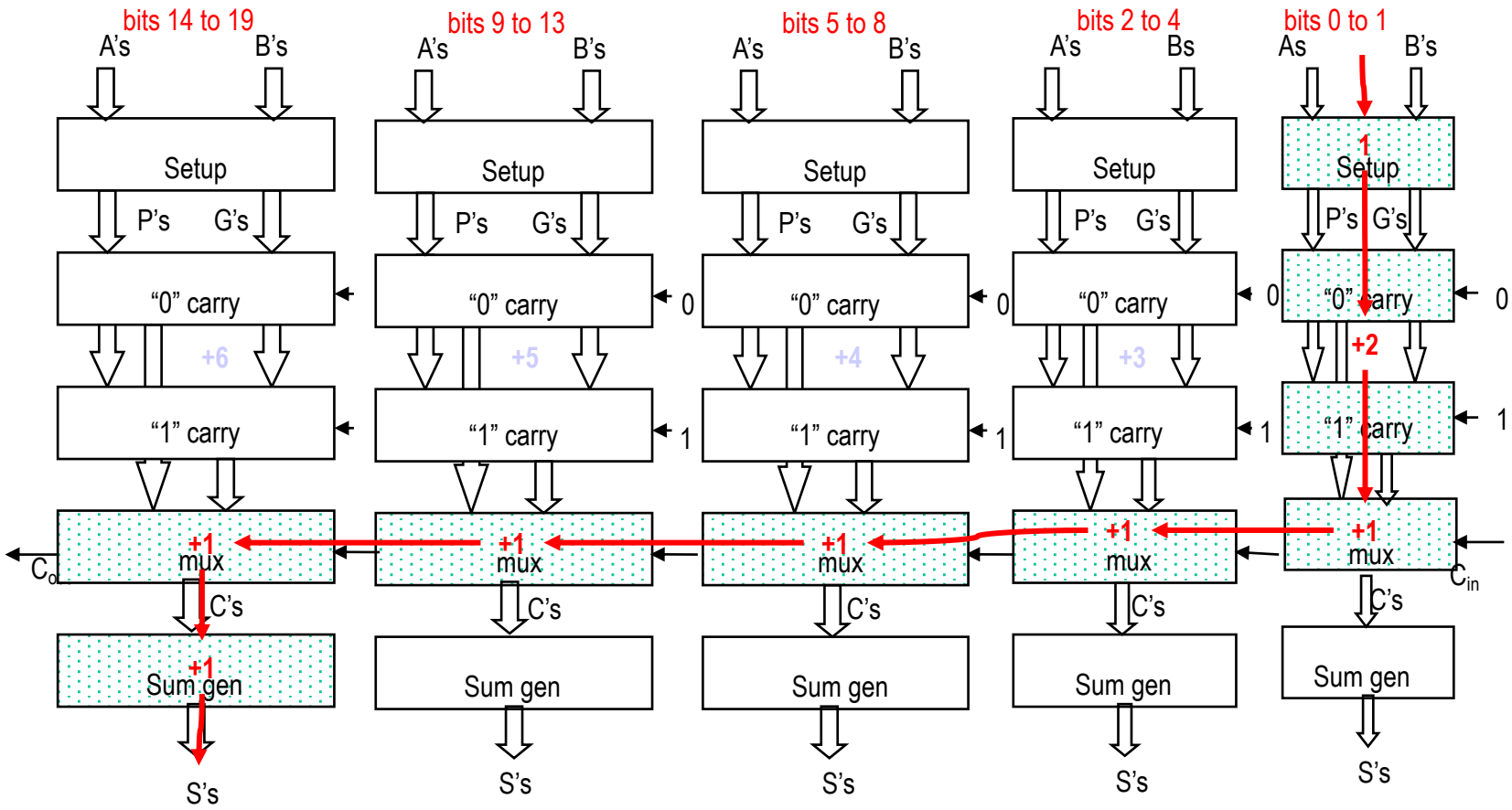


Carry Select Adder: Critical Path



$$T_{add} = t_{setup} + B\,t_{carry} + N/B\,t_{mux} + t_{sum}$$

Square Root Carry Select Adder



$$T_{add} = t_{setup} + 2\,t_{carry} + \sqrt{2N}\,t_{mux} + t_{sum}$$

Carry-select implementation

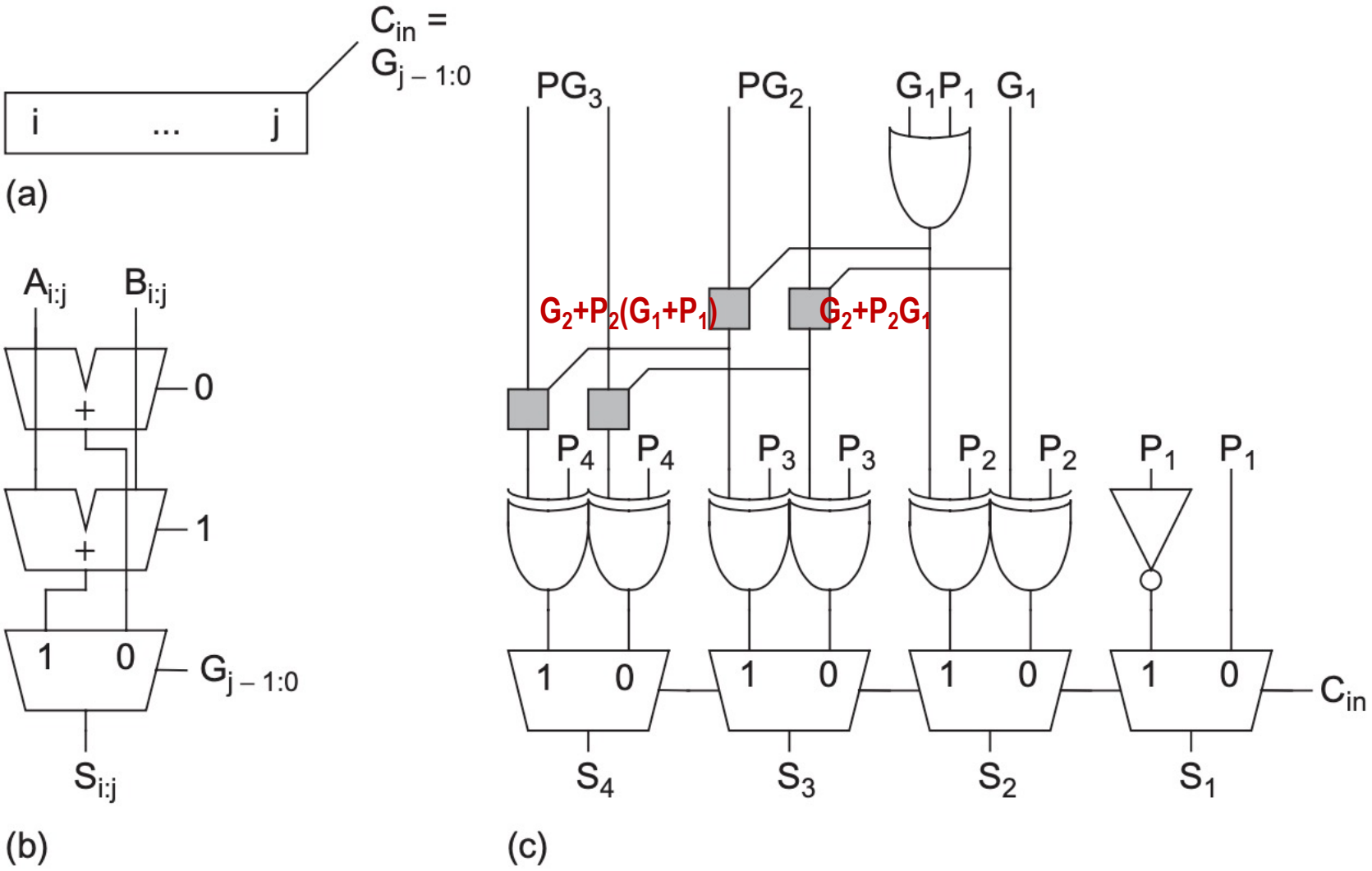
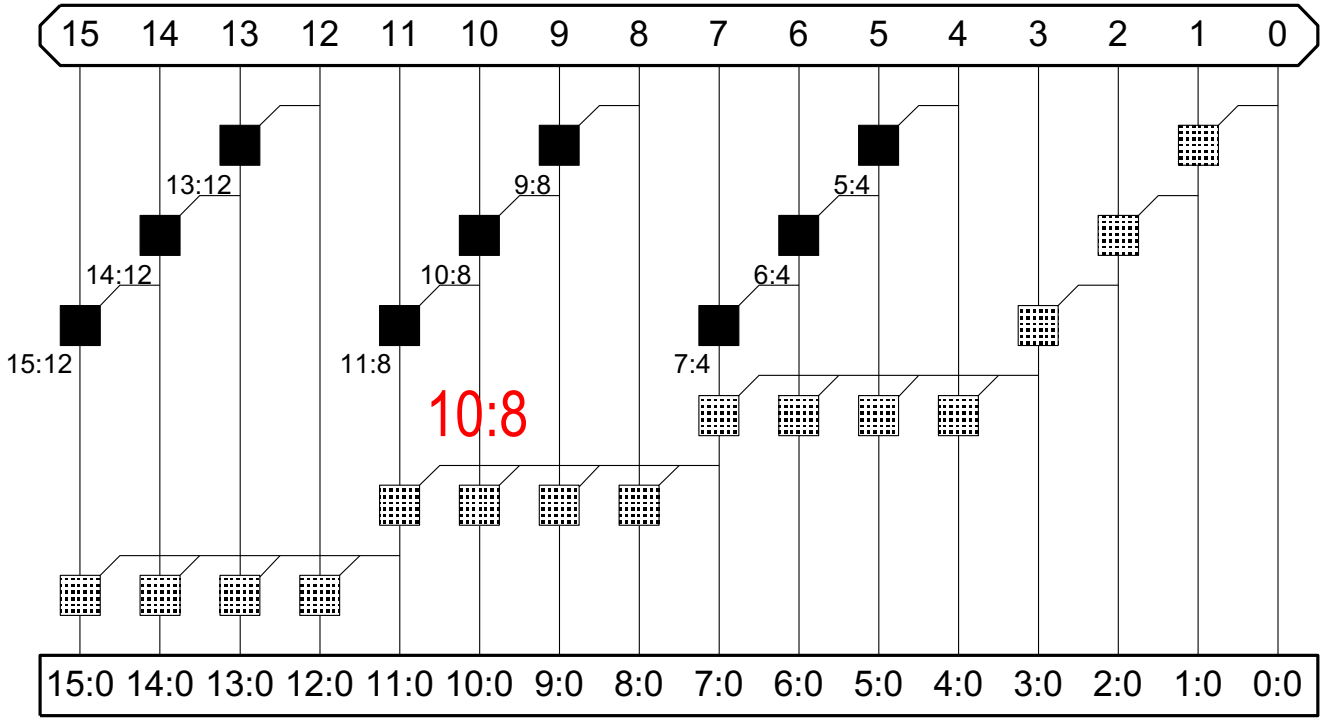


FIGURE 11.33 Carry-select implementation

Carry-Increment Adder

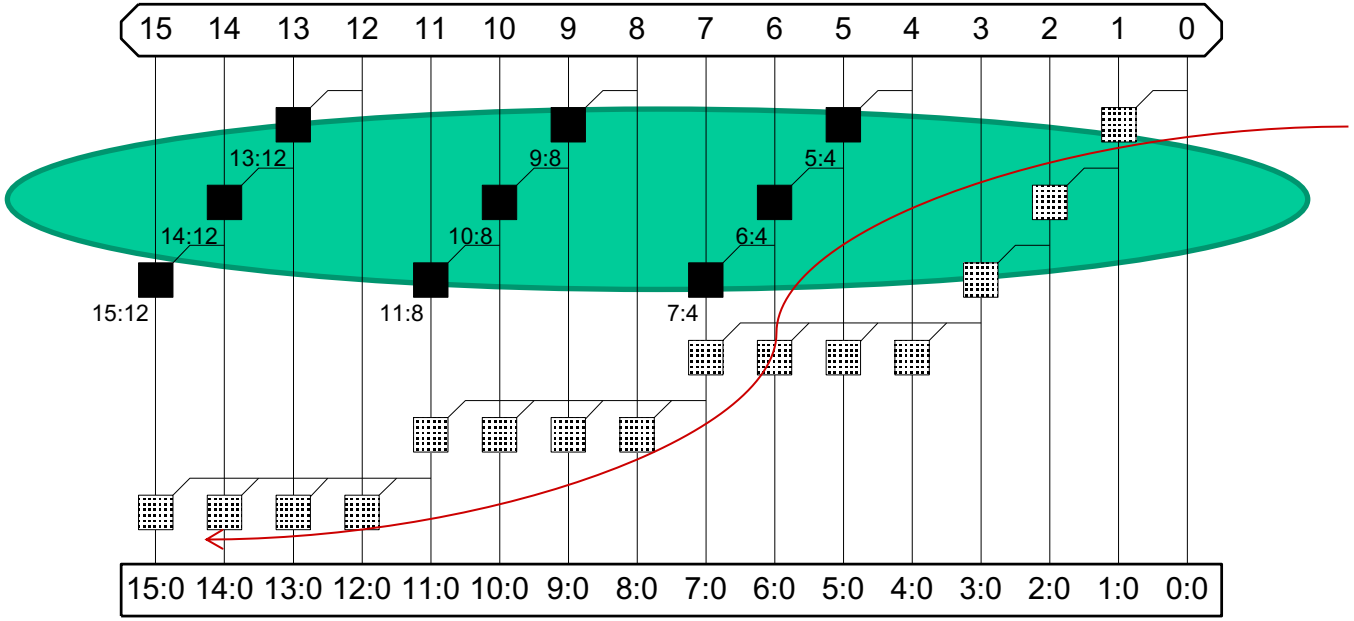
- Factor initial PG and final XOR out of carry-select



$$t_{\text{increment}} =$$

Carry-Increment Adder

- Factor initial PG and final XOR out of carry-select

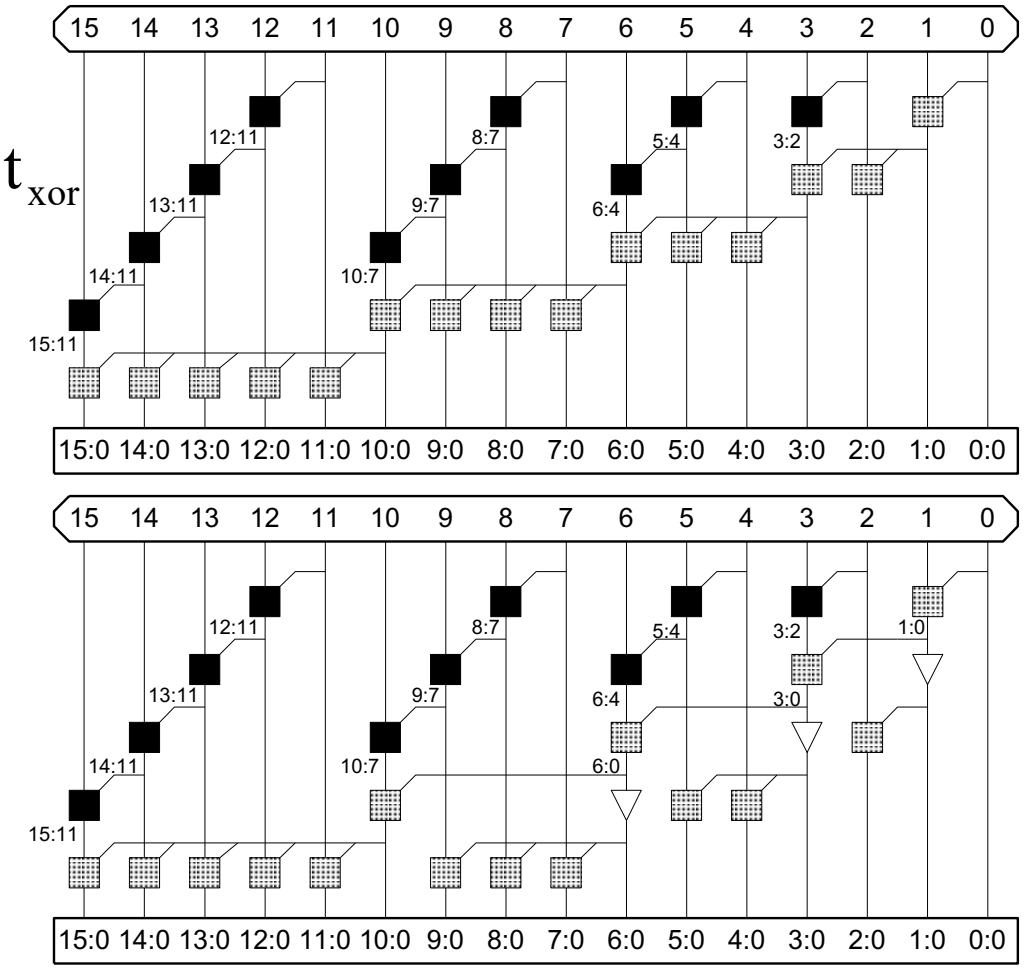


$$t_{\text{increment}} = t_{pg} + \left[(n-1) + (k-1)\right] t_{AO} + t_{\text{xor}}$$

$$t_{\text{increment}} = t_{pg} + [t_{pg(n)} + (k-1)] t_{AO} + t_{\text{xor}}$$

Variable Group Size

$$t_{increment} = t_{pg} + \sqrt{2N}t_{AO} + t_{xor}$$



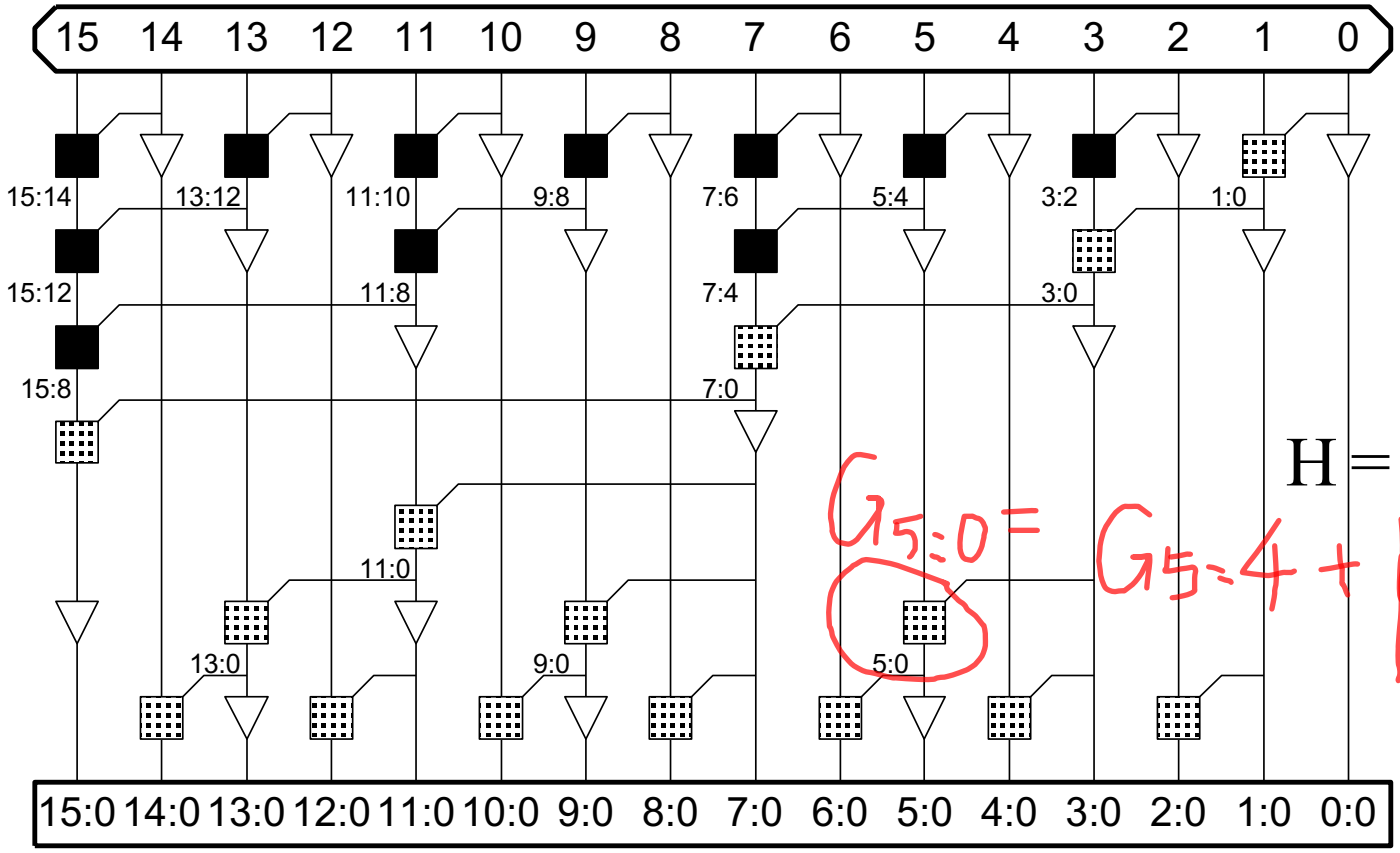
Also buffer noncritical signals

Outline

- Single-bit Addition
- Carry-Ripple Adder
- Carry-Skip Adder
- Carry-Lookahead Adder
- Carry-Select Adder
- Carry-Increment Adder
- ***Tree Adder***

Brent-Kung*(B-tree)

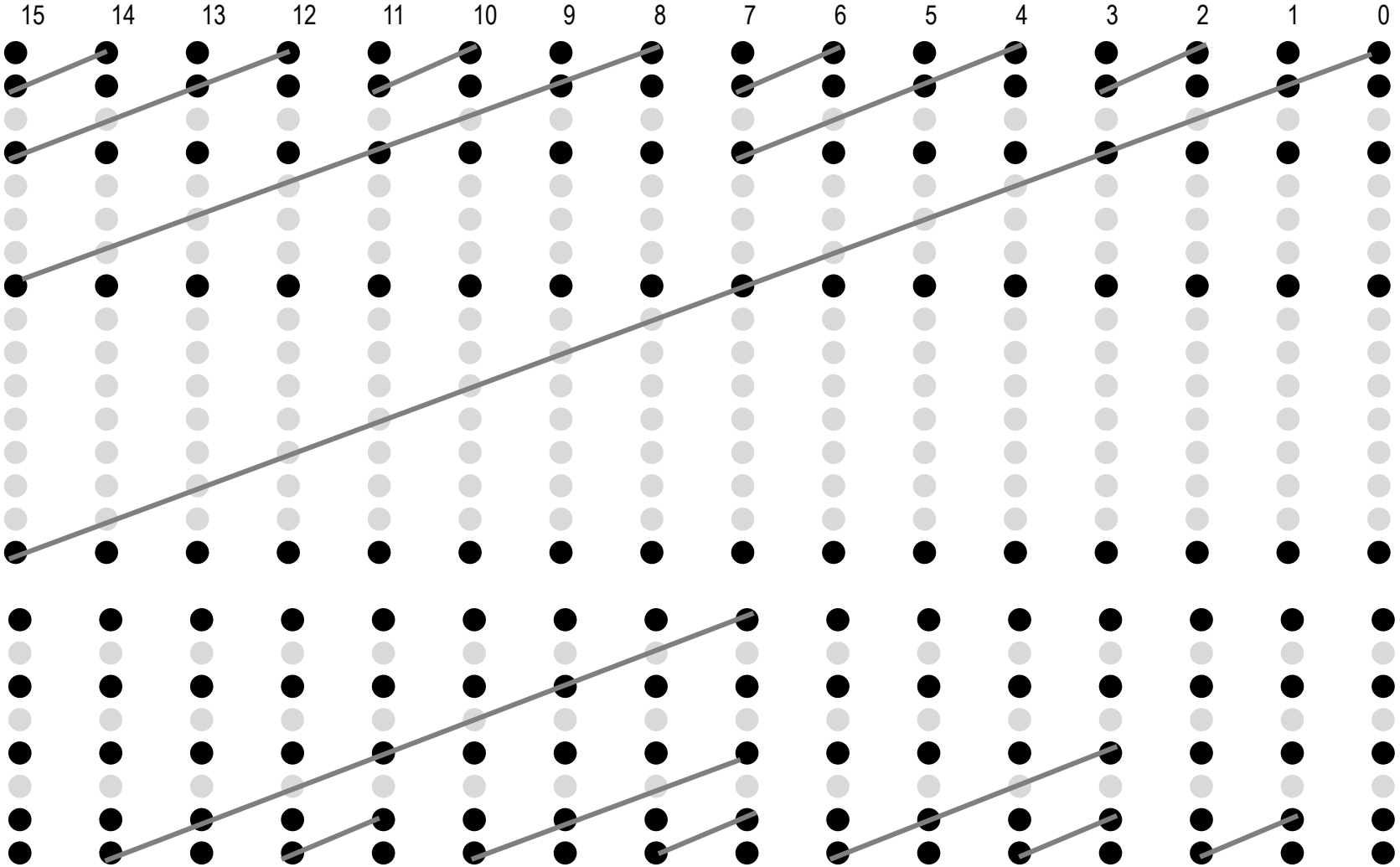
***R.Brent and H.Kung,"a regular layout for parallel adders"**
***IEEE Trans. Computer, vol. C-31,No.3,March 1982,pp.260-264**



$$H = 2\log_2 N - 1$$

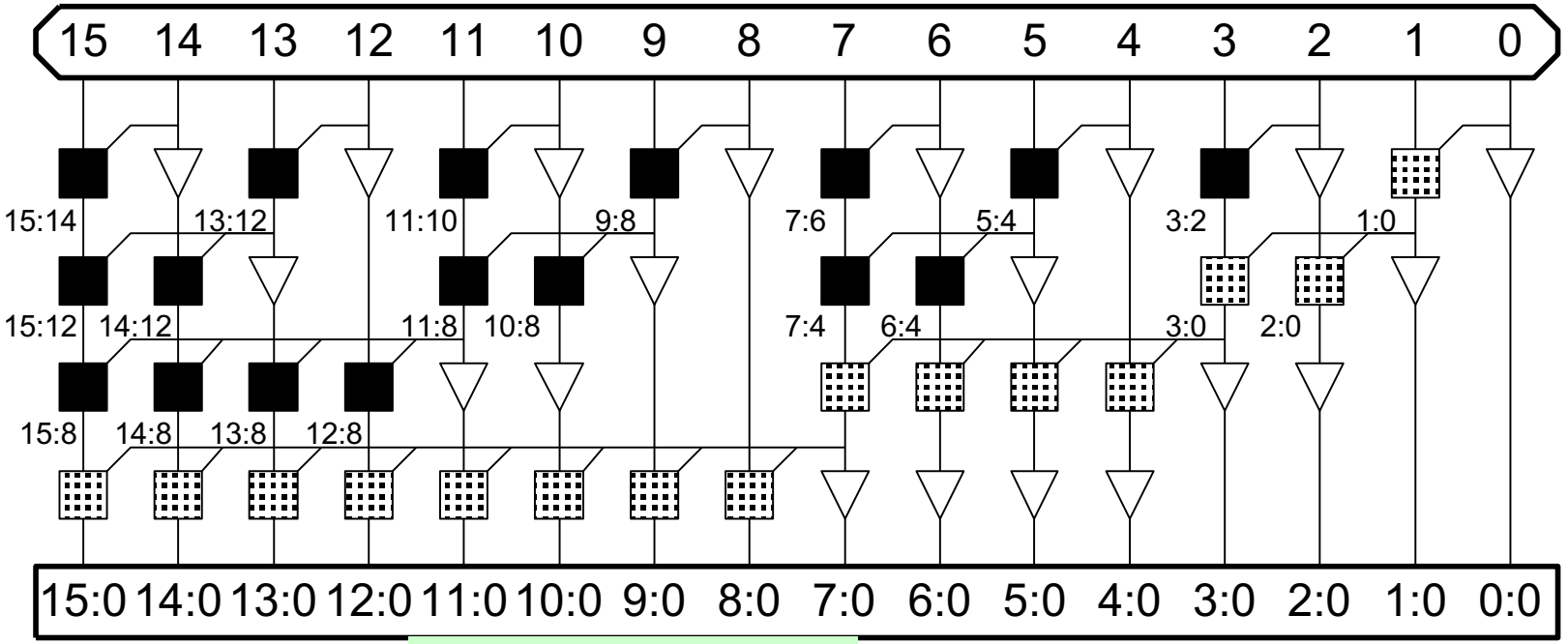
$G_{5:0} = G_{5:4} + P_{5:4} G_{3:0}$

Brent-Kung*(**B**-tree)



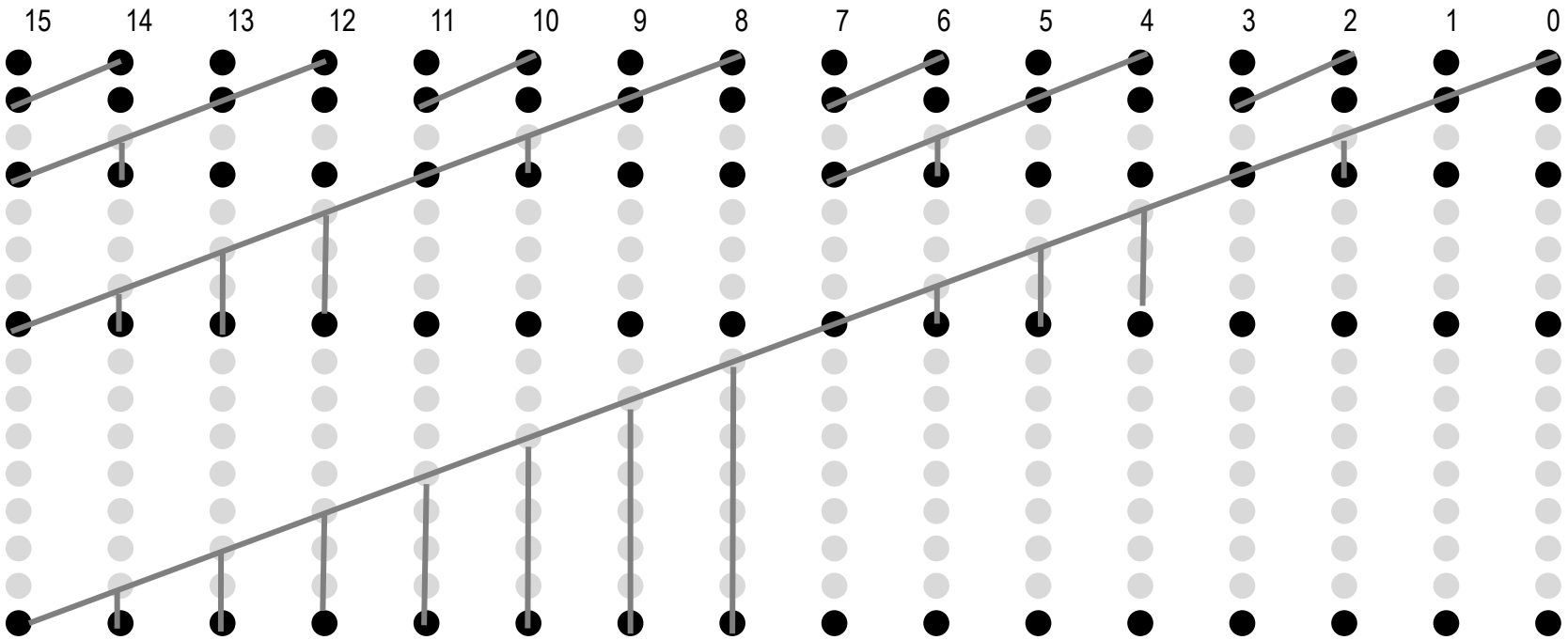
Sklansky(**S**-Tree)

J.Sklansky "conditional-sum addition logic" IER Trans. Electronic computers, vol.EC-9,June 1960,pp.226-231



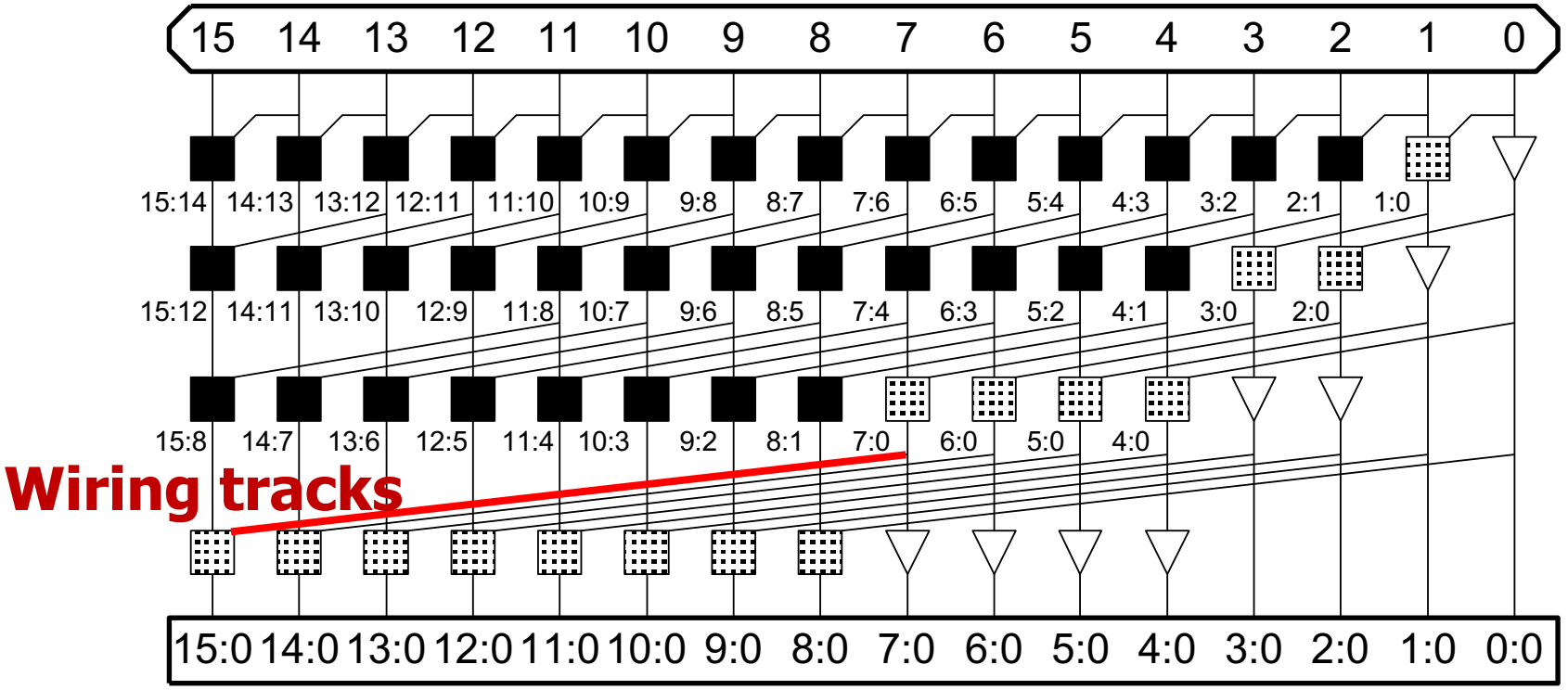
fanout = [8,4,2,1]

Sklansky(**S**-Tree)

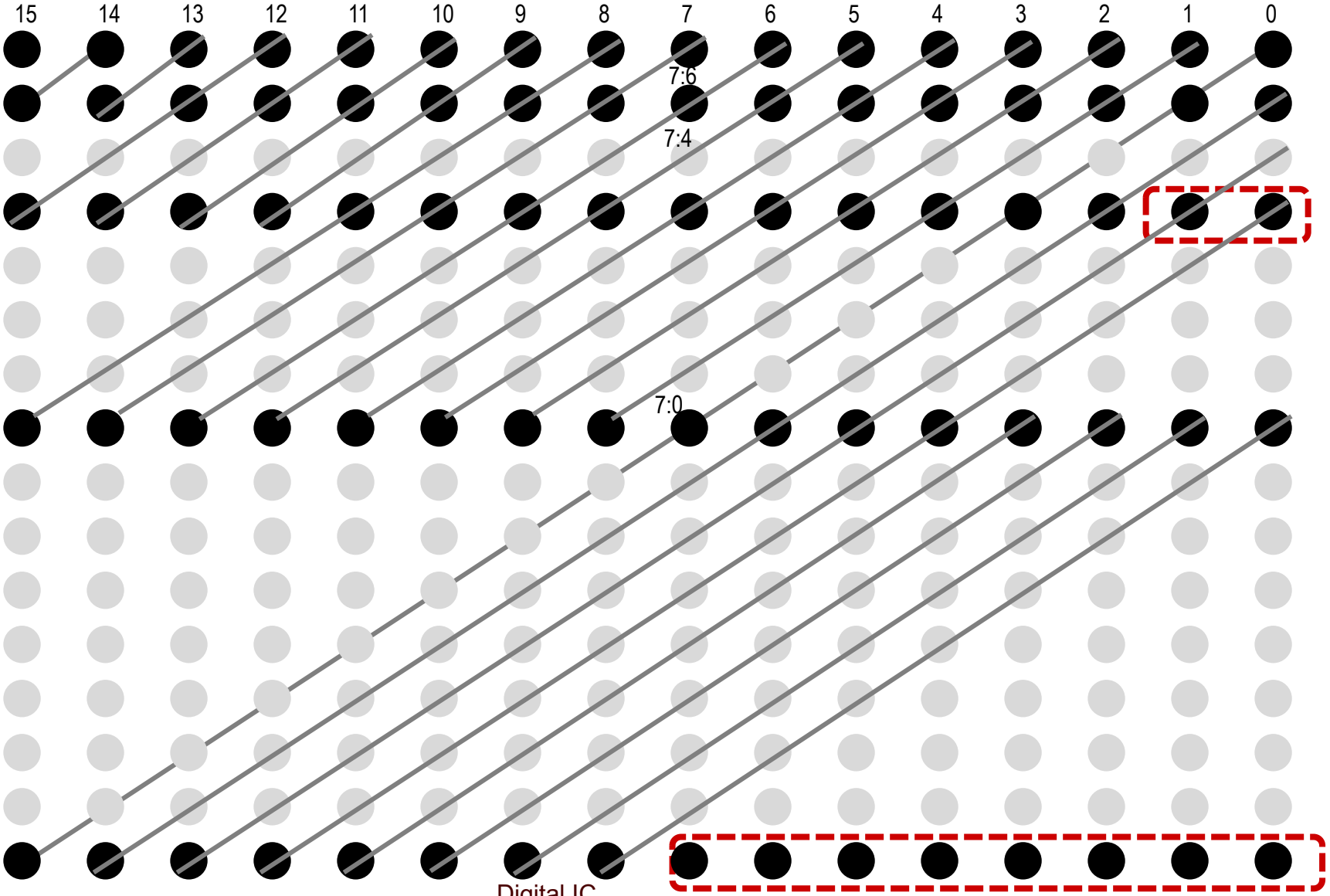


Kogge-Stone*(**K**-tree)

**P.Kogge and H.Stone,"a parallel algorithm for the efficient solution of a general class of recurrence equations" IEEE Trans. Computer, vol.C-22,No.8,Aug. 1973,pp.786-793*



16-bit Kogge-Stone prefix graph

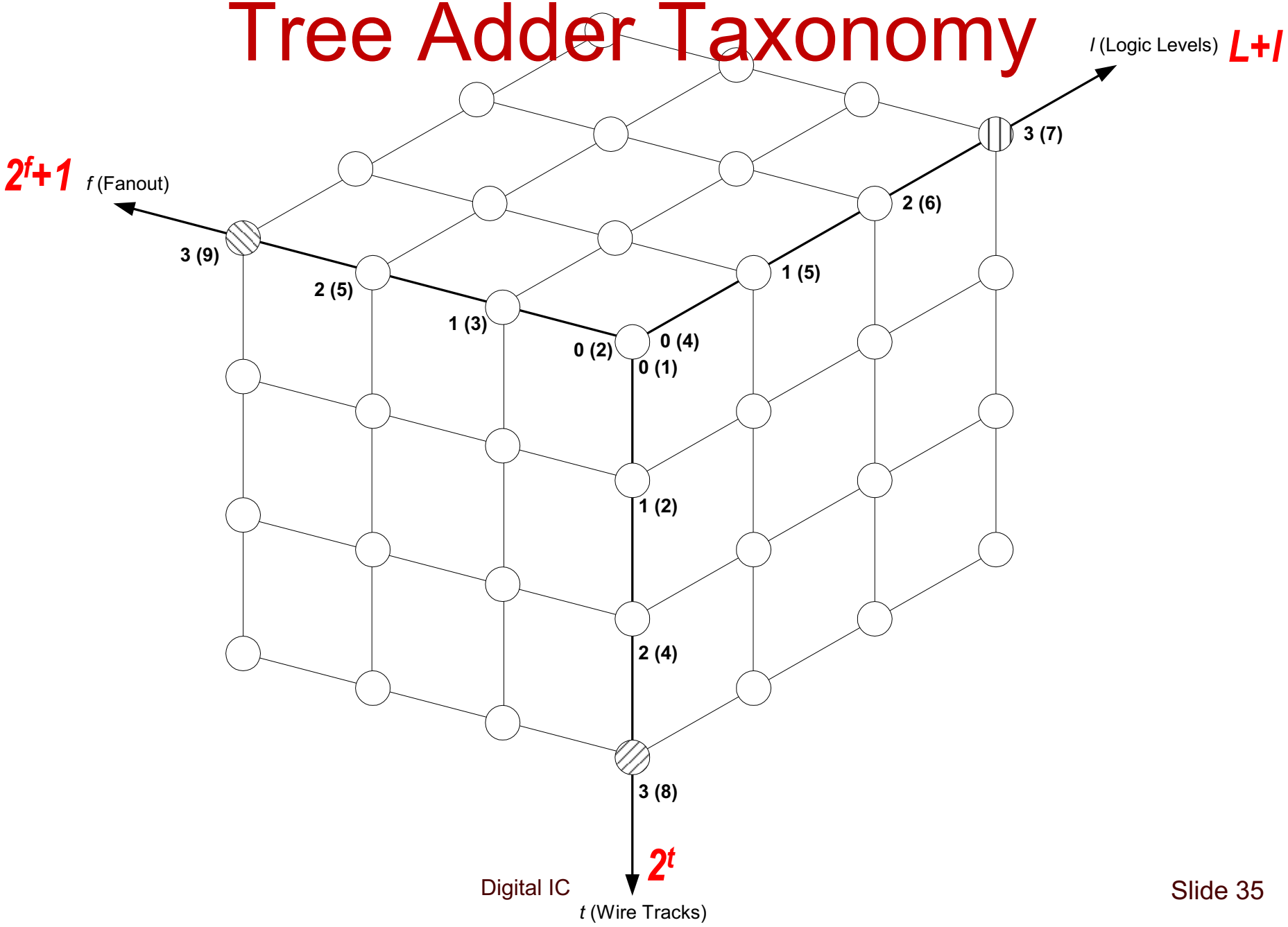


Tree Adder Taxonomy

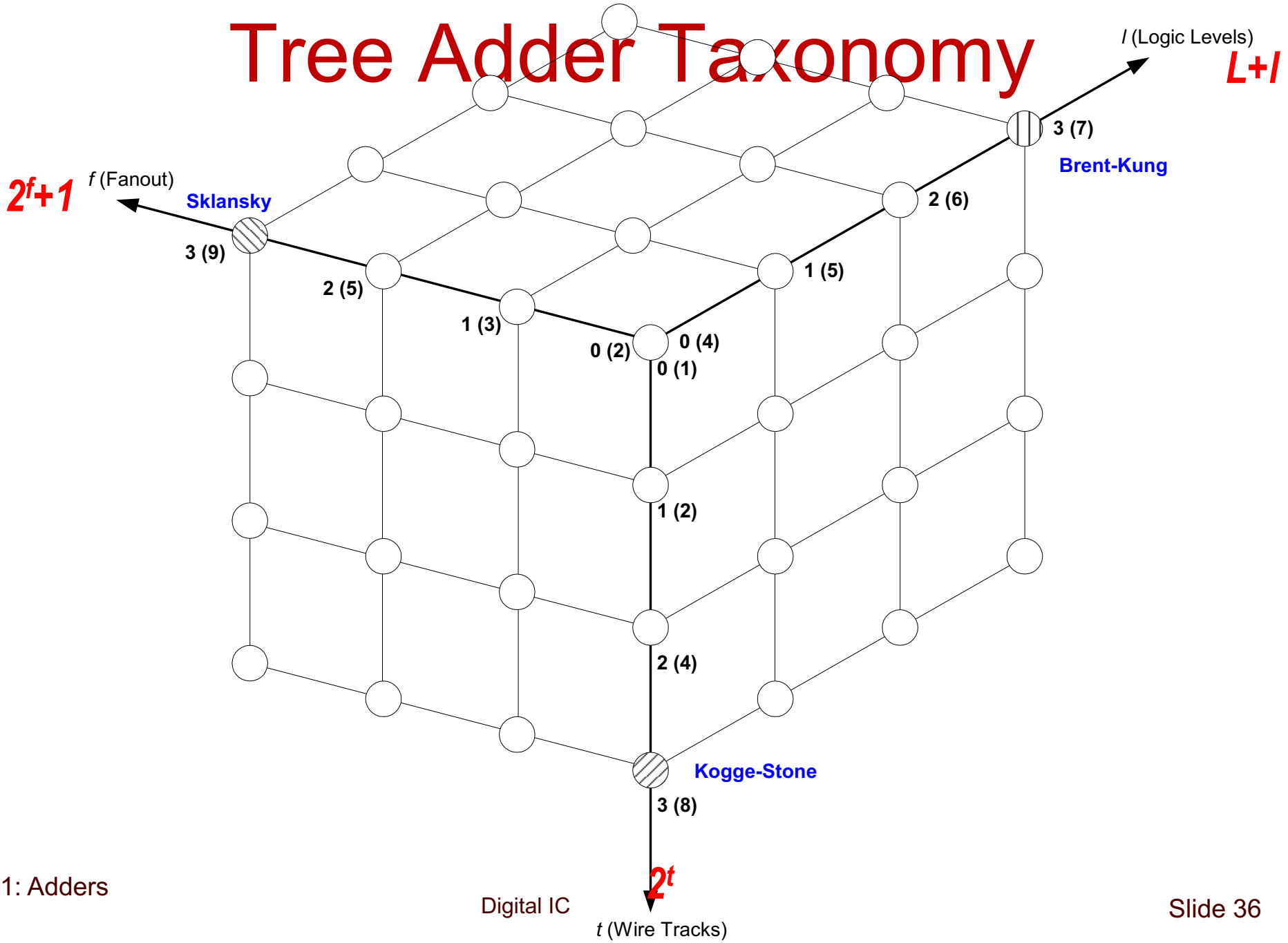
- Ideal N-bit tree adder would have
  - L = log N logic levels
  - Fanout never exceeding 2
  - No more than one wiring track between levels
- Describe adder with 3-D taxonomy (l, f, t)
  - **Logic levels:** $L + l$
  - **Fanout:** $2^f + 1$
  - **Wiring tracks:** 2^t
- Known tree adders sit on plane defined by

$$l + f + t = L\text{-}1$$

Tree Adder Taxonomy

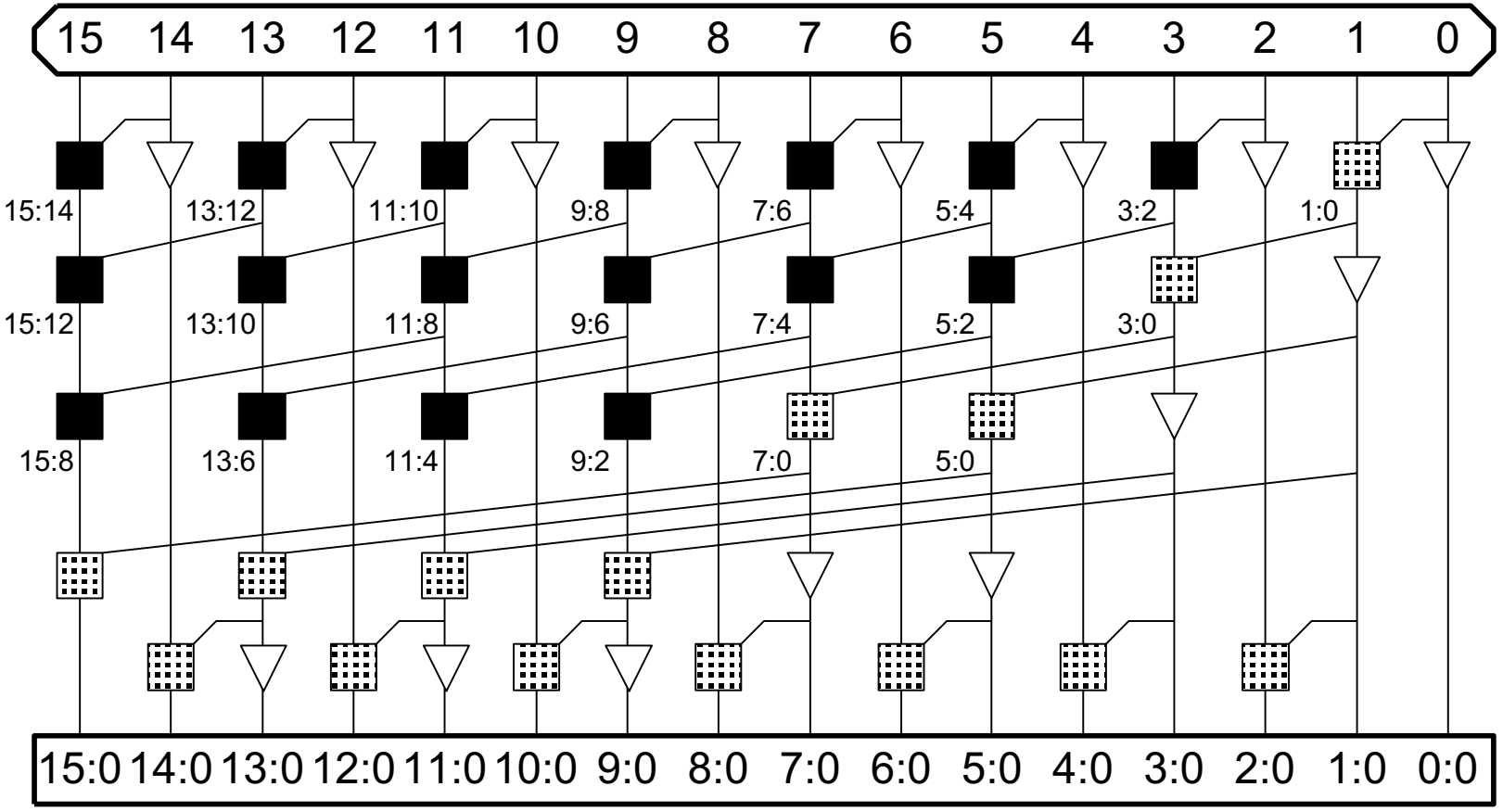


Tree Adder Taxonomy

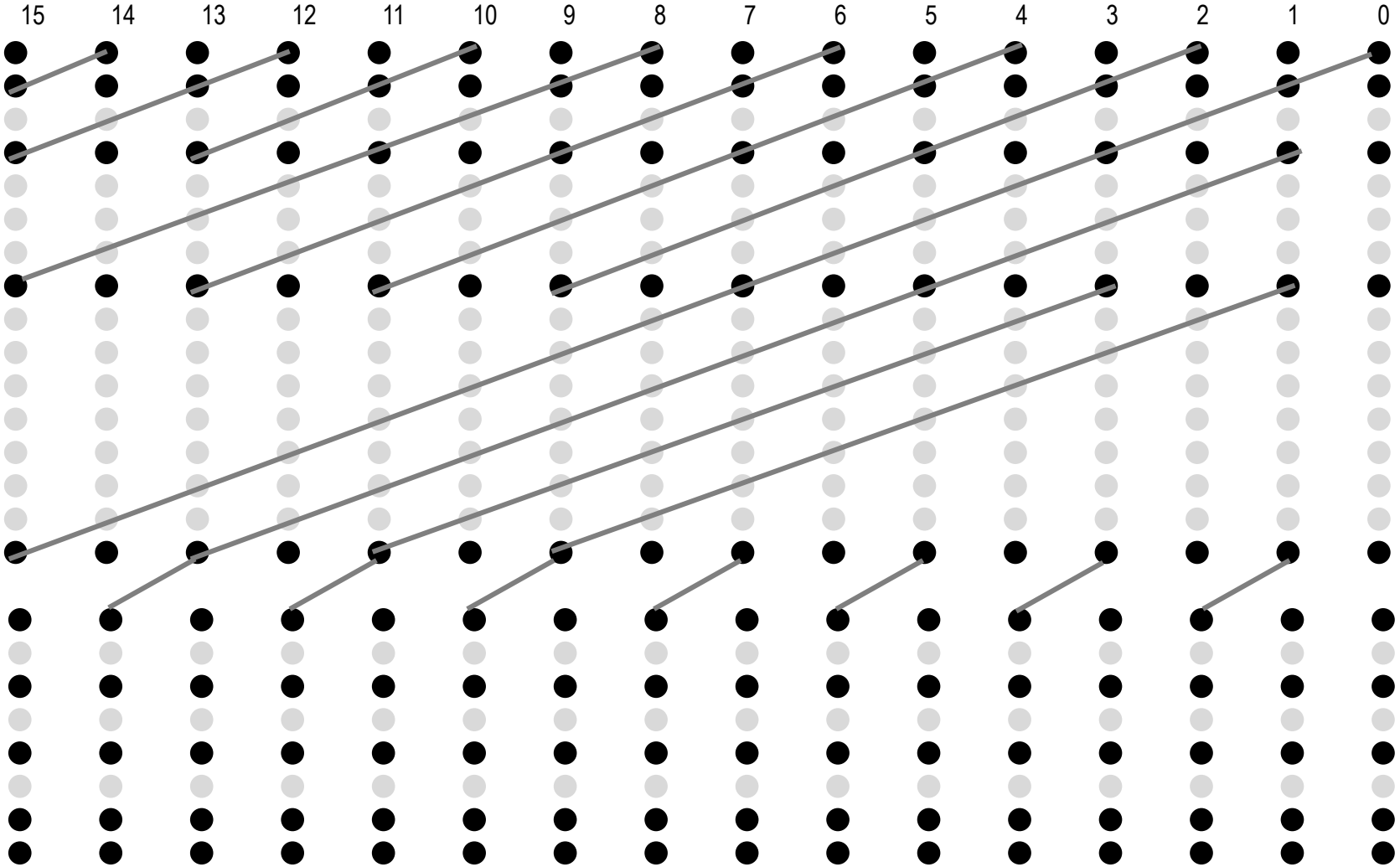


Han-Carlson(B+K tree)

Kogge Stone————Brent Kung

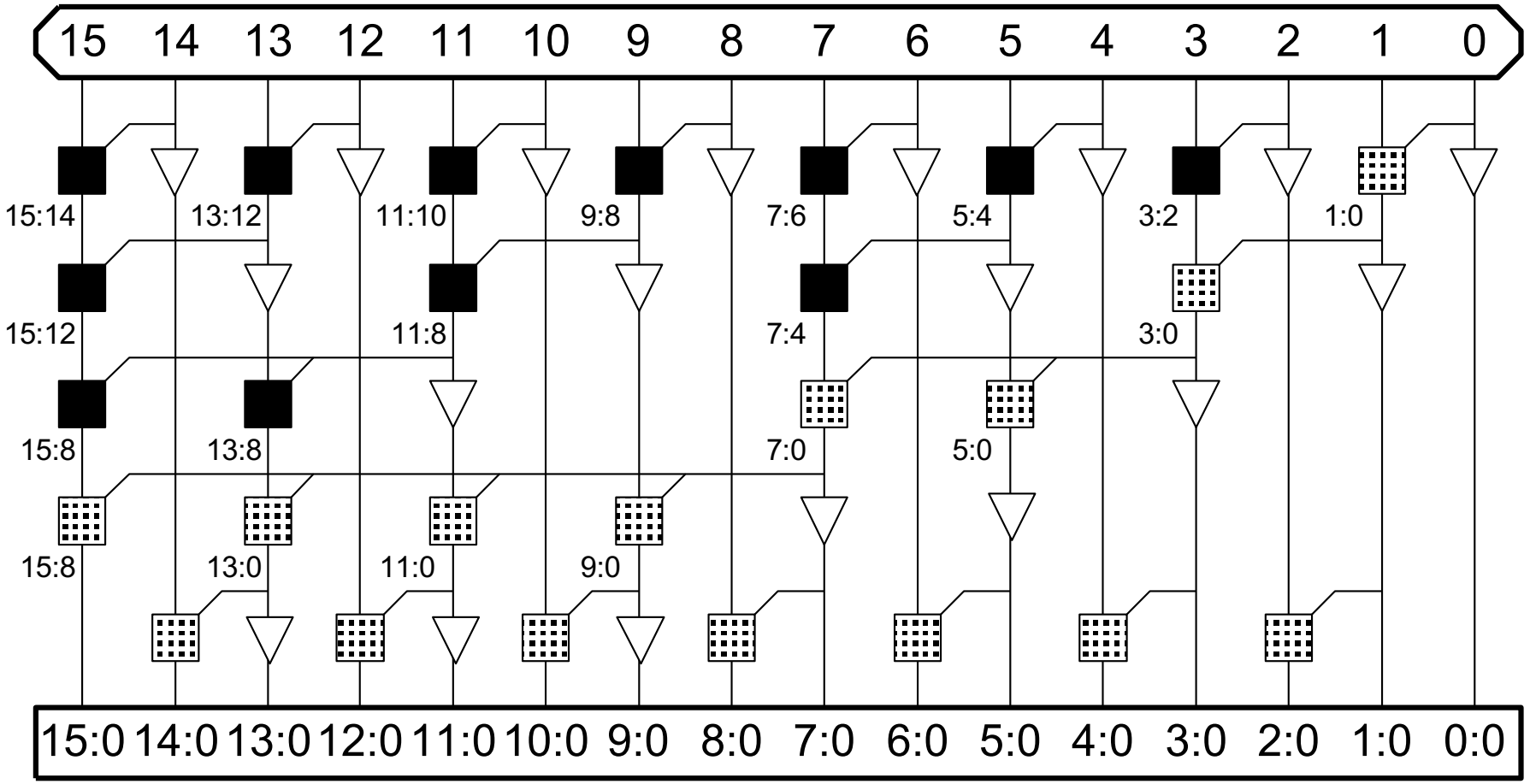


Han-Carlson(B+K tree)

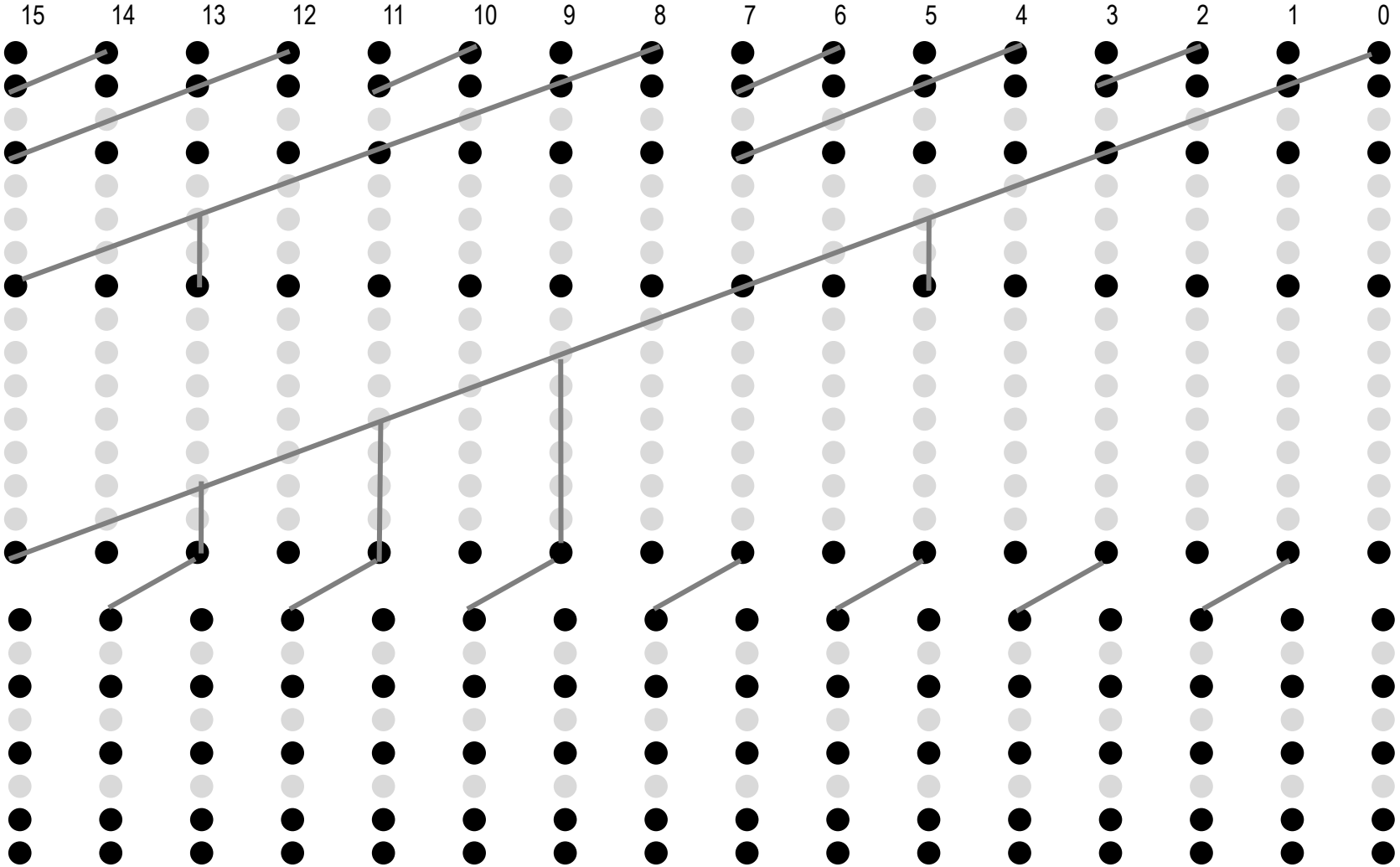


Ladner-Fischer(S+B)

Sklansky————Brent Kung

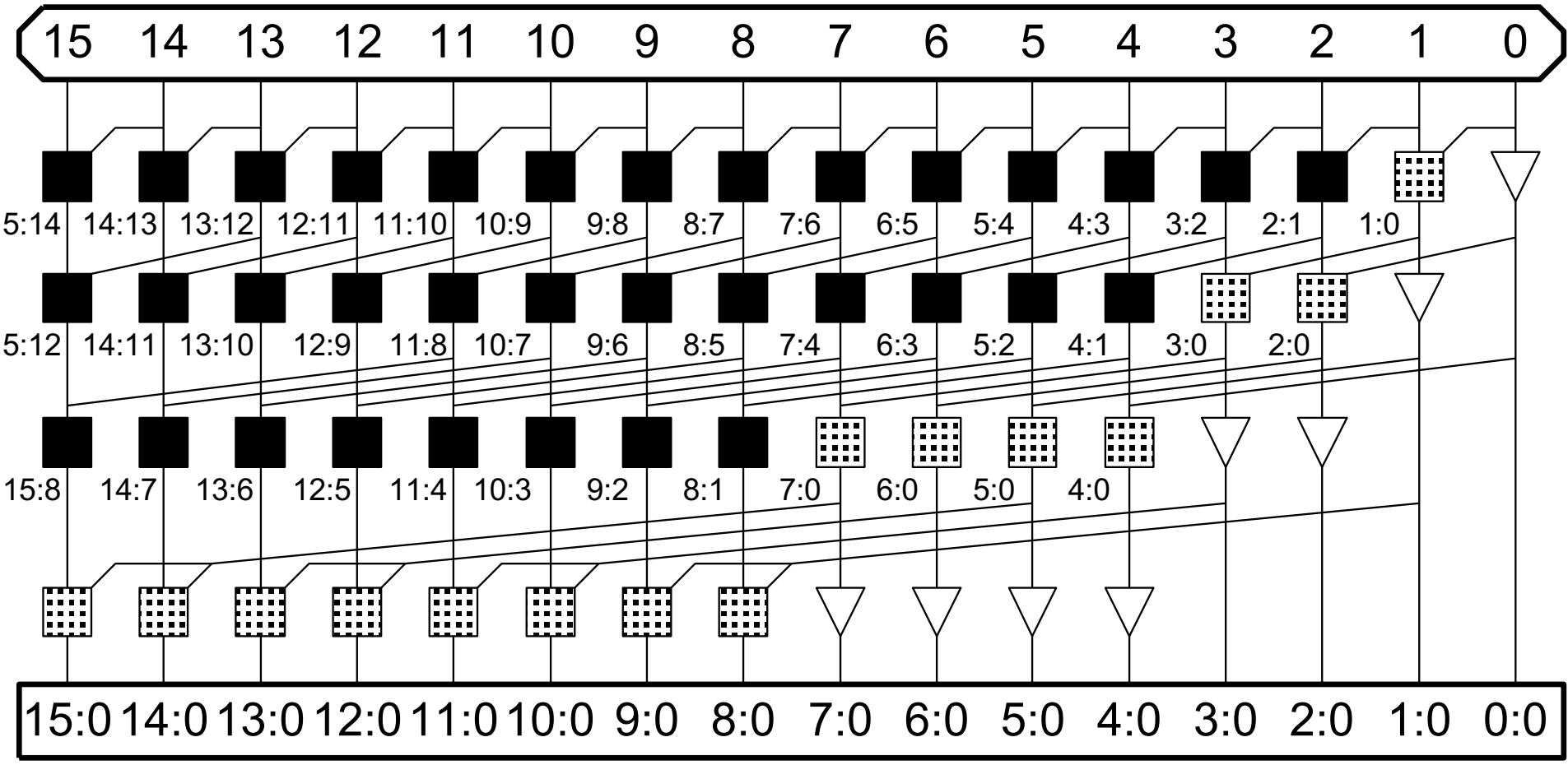


Ladner-Fischer(S+B)

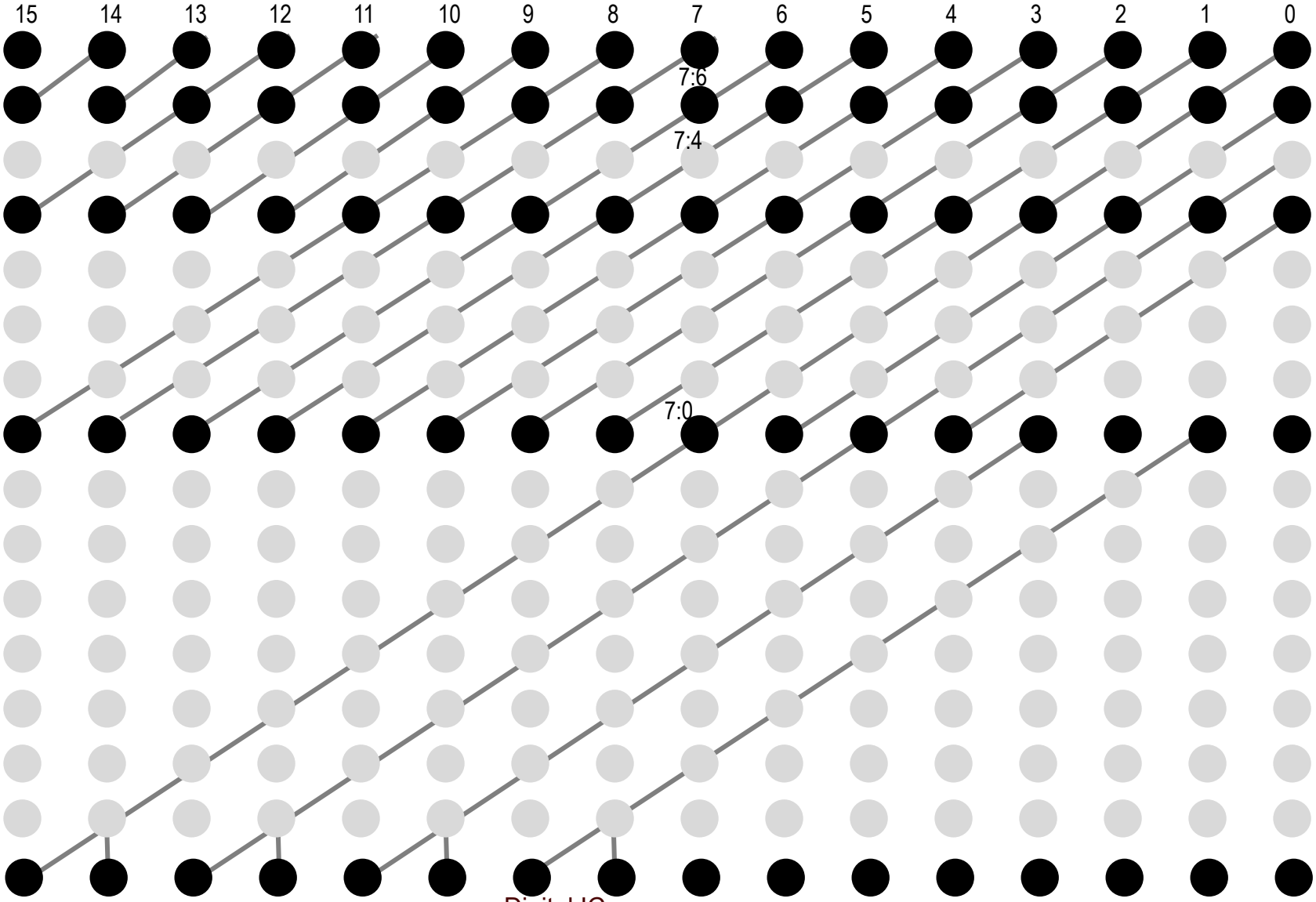


Knowles [1, 1, 1,2](S+K)

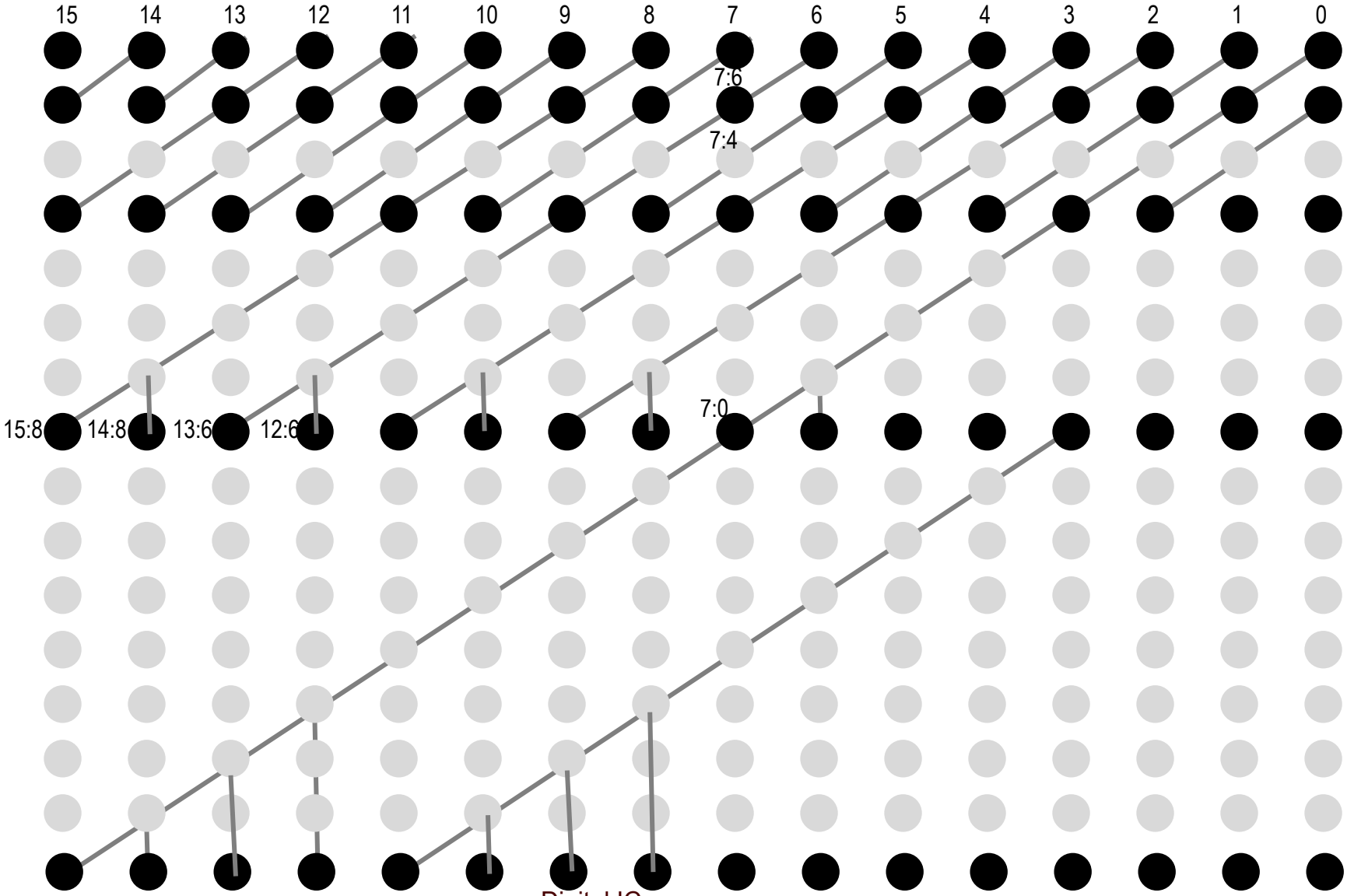
Kogge Stone————Sklansky



Knowles [1,1,1,2](S+K)

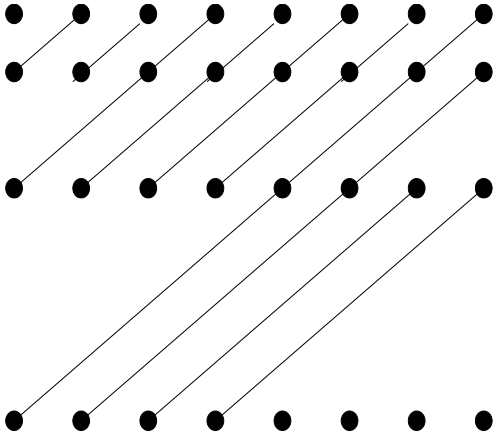


Knowles [1,1,2,4](S+K)

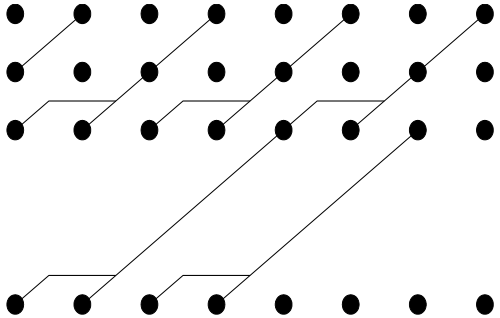


Knowles' 8-bit prefix trees

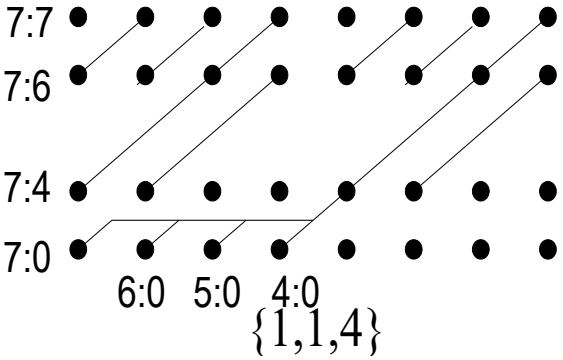
- All trees are log-depth



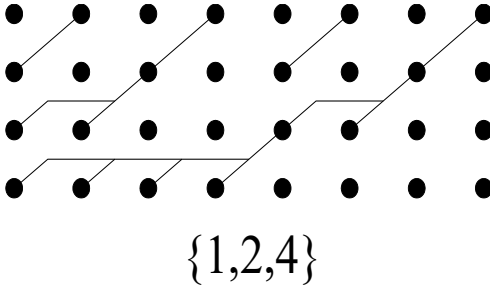
{1,1,1}



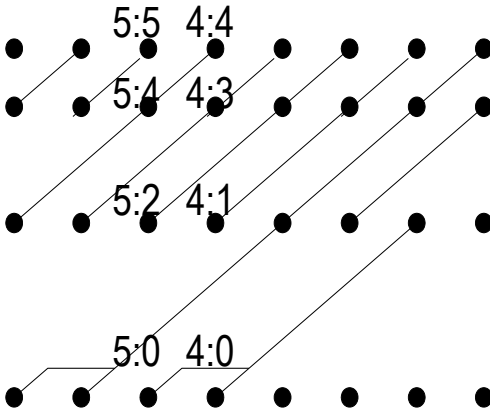
{1,2,2}



{1,1,4}



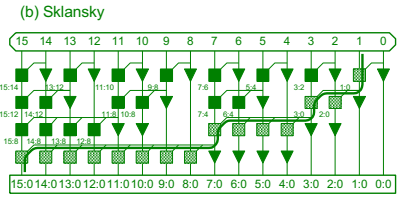
{1,2,4}

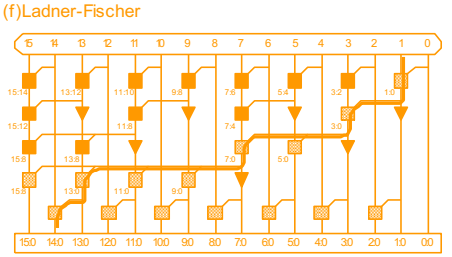


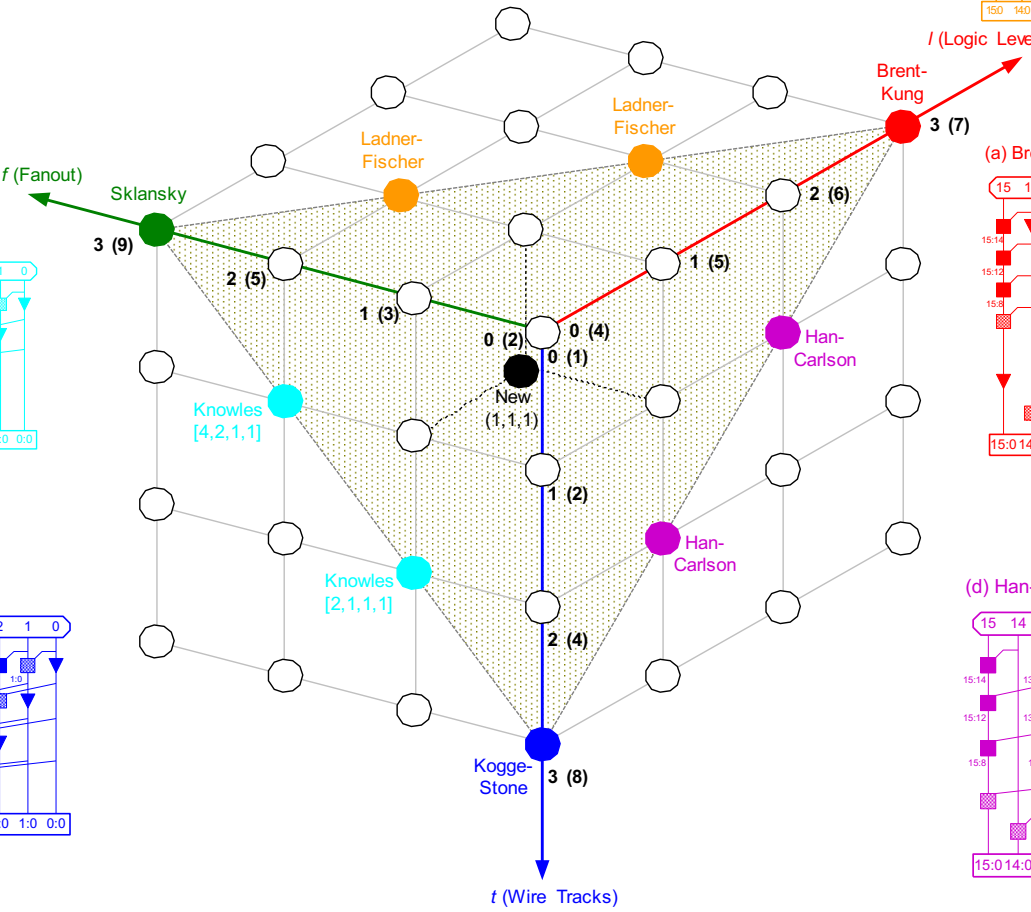
{1,1,2}

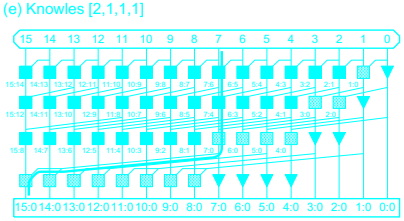
$G_{4:1}+P_{4:1}G_{1:0}=G_{4:1}+P_{4:1}(G_{1:1}+P_{1:1}G_{0:0})$
$=G_{4:1}+P_{4:1}G_{0:0}$

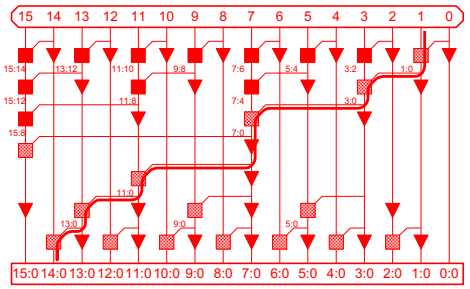
Taxonomy Revisited

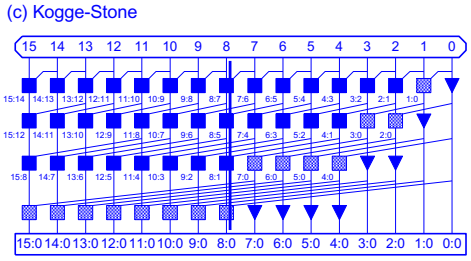


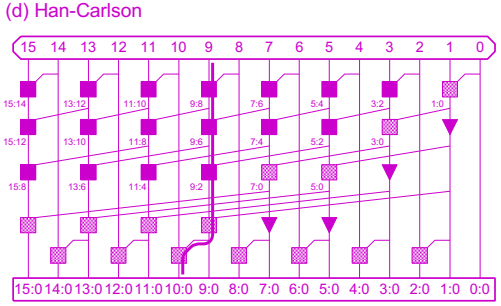












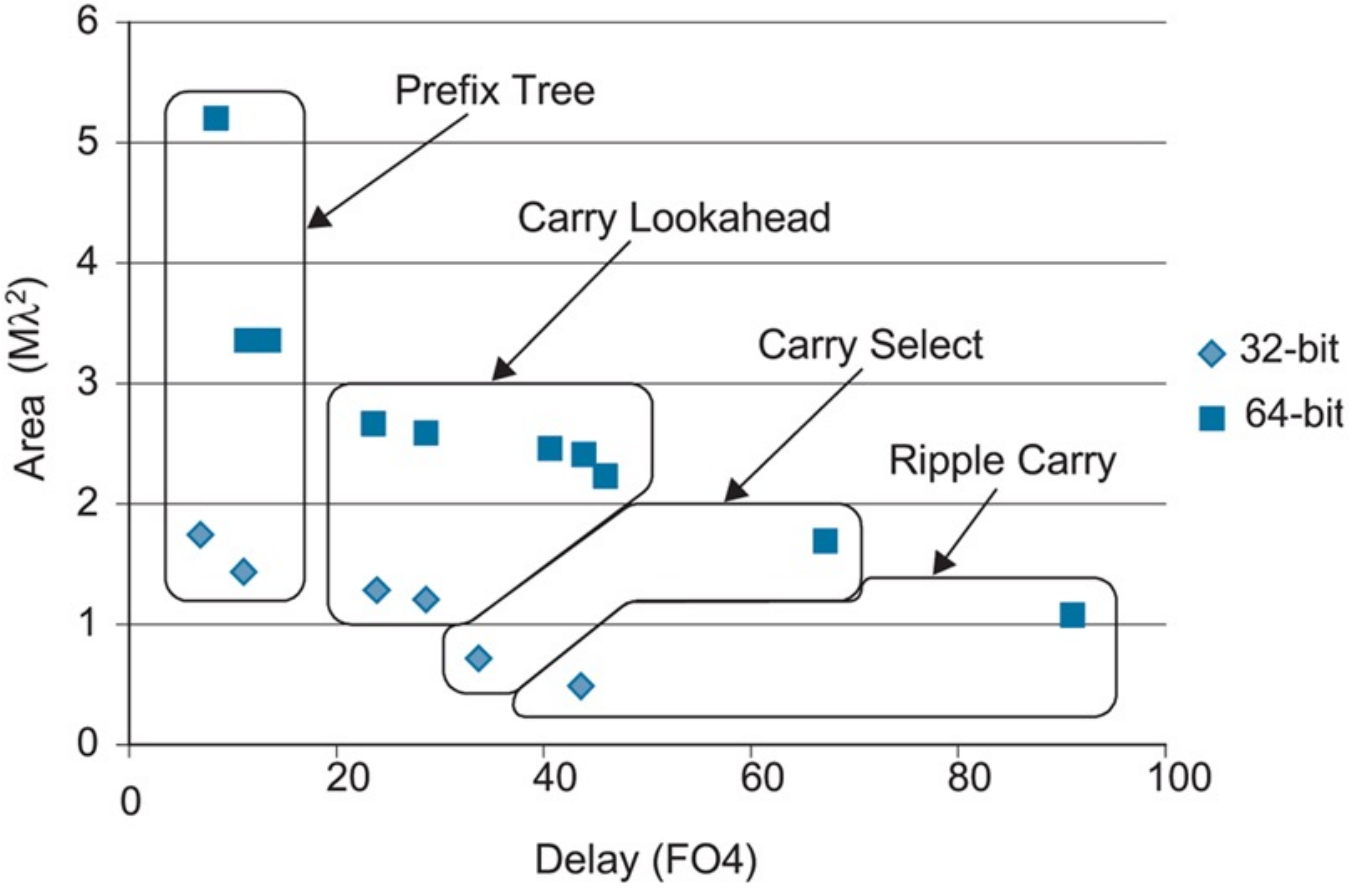
Summary

Adder architectures offer area / power / delay tradeoffs.

Choose the best one for your application.

Architecture	**Classification**	**Logic Levels**	**Max Fanout**	**Tracks**	**Cells**
Carry-Ripple		N-1	1	1	N
Carry-Skip n=4		N/4 + 5	2	1	1.25N
Carry-Inc. n=4		N/4 + 2	4	1	2N
Brent-Kung	(L-1, 0, 0)	$2\log_2 N - 1$	2	1	2N
Sklansky	(0, L-1, 0)	$\log_2 N$	N/2 + 1	1	$0.5\ N\log_2 N$
Kogge-Stone	(0, 0, L-1)	$\log_2 N$	2	N/2	$N\log_2 N$

summary


Prefix Tree
Carry Lookahead
Carry Select
Ripple Carry
32-bit
64-bit
Area (Mλ²)
Delay (FO4)
0
1
2
3
4
5
6
0
20
40
60
80
100