

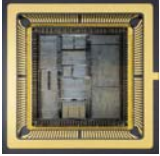


Lecture 14 : Adders

CSCI 5330 Digital CMOS VLSI Design

Instructor: Saraju P. Mohanty, Ph. D.

NOTE: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.



Outline

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

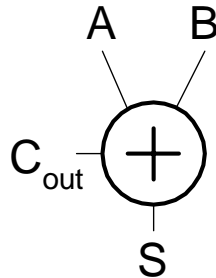


Single-Bit Addition

Half Adder

$$S = A \oplus B$$

$$C_{out} = A \cdot B$$

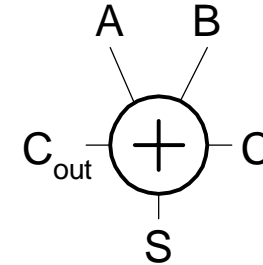


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A, B, C)$$

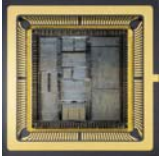


A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



PGK

- For a full adder, define what happens to carries
 - Generate: $C_{out} = 1$ independent of C
 - $G = A \cdot B$
 - Propagate: $C_{out} = C$
 - $P = A \oplus B$
 - Kill: $C_{out} = 0$ independent of C
 - $K = \sim A \cdot \sim B$

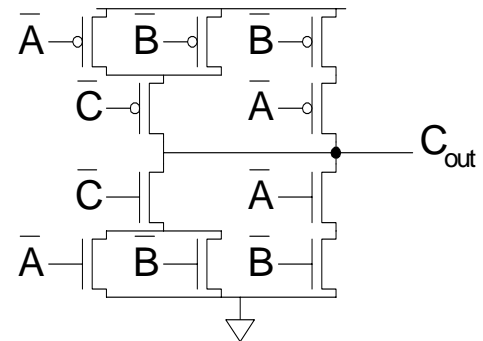
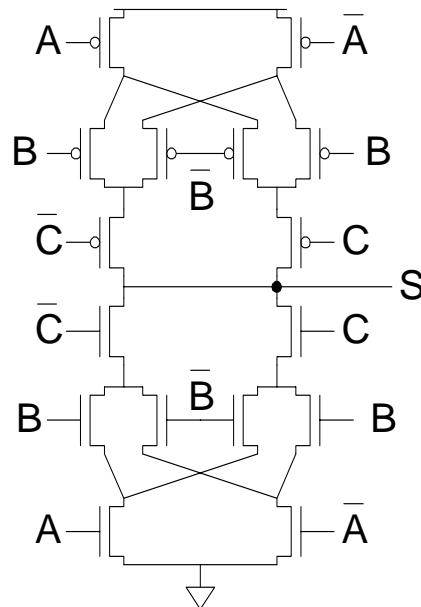
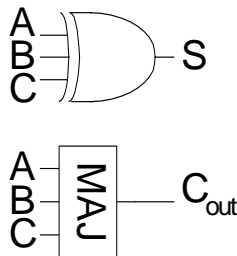
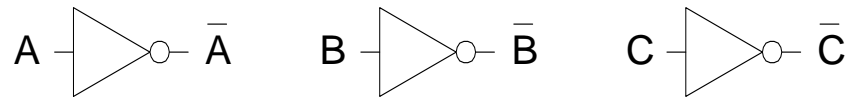


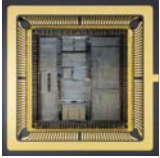
Full Adder Design I

- Brute force implementation from eqns

$$S = A \oplus B \oplus C$$

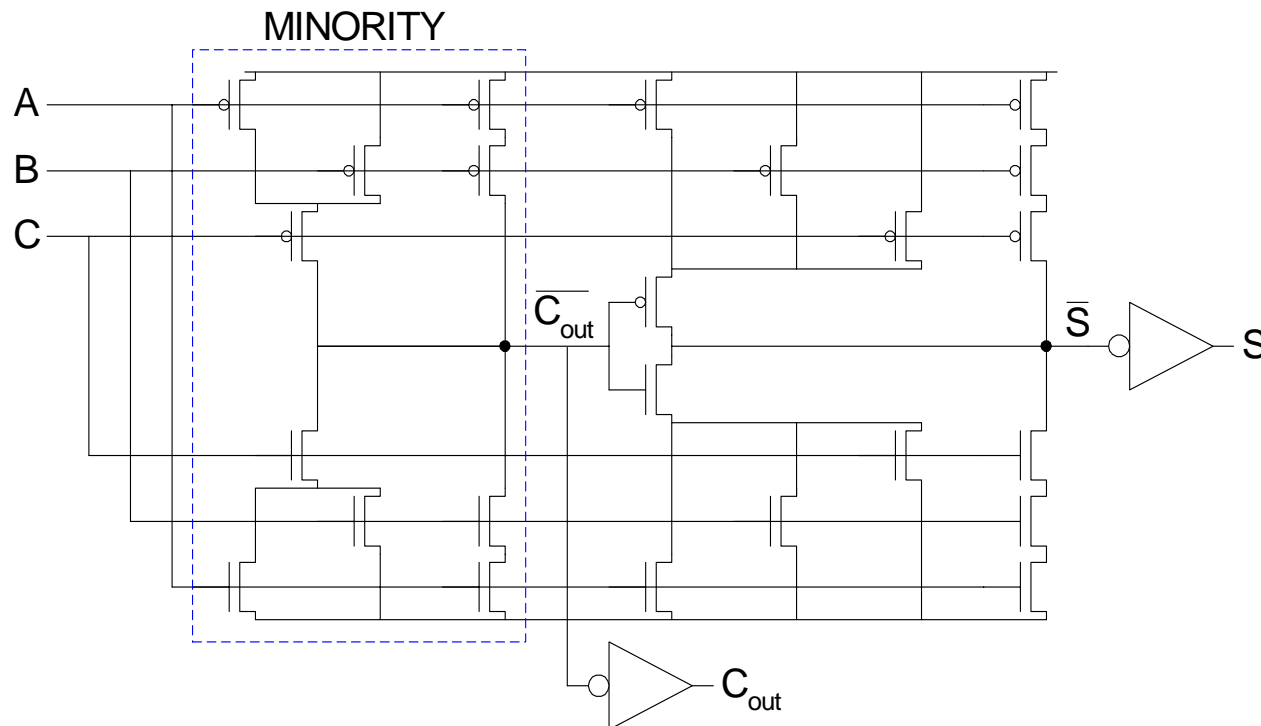
$$C_{out} = MAJ(A, B, C)$$

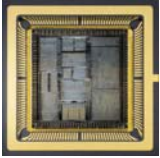




Full Adder Design II

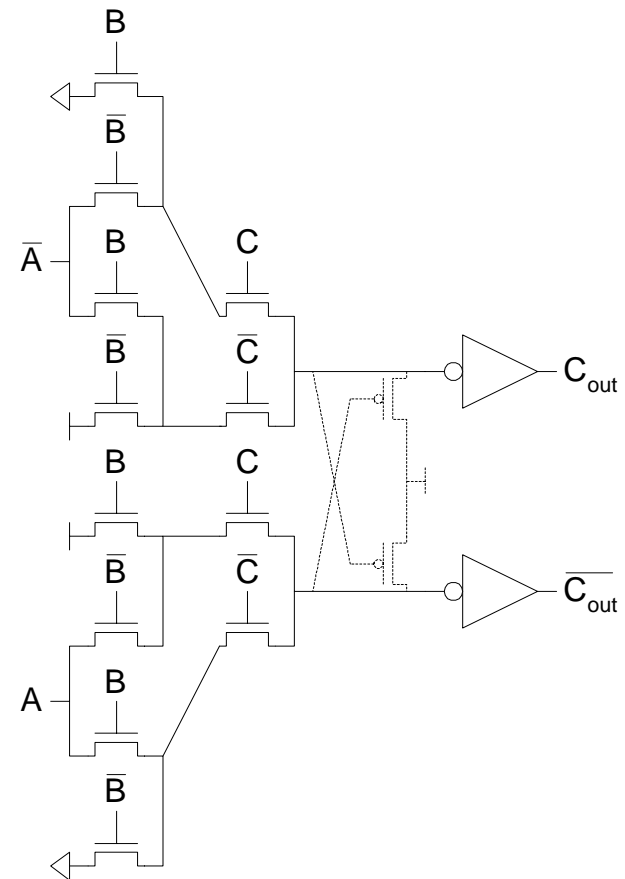
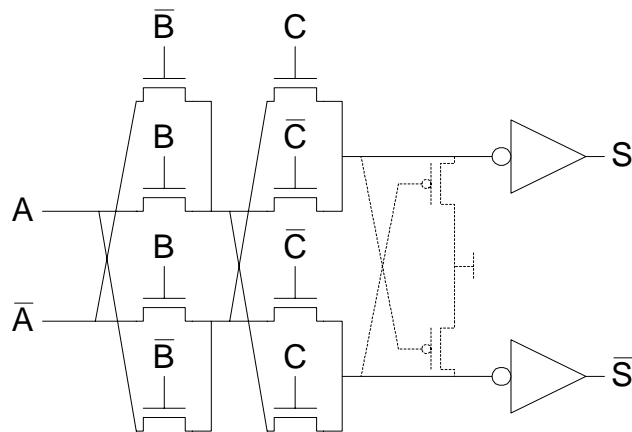
- Factor S in terms of C_{out}
$$S = ABC + (A + B + C)(\sim C_{out})$$
- Critical path is usually C to C_{out} in ripple adder

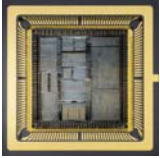




Full Adder Design III

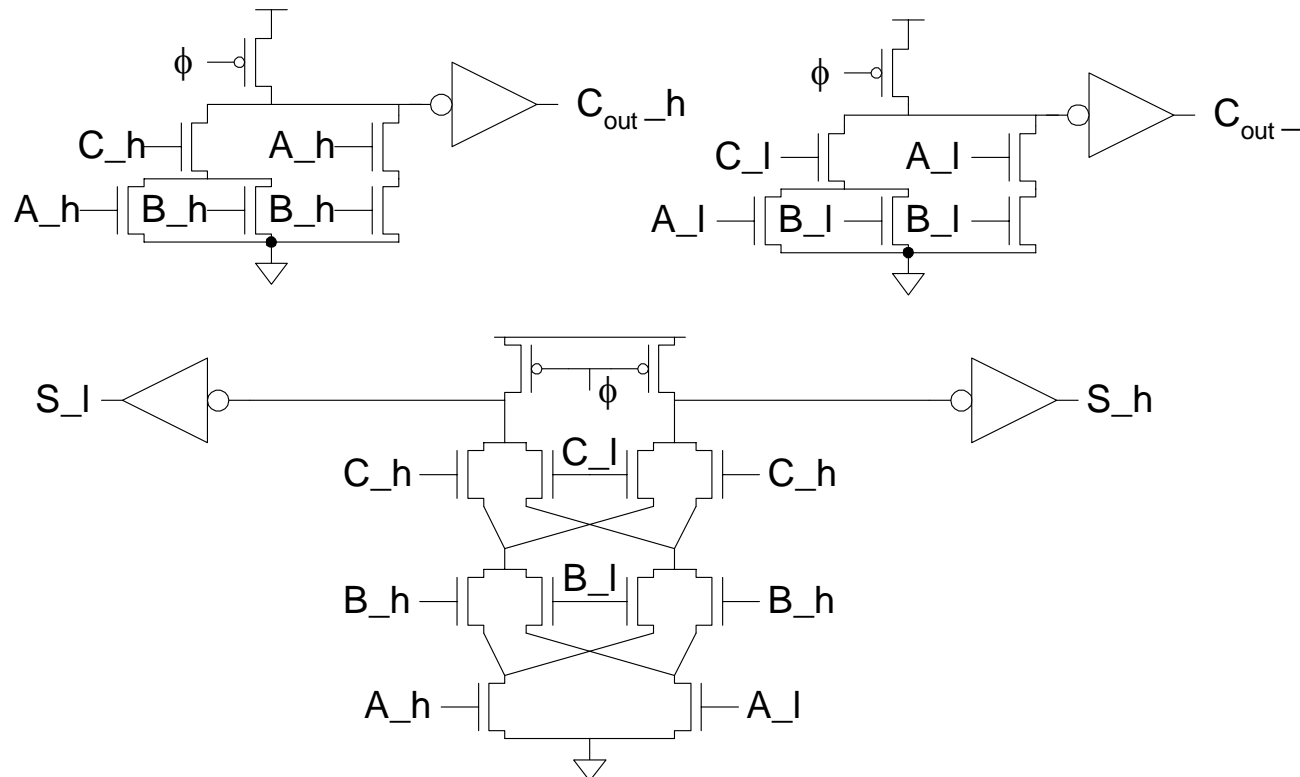
- Complementary Pass Transistor Logic (CPL)
 - Slightly faster, but more area

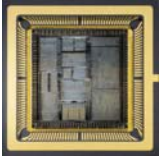




Full Adder Design IV

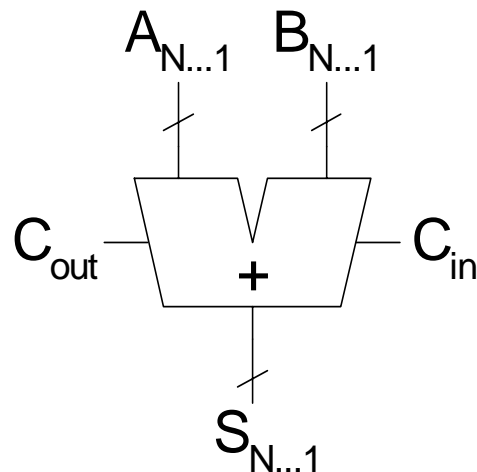
- Dual-rail domino
 - Very fast, but large and power hungry
 - Used in very fast multipliers





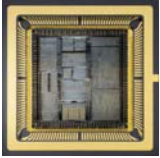
Carry Propagate Adders

- N-bit adder called CPA
 - Each sum bit depends on all previous carries
 - How do we compute all these carries quickly?



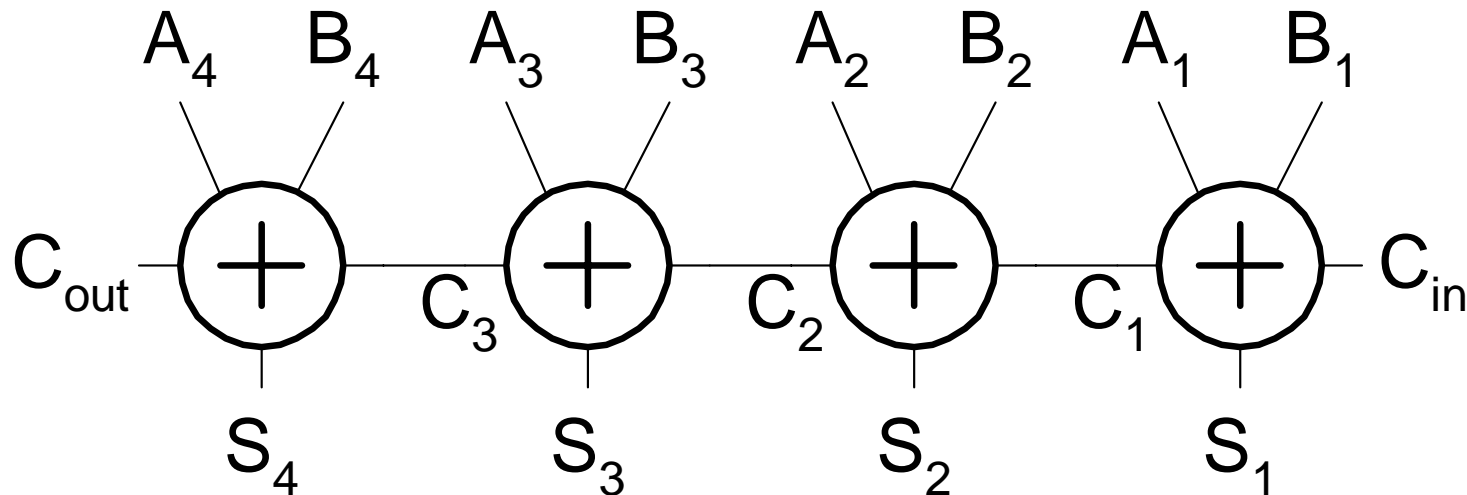
$$\begin{array}{r} \text{C}_{out} \swarrow \quad \nwarrow \text{C}_{in} \\ \textcircled{0}000\textcircled{0} \\ 1111 \\ +0000 \\ \hline 1111 \end{array}$$

$$\begin{array}{r} \text{C}_{out} \swarrow \quad \nwarrow \text{C}_{in} \\ \textcircled{1}111\textcircled{1} \\ 1111 \\ +0000 \\ \hline 0000 \end{array} \quad \begin{array}{l} \text{carries} \\ A_{4...1} \\ B_{4...1} \\ S_{4...1} \end{array}$$



Carry-Ripple Adder

- Simplest design: cascade full adders
 - Critical path goes from C_{in} to C_{out}
 - Design full adder to have fast carry delay





Generate / Propagate

- Equations often factored into G and P
- Generate and propagate for groups spanning $i:j$

$$G_{i:j} = G_{i:k} + P_{i:k} \square G_{k-1:j}$$

$$P_{i:j} = P_{i:k} \square P_{k-1:j} \quad \dots\dots\dots$$

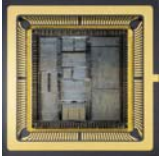
- Base case

$$G_{i:i} \equiv G_i = A_i \square B_i \qquad G_{0:0} \equiv G_0 = C_{in}$$

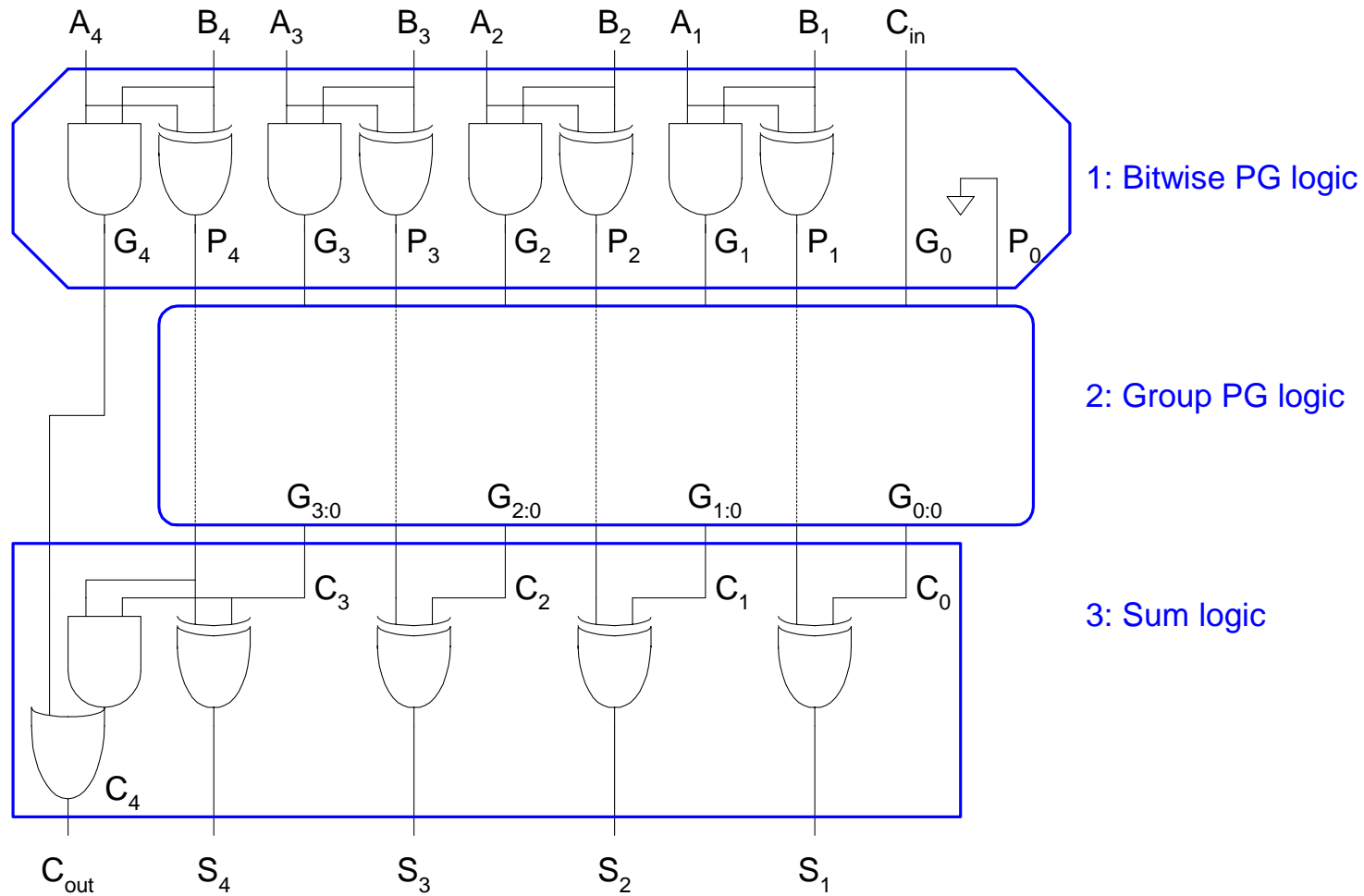
$$P_{i:i} \equiv P_i = A_i \oplus B_i \qquad P_{0:0} \equiv P_0 = 0$$

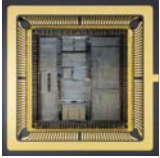
- Sum:

$$S_i = P_i \oplus G_{i-1:0}$$



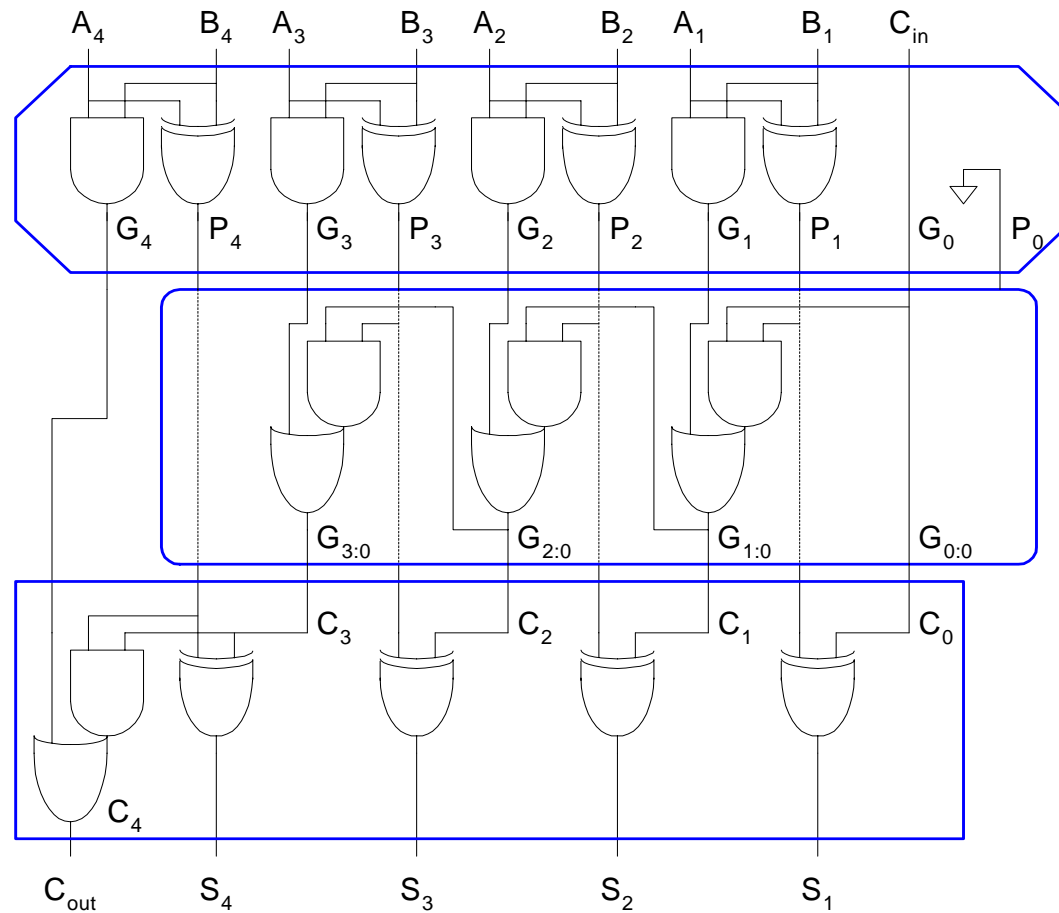
PG Logic

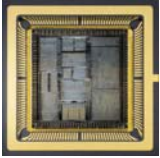




Carry-Ripple Revisited

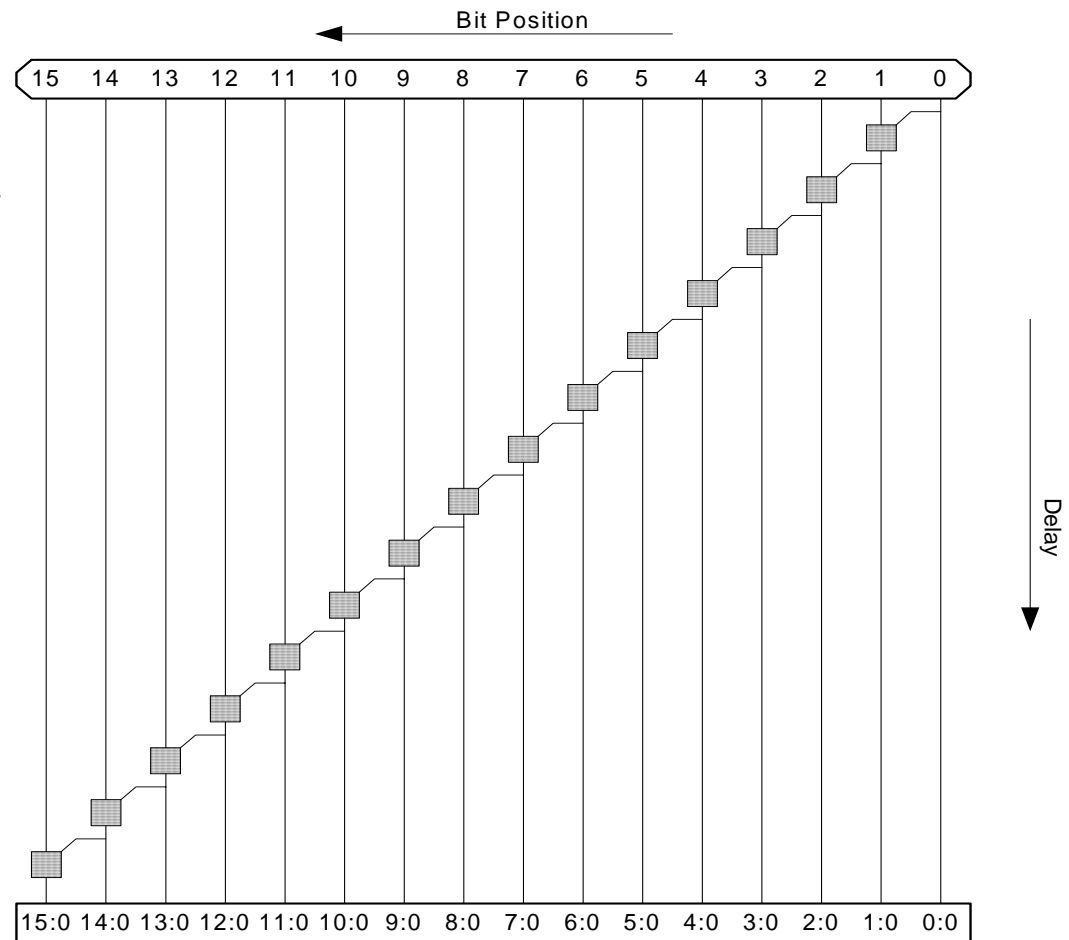
$$G_{i:0} = G_i + P_i \square G_{i-1:0}$$





Carry-Ripple PG Diagram

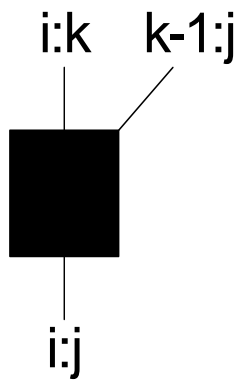
$$t_{\text{ripple}} = t_{pg} + (N - 1)t_{AO} + t_{\text{xor}}$$



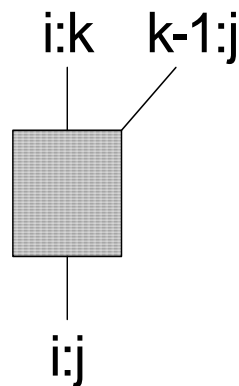


PG Diagram Notation

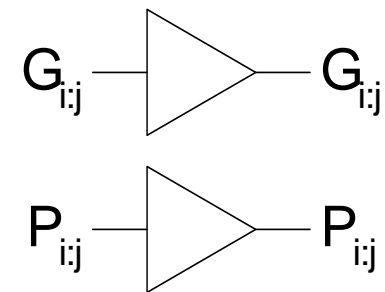
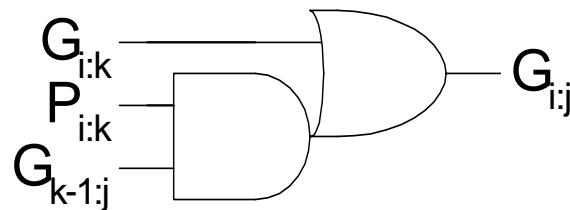
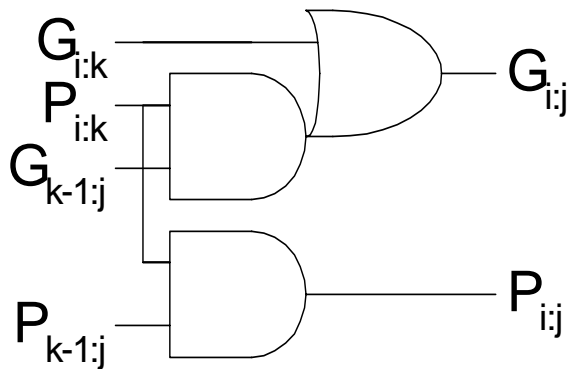
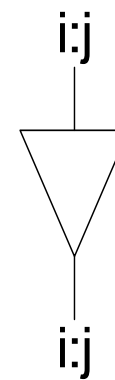
Black cell

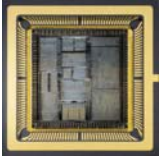


Gray cell



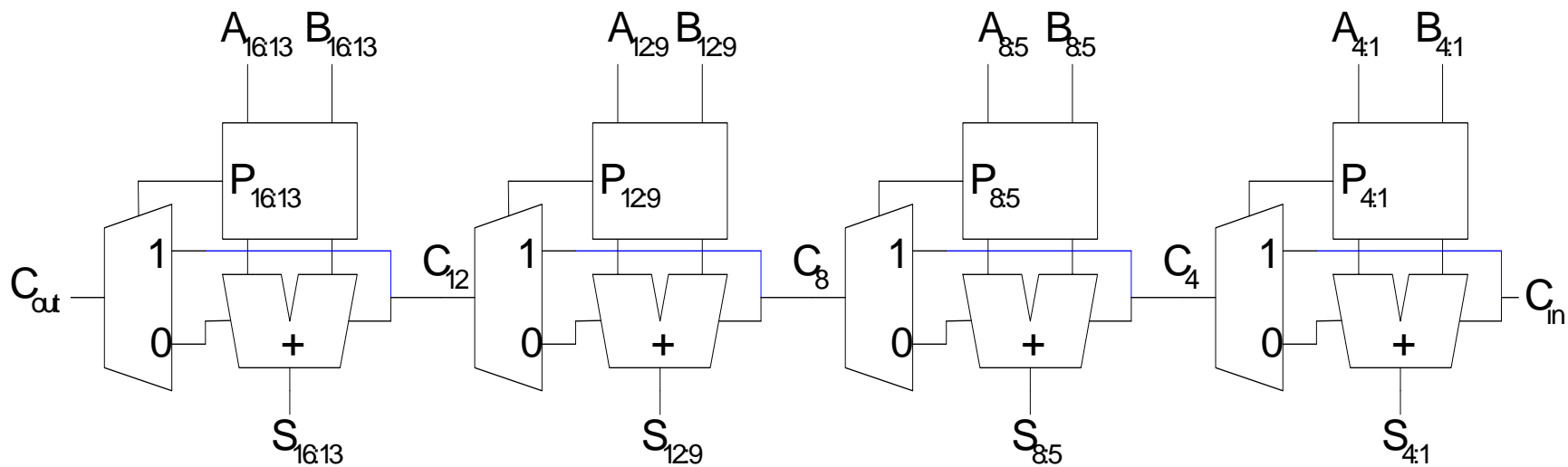
Buffer

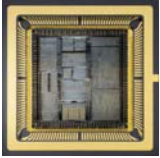




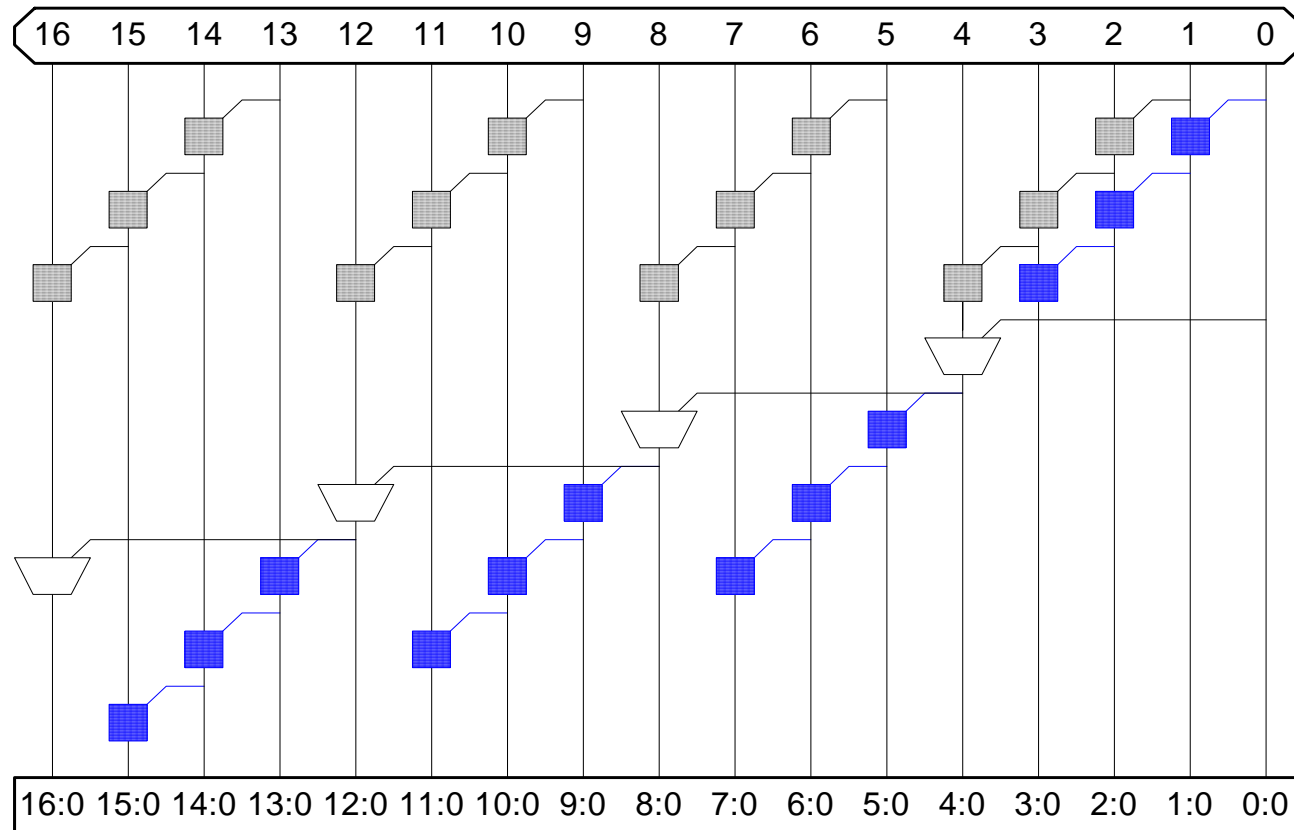
Carry-Skip Adder

- 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



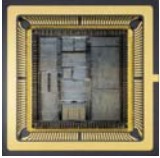


Carry-Skip PG Diagram

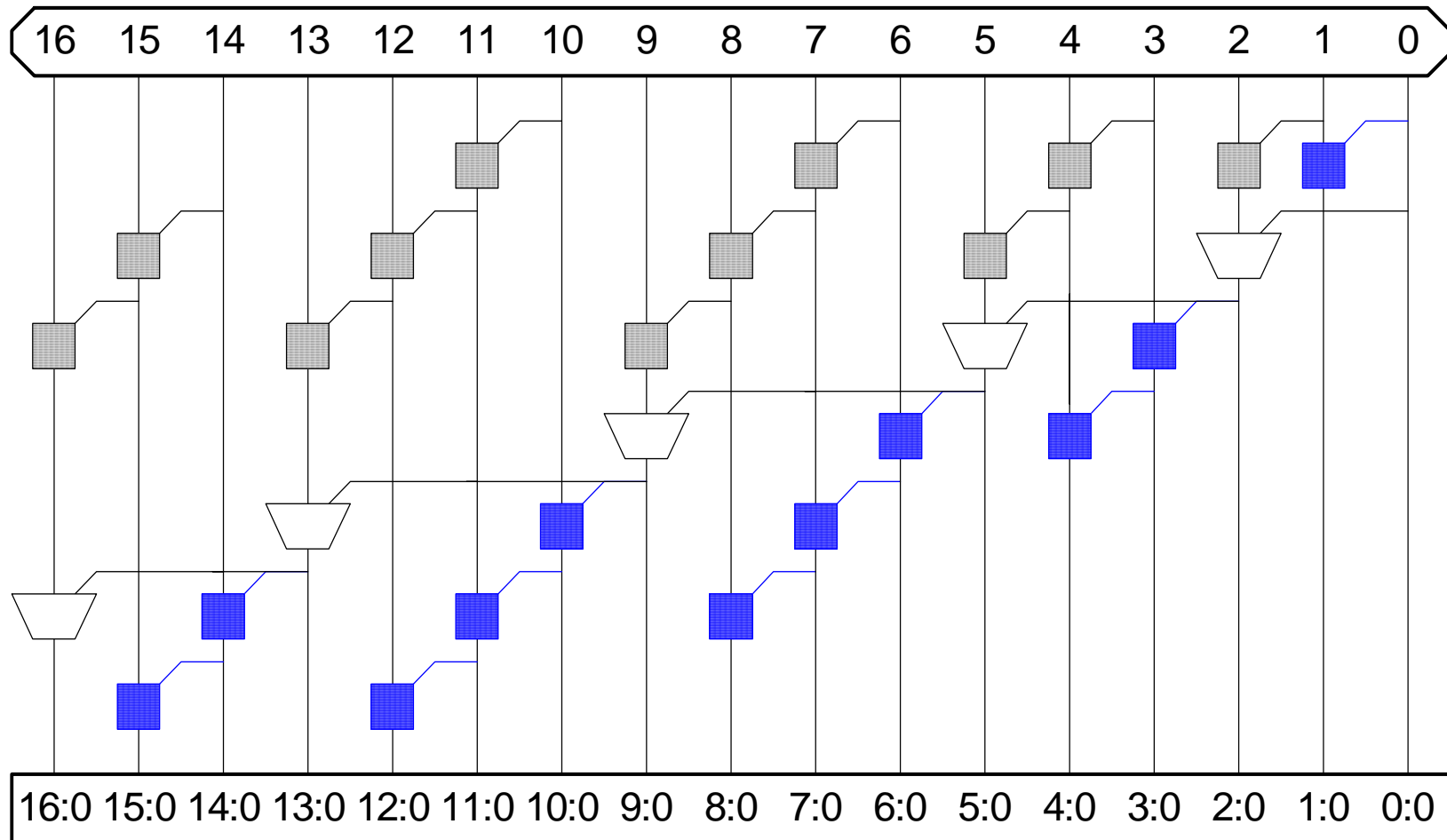


For k n-bit groups ($N = nk$)

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



Variable Group Size

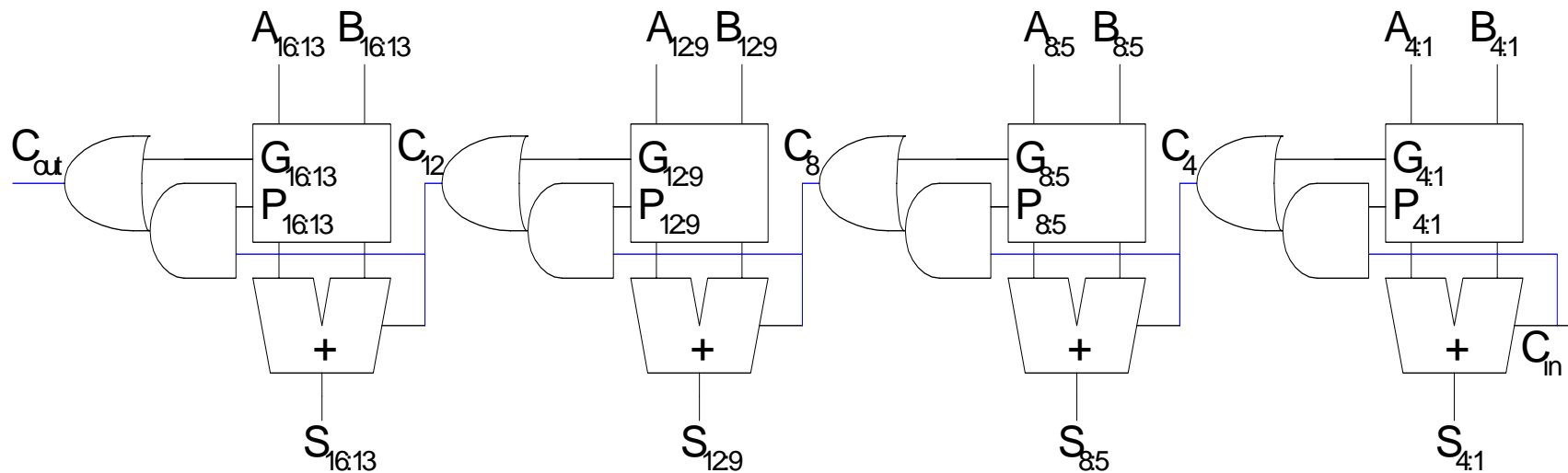


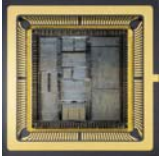
Delay grows as $O(\sqrt{N})$



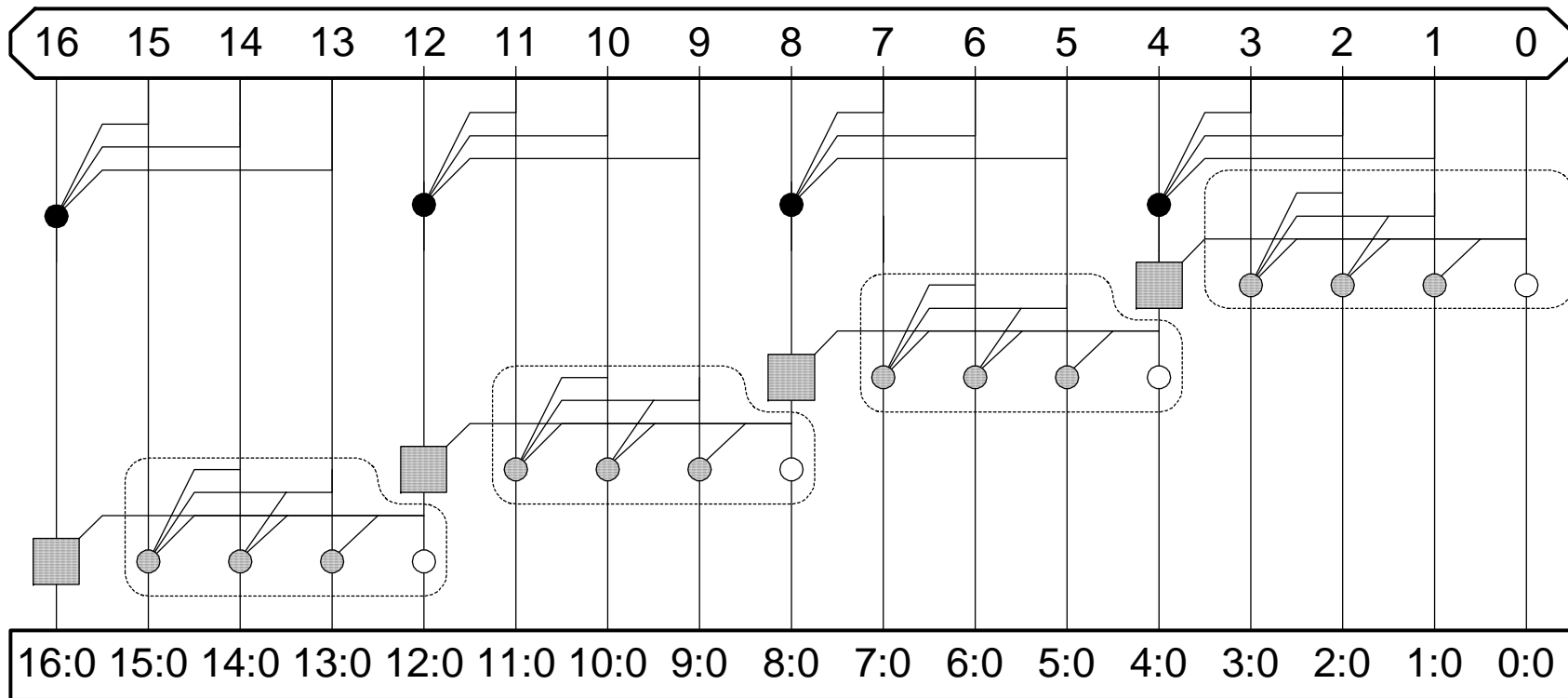
Carry-Lookahead Adder

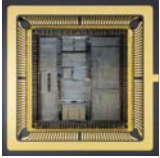
- Carry-lookahead adder computes $G_{i:0}$ for many bits in parallel.
- Uses higher-valency cells with more than two inputs.



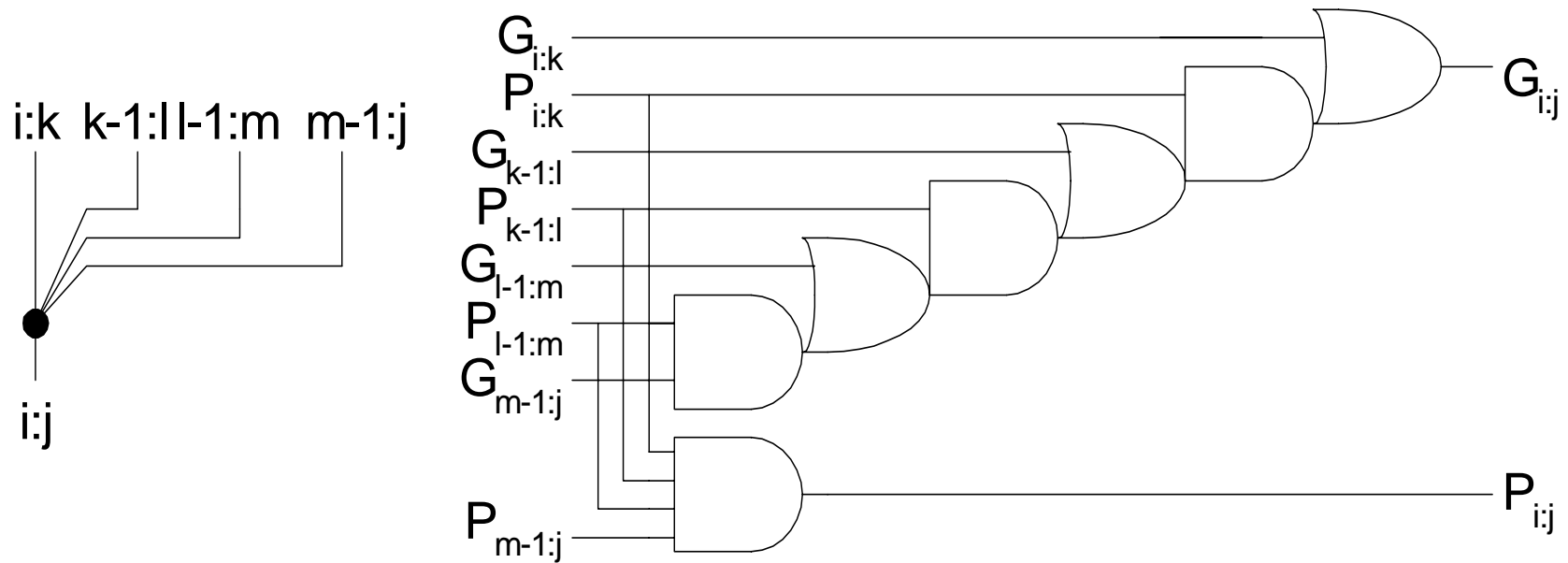


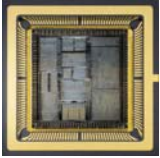
CLA PG Diagram





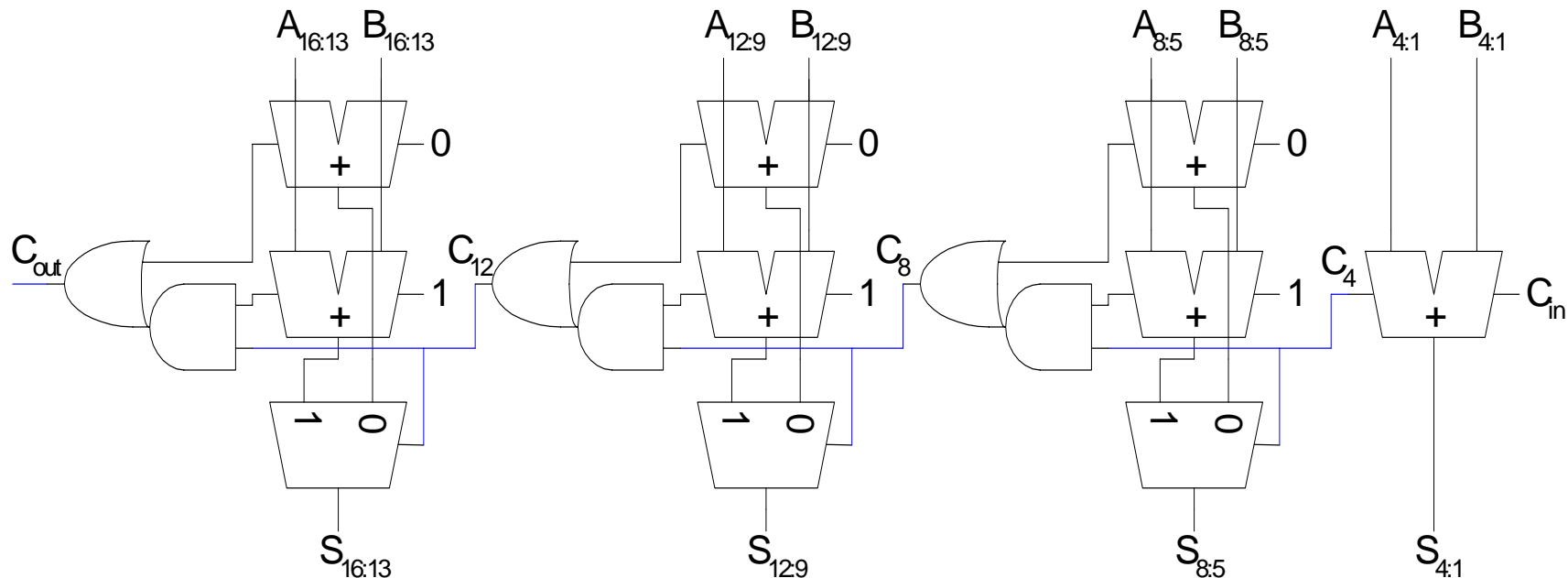
Higher-Valency Cells

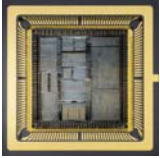




Carry-Select Adder

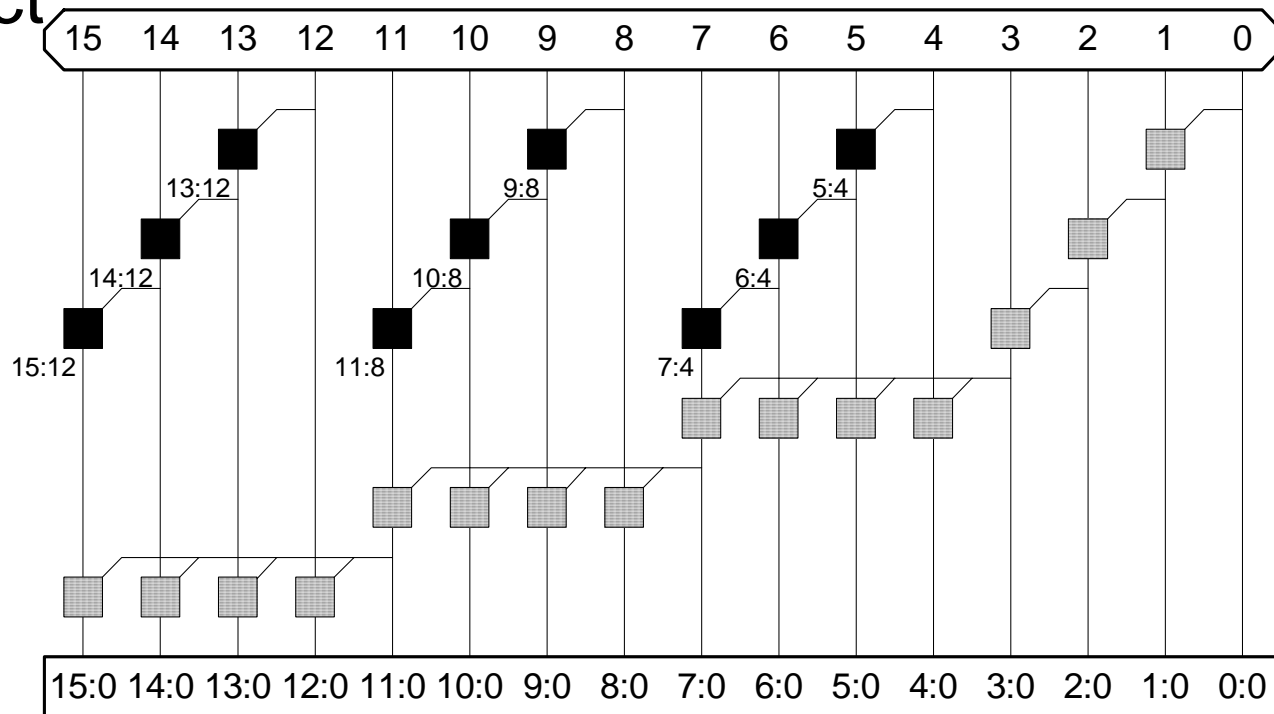
- Trick for critical paths dependent on late input X
 - Precompute two possible outputs for $X = 0, 1$
 - Select proper output when X arrives
- Carry-select adder precomputes n-bit sums
 - For both possible carries into n-bit group



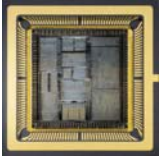


Carry-Increment Adder

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

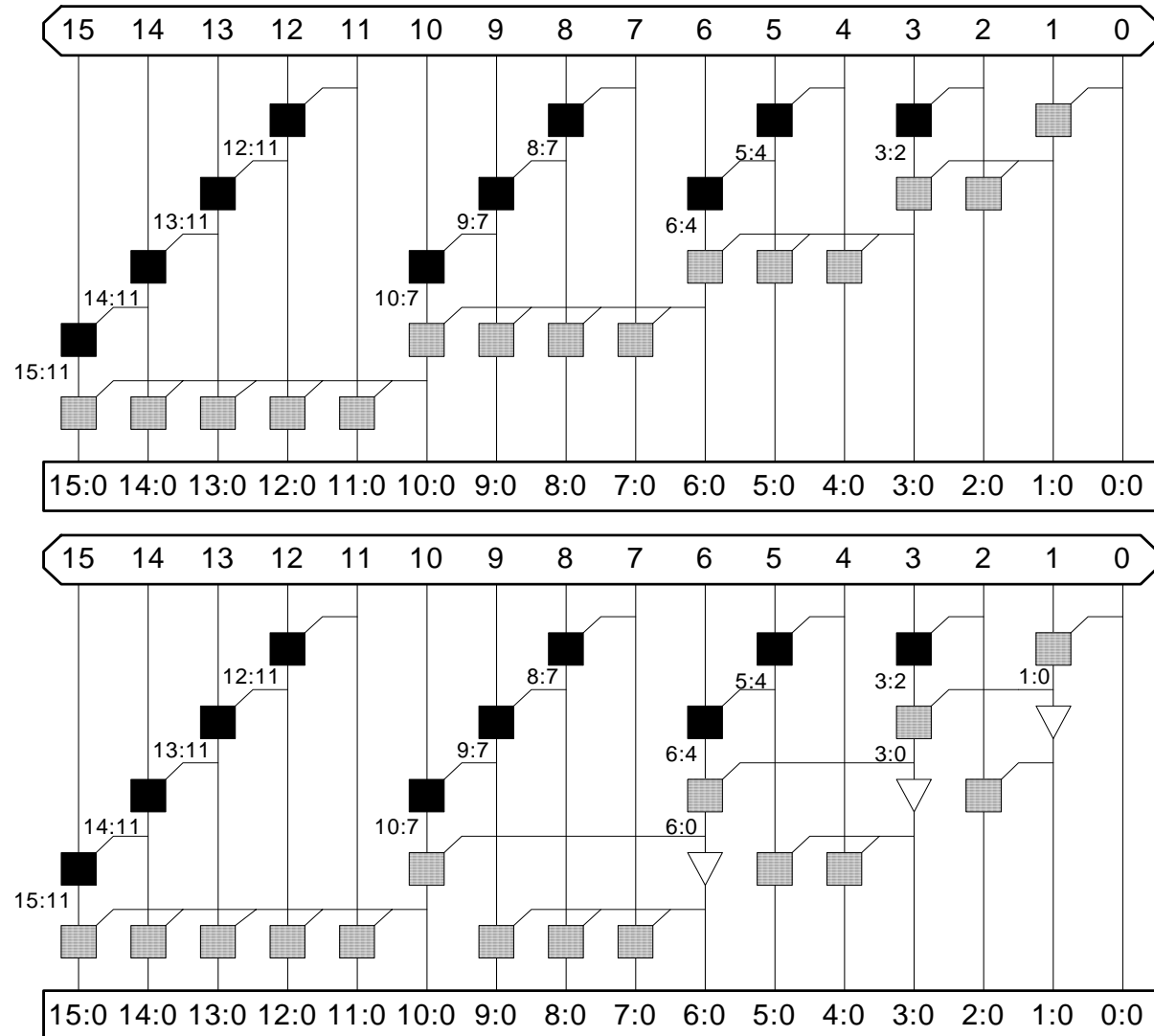


$$t_{\text{increment}} = t_{pg} + \left[(n-1) + (k-1) \right] t_{AO} + t_{xor}$$



Variable Group Size

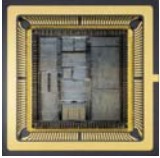
- Also buffer noncritical signals



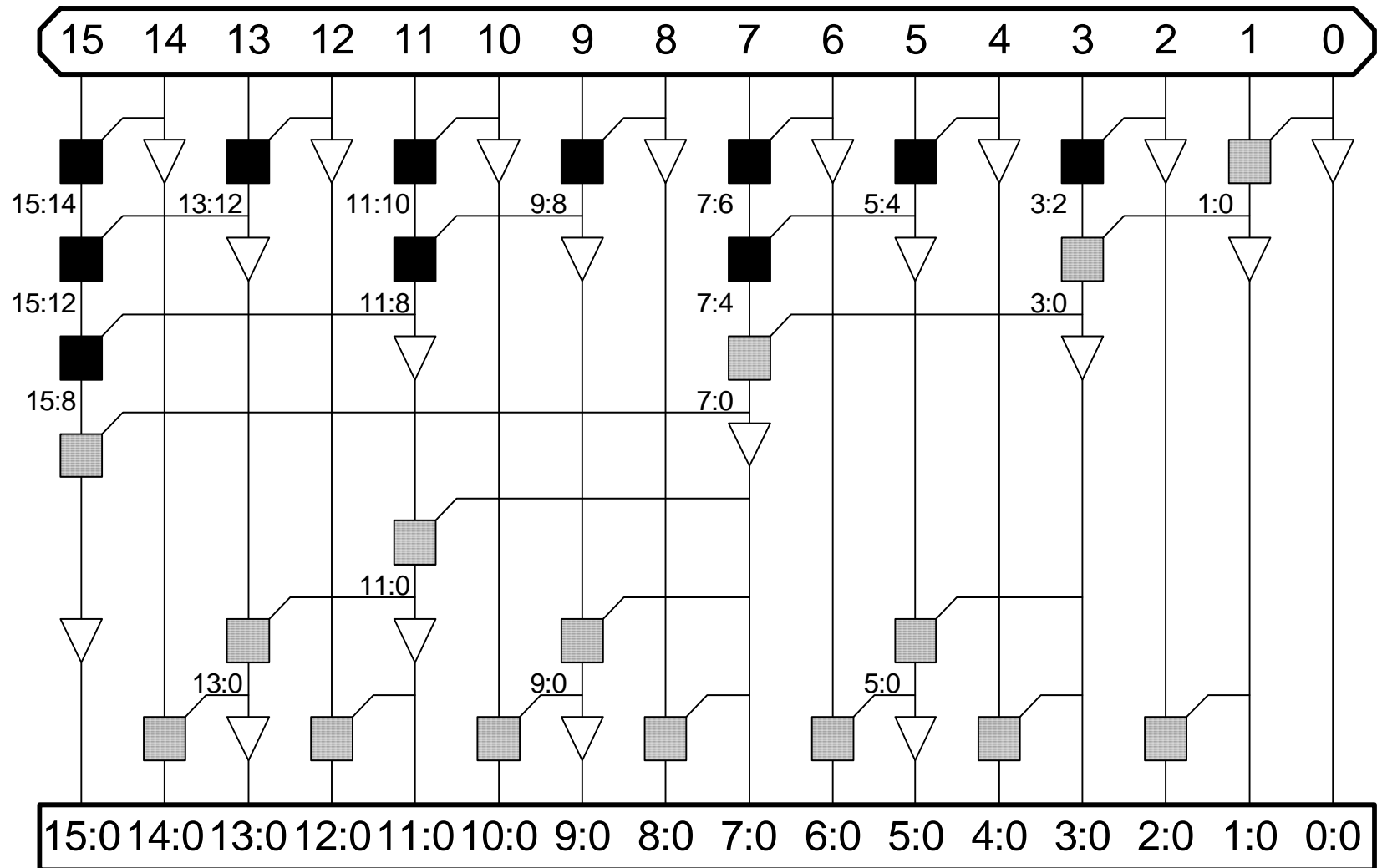


Tree Adder

- If lookahead is good, lookahead across lookahead!
 - Recursive lookahead gives $O(\log N)$ delay
- Many variations on tree adders

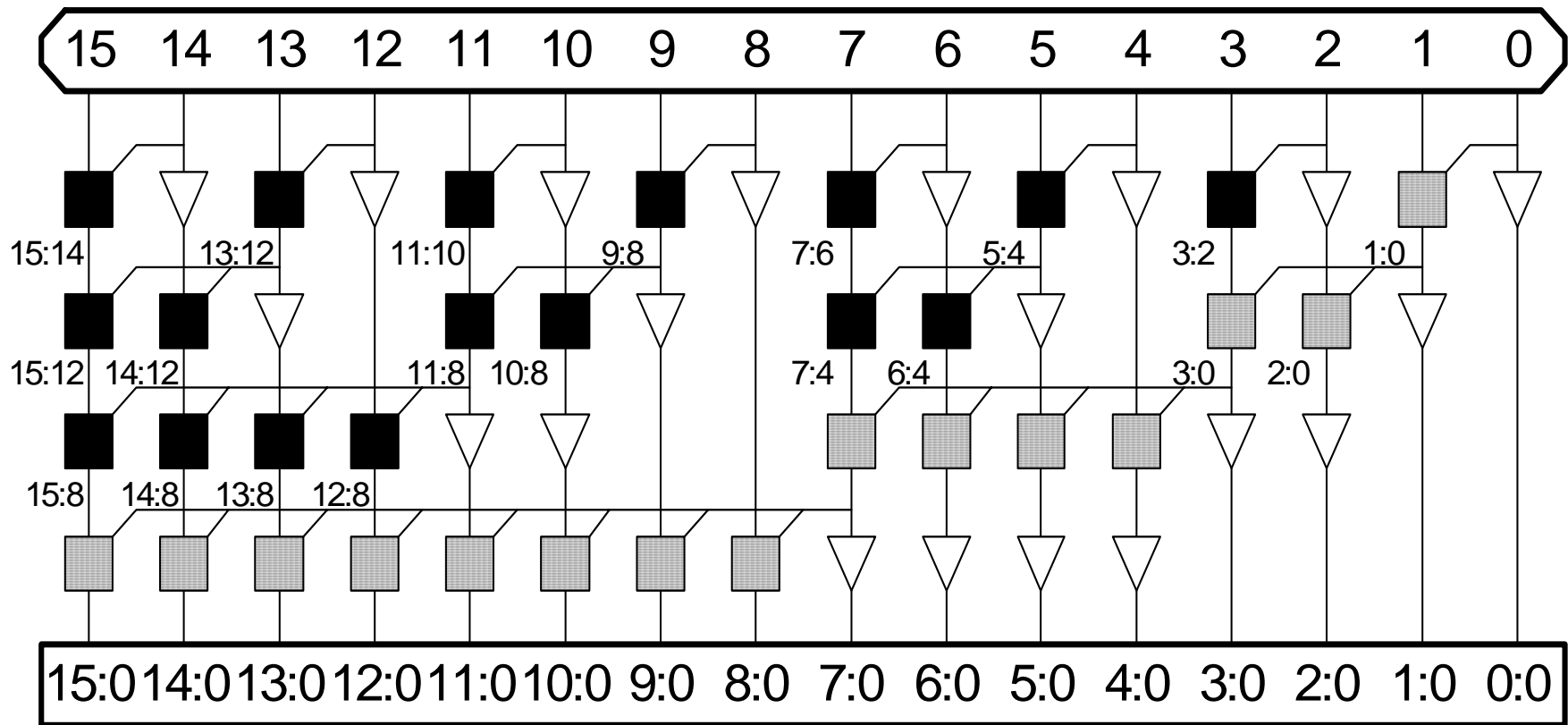


Brent-Kung



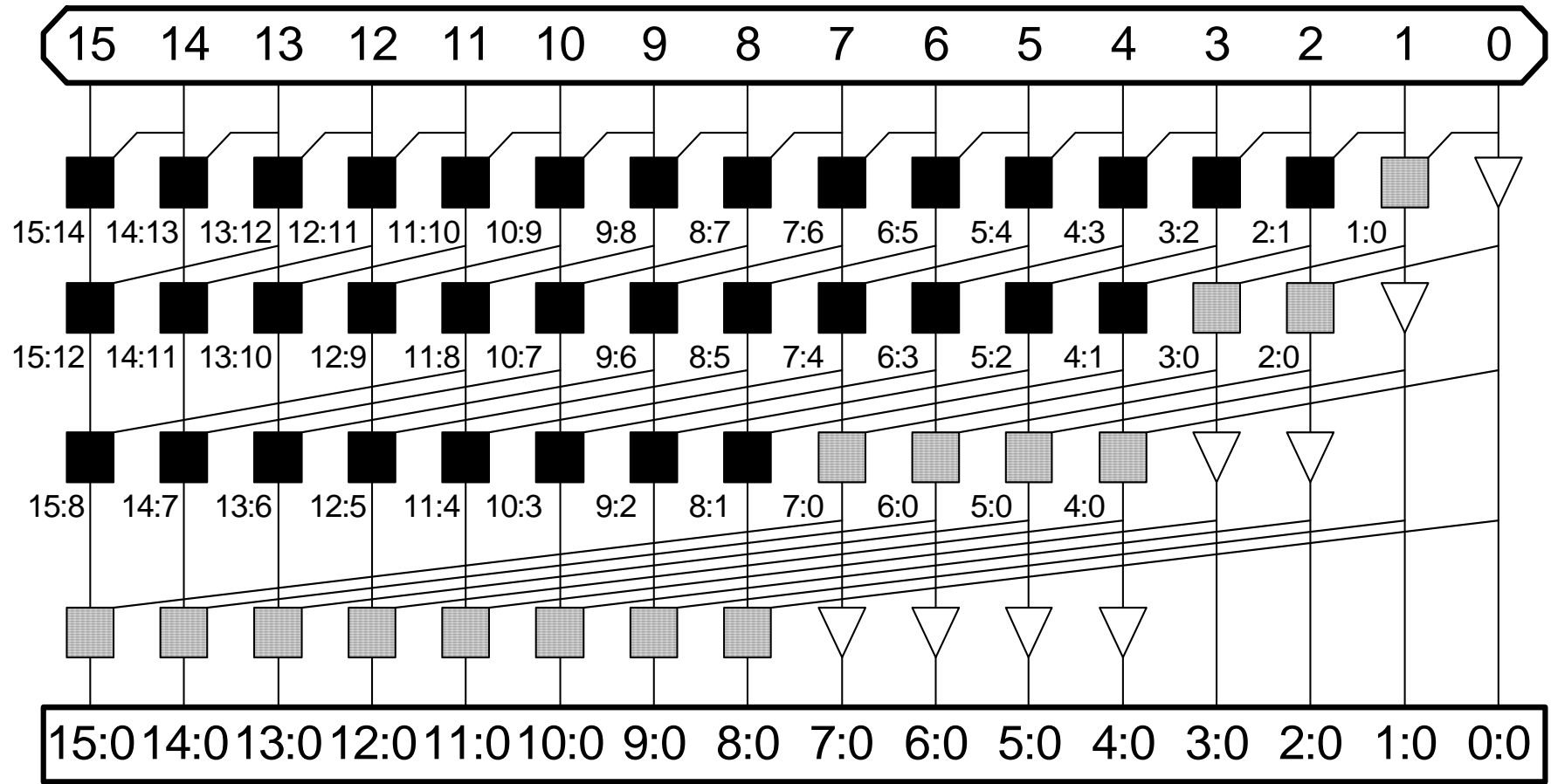


Sklansky





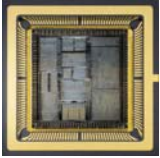
Kogge-Stone



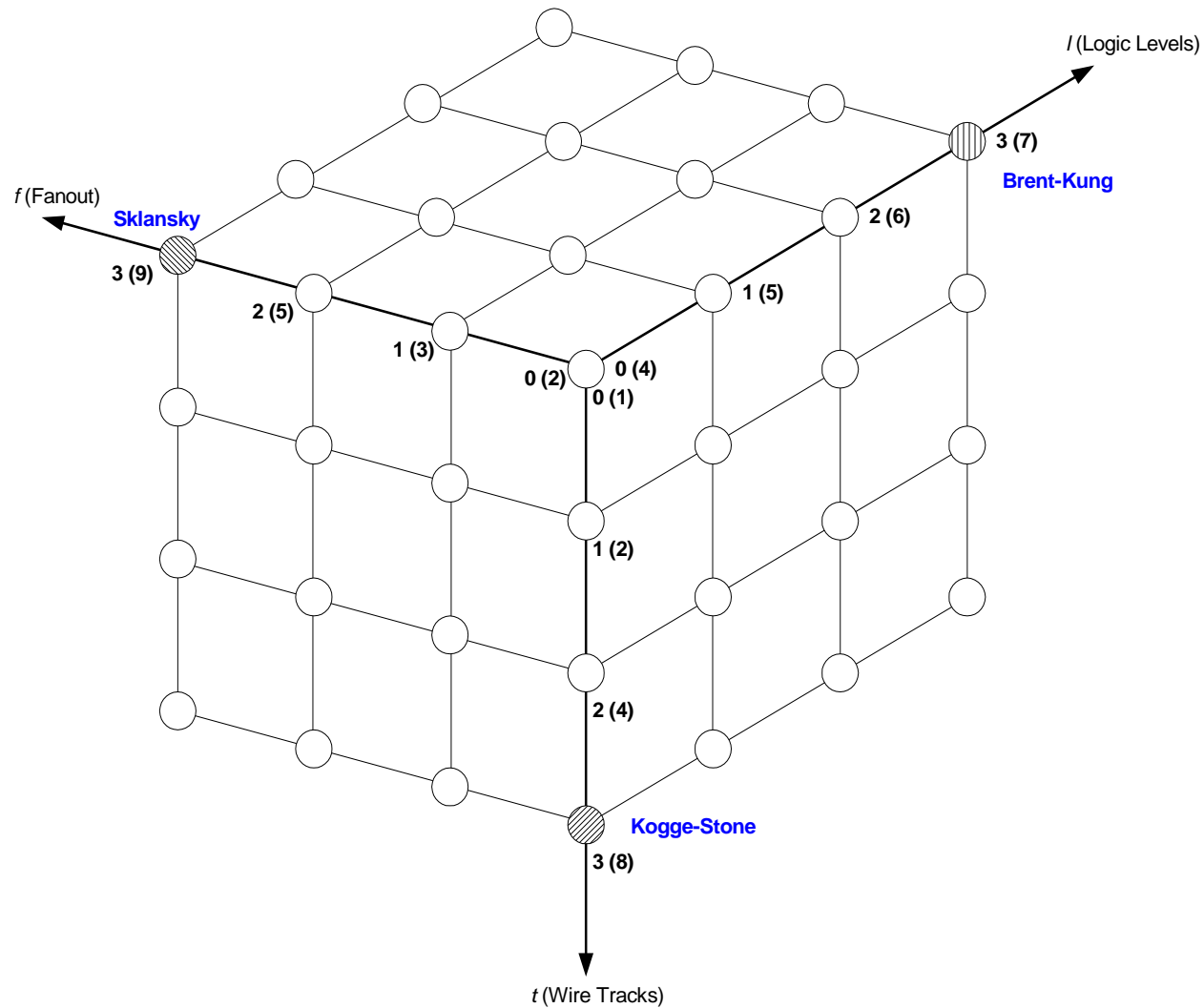


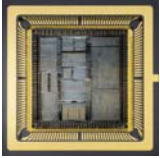
Tree Adder Taxonomy

- 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
$$l + f + t = L - 1$$

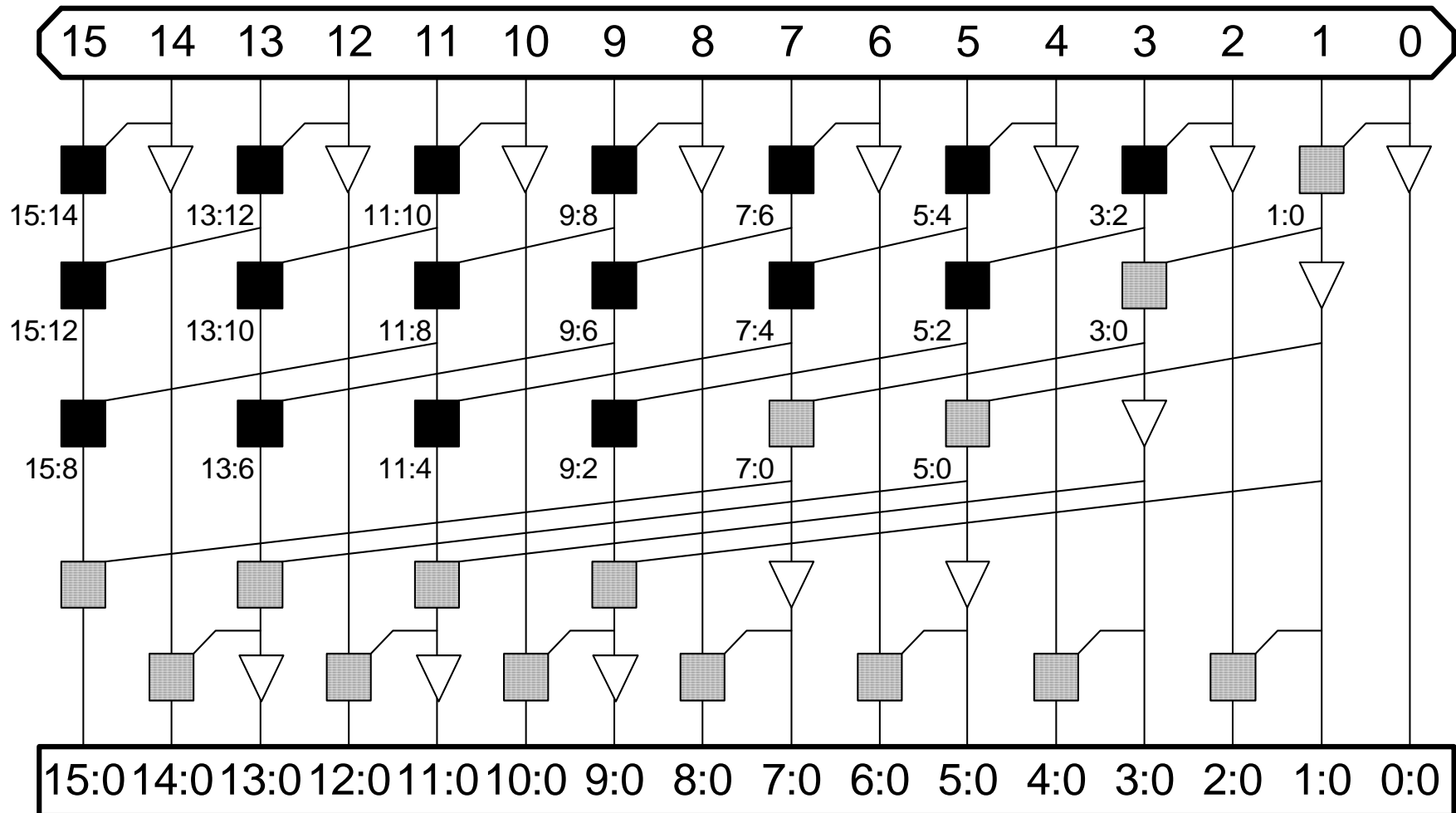


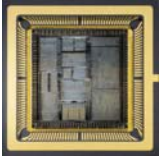
Tree Adder Taxonomy



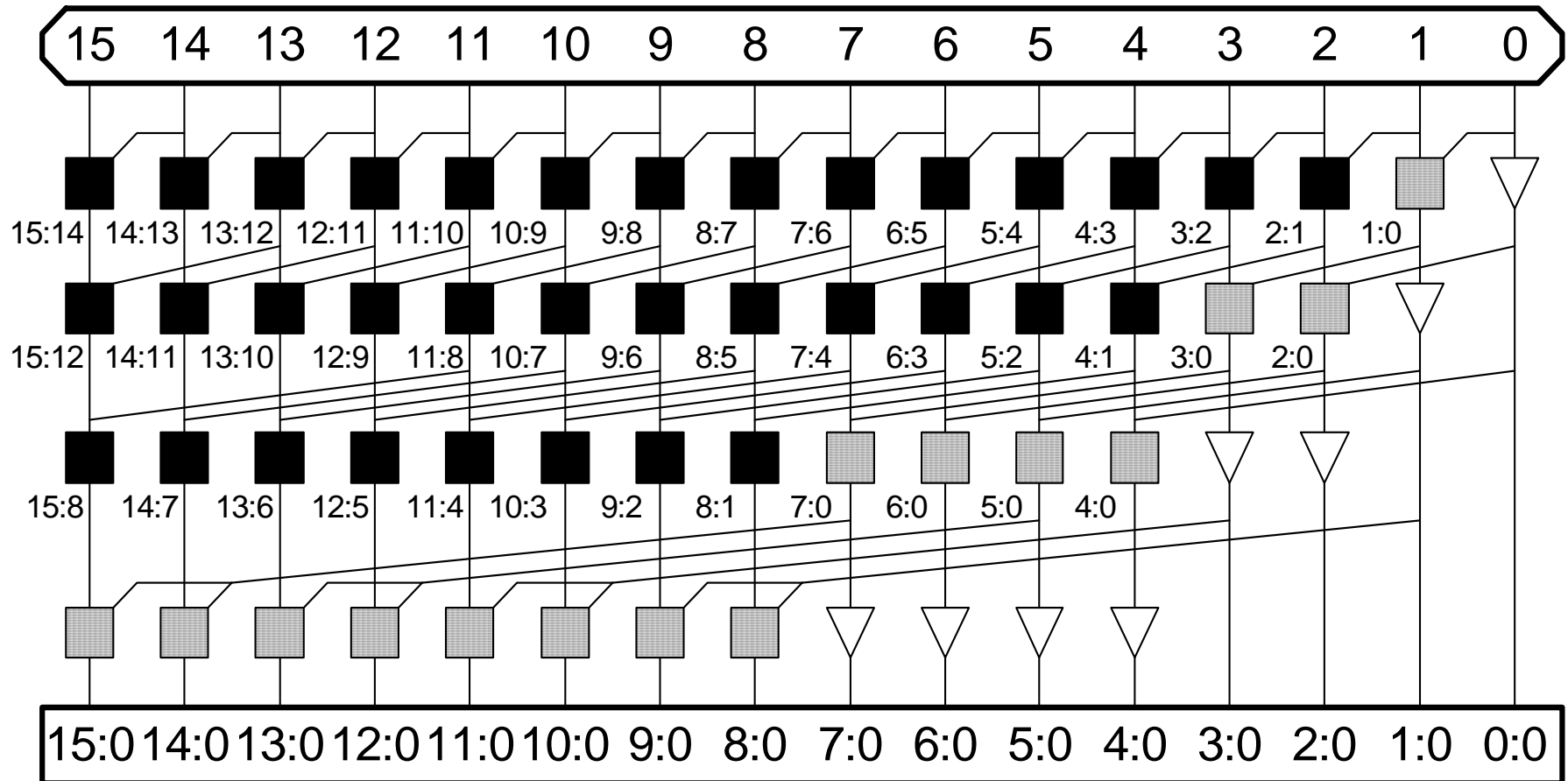


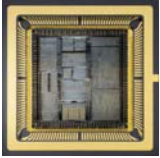
Han-Carlson



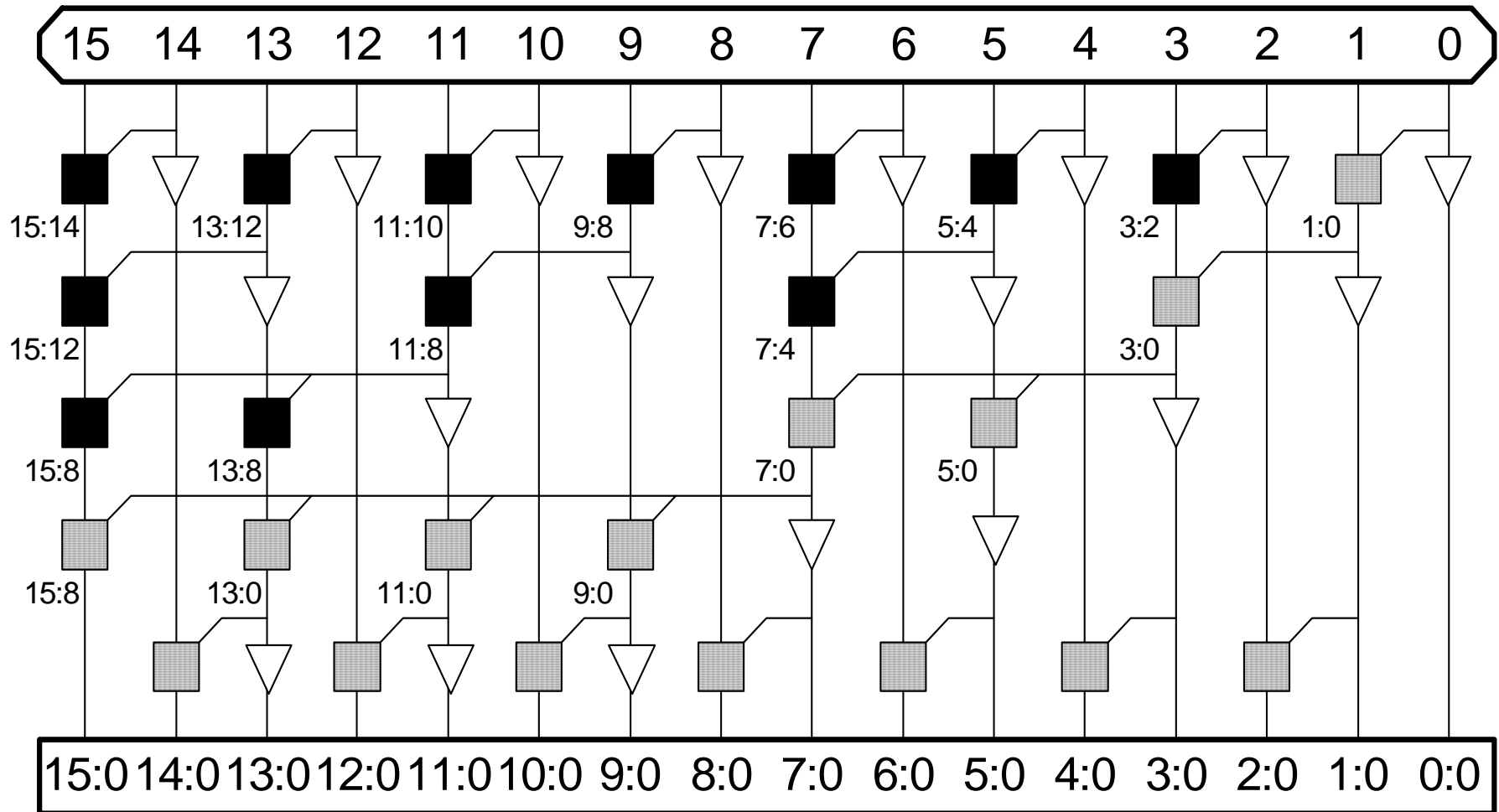


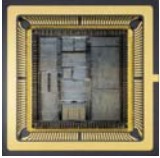
Knowles [2, 1, 1, 1]



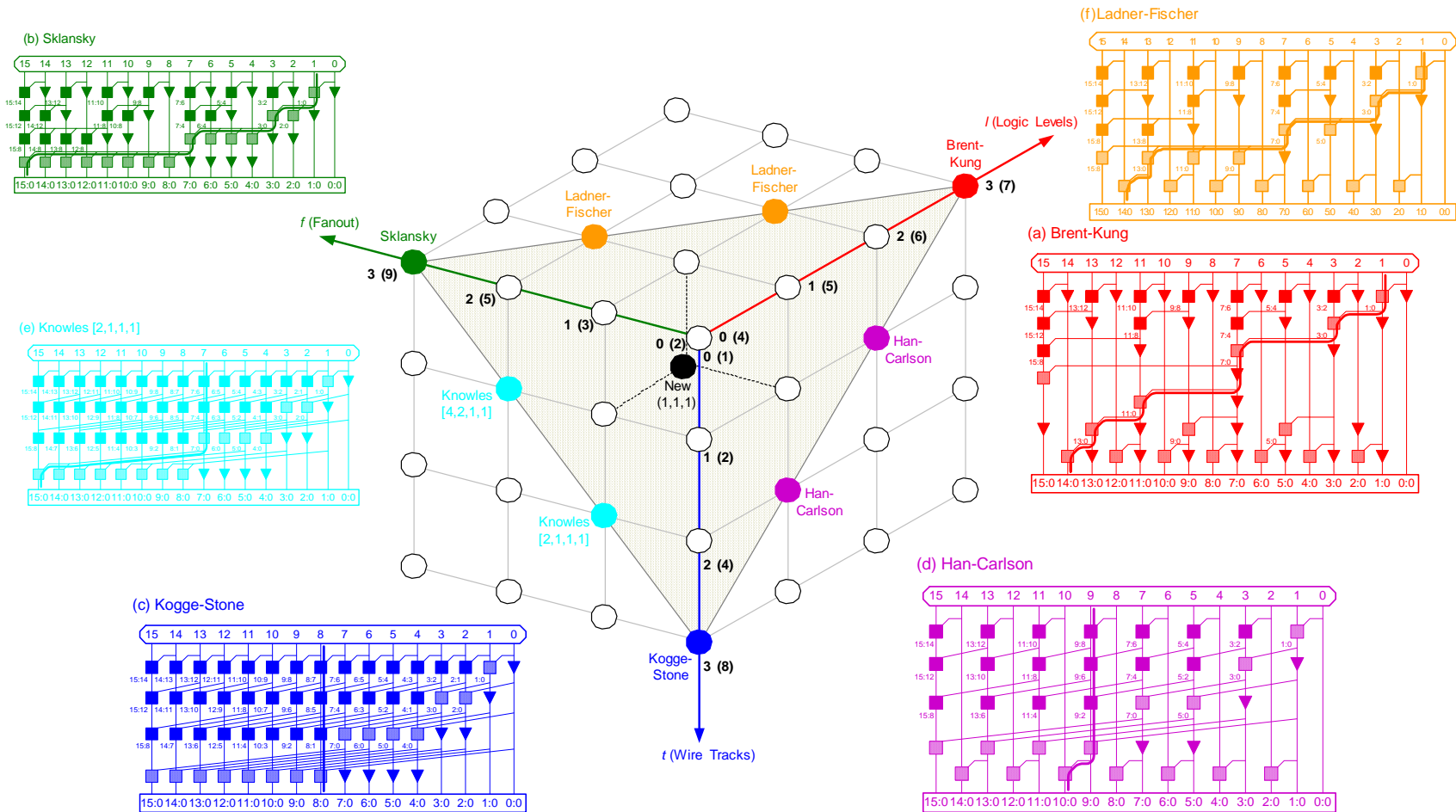


Ladner-Fischer





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	$N/2$	$N \log_2 N$