Very Large Scale Integration II - VLSI II

Adder Topologies

Gürer Özbek

ITU VLSI Laboratories

Istanbul Technical University

Outline

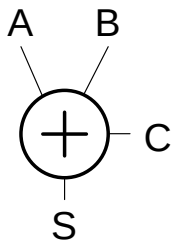
- Single Bit Addition
- Carry Propagate Adders
- PGK Representation & PG Diagram
- Tree Adders (Parallel Prefix Adders)

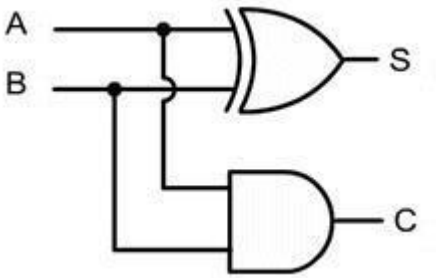
Adder Topologies

- Single Bit Addition
  - Half Adder
  - Full Adder
- Carry Propagate Adders
  - Carry Ripple (normal & inverse)
  - Carry Skip
  - Carry Select
  - Carry Lookahead
- Tree Adders (parallel prefix adders)
  - Brent Kung
  - Sklansky
  - Kogge-Stone
  - Ladner-Fischer
  - Knowles
  - Han-Carlson
  - Sparse Tree

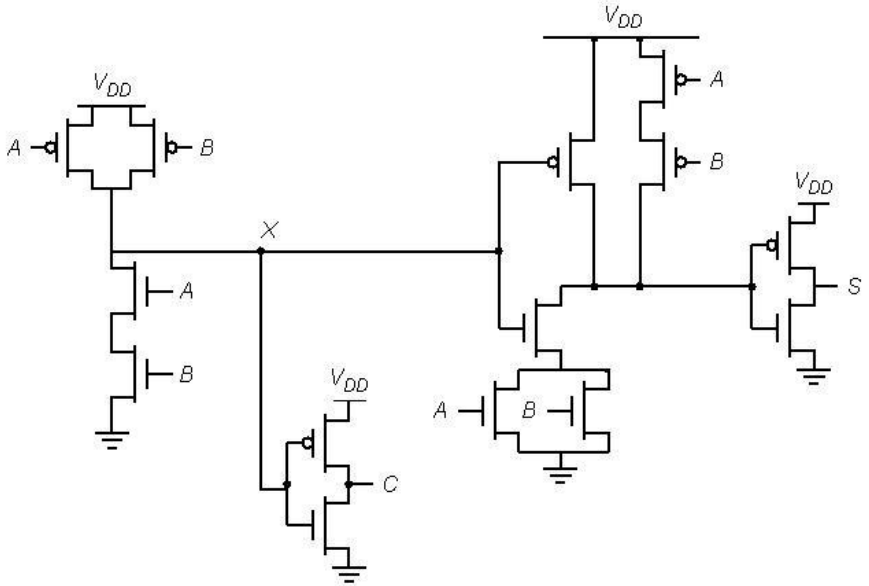
Single Bit Addition

- What's the deal?
  - All we want to do is add up a couple numbers…





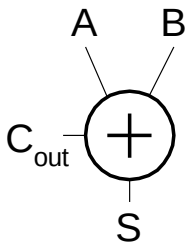
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



- A⊕B etc.

Half Adder

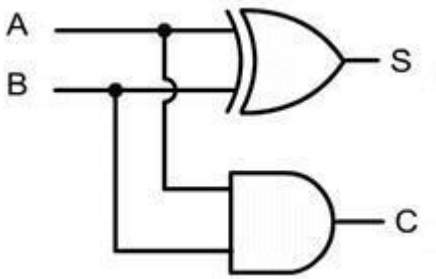
- 2 bit input, 2 bit output
- Used to build a Full Adder



$$S = A \oplus B$$

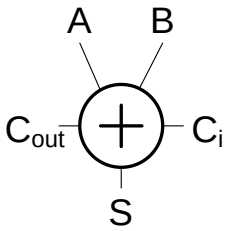
$$C_{out} = A.B$$

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



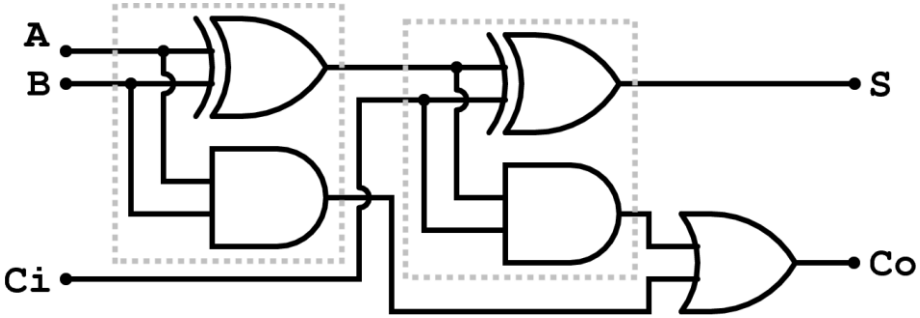
Full Adder

- Main element of n-bit adders
- Consists of 2 HAs



$$S = A \oplus B \oplus C_i$$

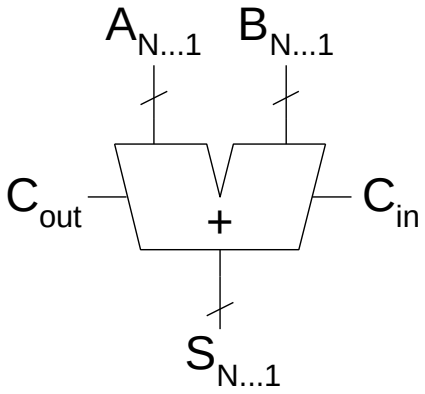
$$C_{out} = MAJ(A, B, C_i)$$

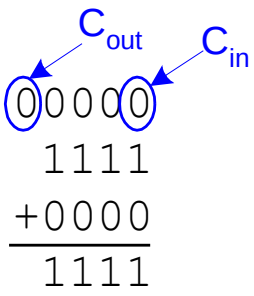


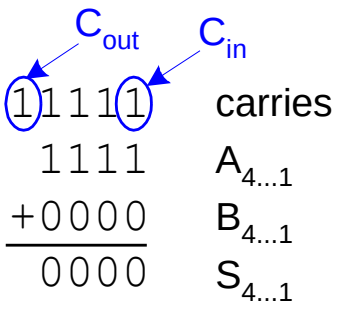
A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Carry Propagate Adders

- N-bit adder called as CPA
  - Each sum bit consists inf. of all previous carries
  - It's the main problem to calculate them all quickly

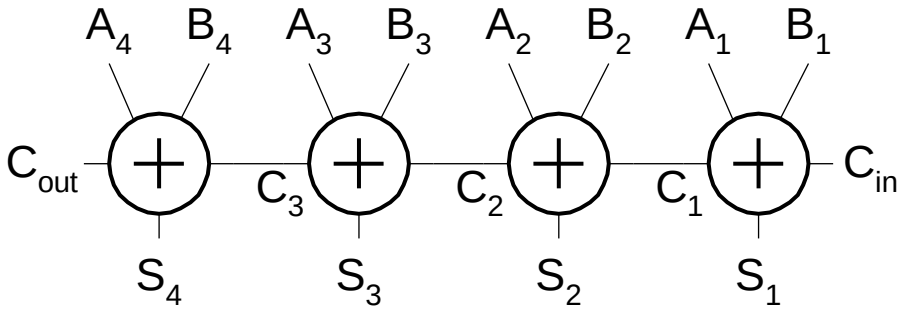






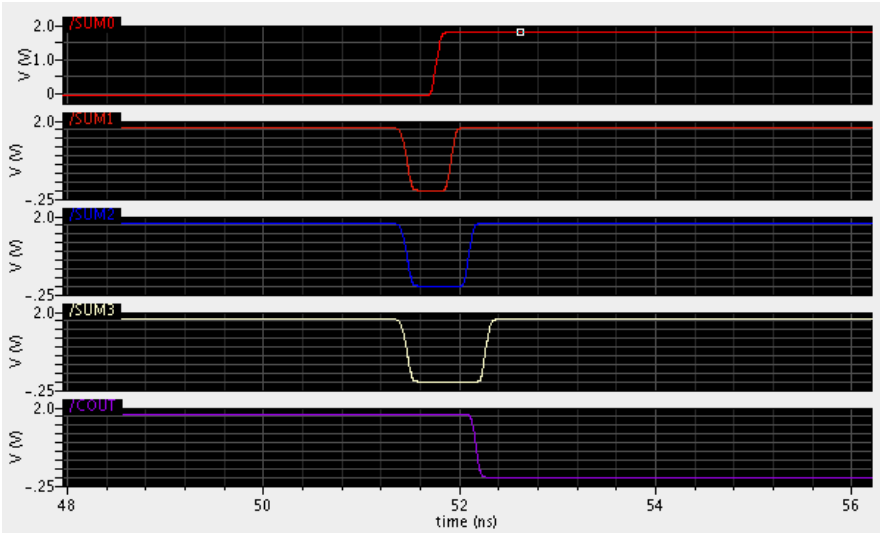
Carry Ripple Adder

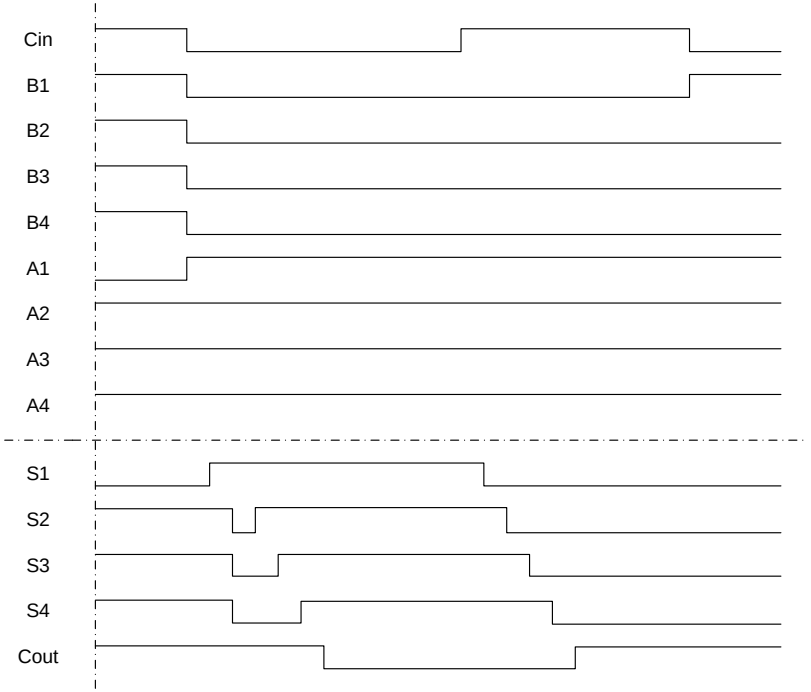
- Simplest Design: Cascaded FAs
  - Second area efficient of all ☺
  - Slowest of all ☹
  - Default topology to be synthesized



Carry Ripple Adder Delay

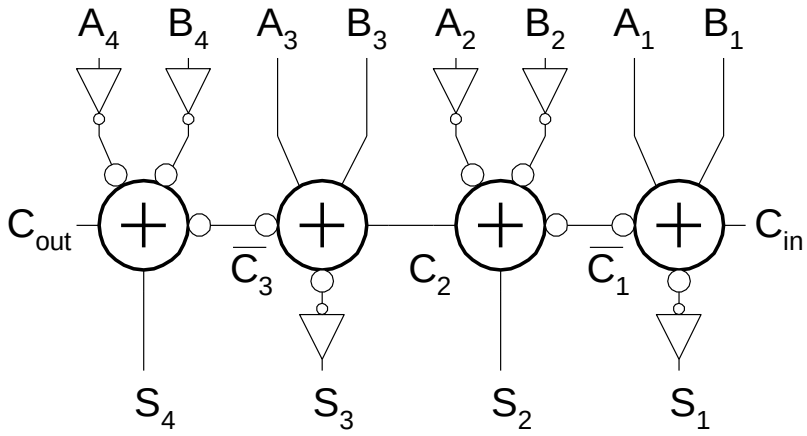
- Delay grows with O(N)
- Every FA waits for previous' output

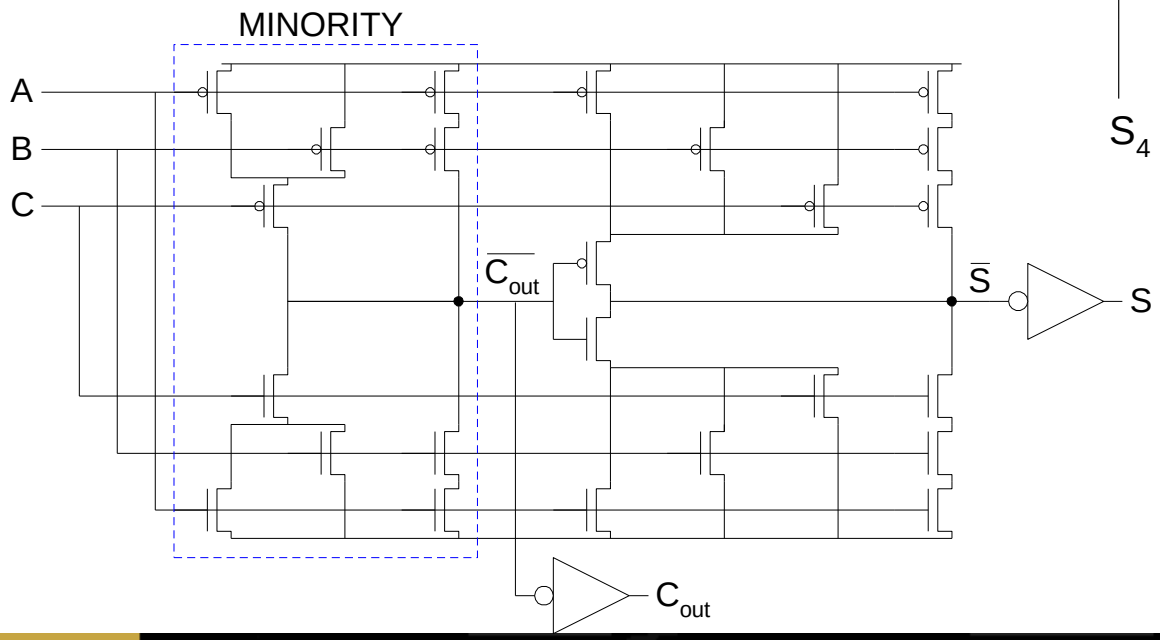




Inverse Carry Ripple Adder

- Uses inverting FAs
  - Most area efficient of all ☺
  - Second Slowest of all





Propagate, Generate and Kill the Carry

- Three operation can be defined to describe status of carry
  - Propagate: Previous carry is propagated to next bit
  - Generate: Generate a carry bit
  - Kill: Kill the previous carry

$$G_{i:i} \equiv G_i = A_i . B_i$$

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

$$K_{i:i} \equiv K_i = \overline{A_i + B_i}$$

A	B	P	G	K
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	0

Propagate and Generate the Carry

- Kill is not used mostly
- Carry Merge Tree (CM)

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

$$P_{i:j} = P_{i:k}.P_{k-1:j}$$

- Initial values

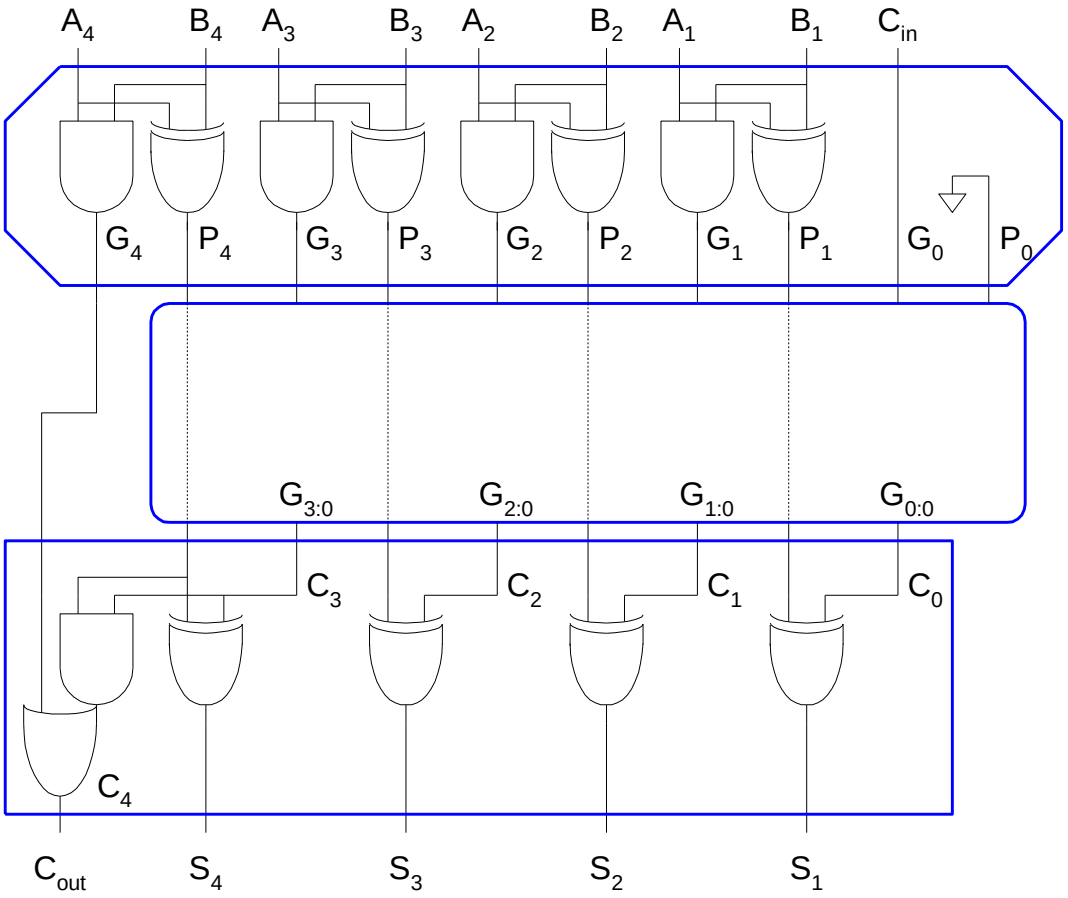
$$G_{0:0} \equiv G_0 = C_{in}$$

$$P_{0:0} \equiv P_0 = 0$$

- Final Sum

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

PG Diagram



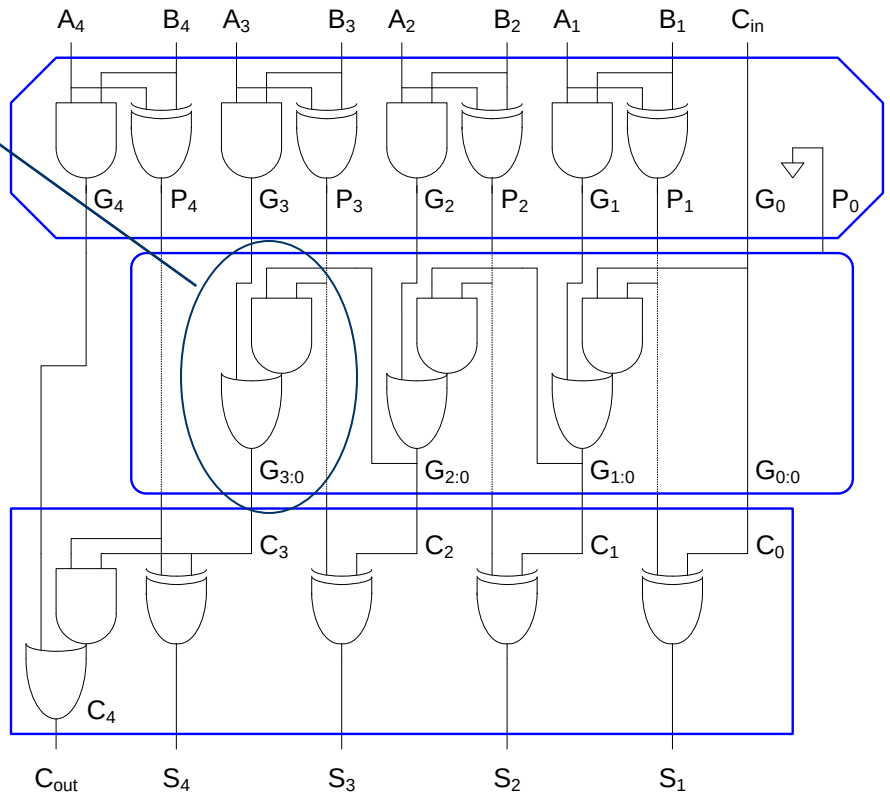
1: Bitwise PG logic

2: Group PG logic

3: Sum logic

Carry Ripple in PG Diagram 1

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



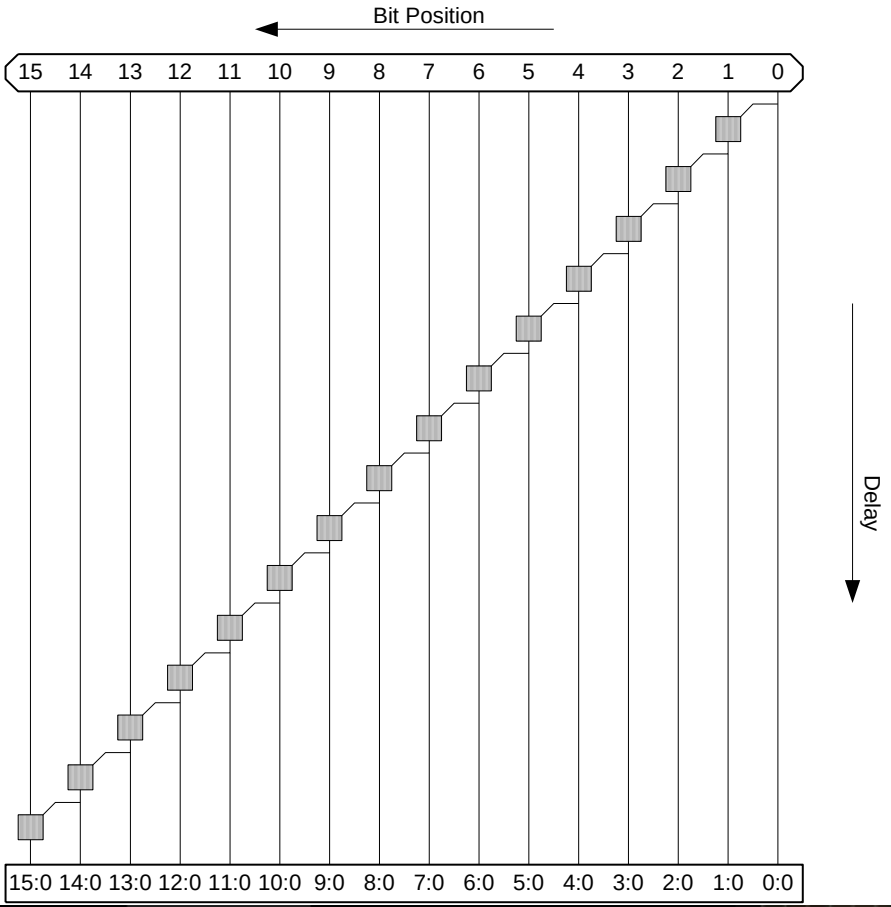
Carry Ripple in PG Diagram 2

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

1-bit prop/gen cell

delay of And/Or in grey cell

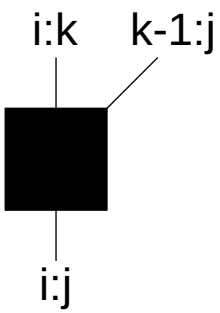
Final SUM bit xor



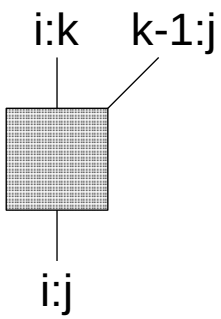
- Delay grows as O(N)

PG Diagram Notation

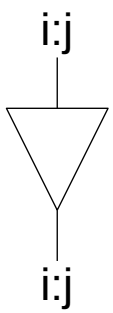
Black cell

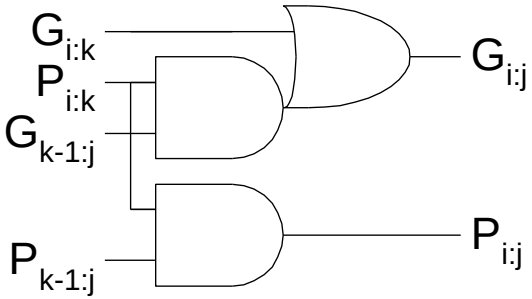


Gray cell

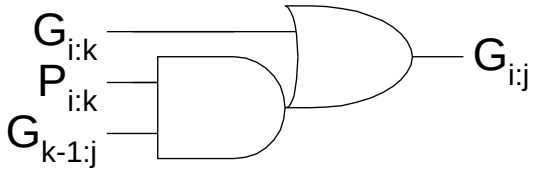


Buffer

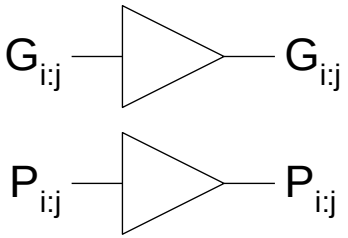




Both Gen/Prop



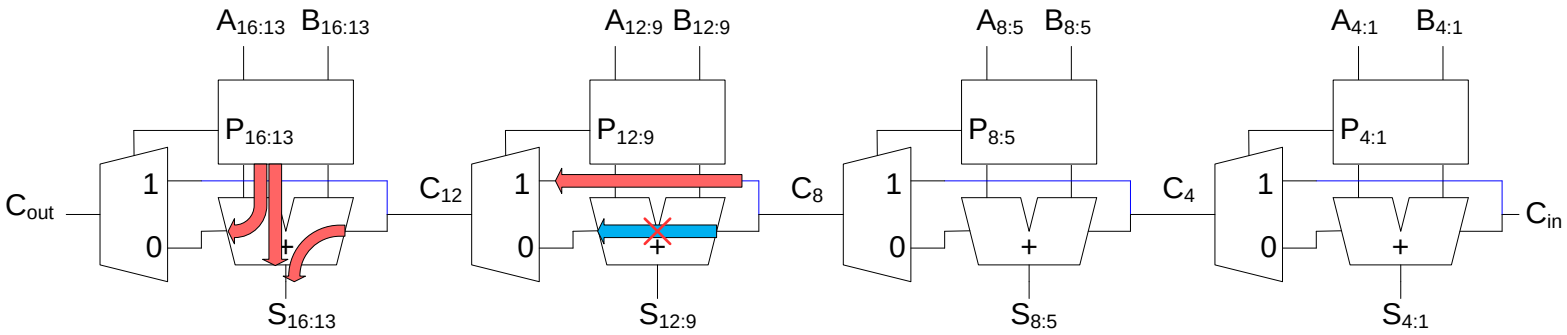
Generate only



Different load

Carry Skip Adder

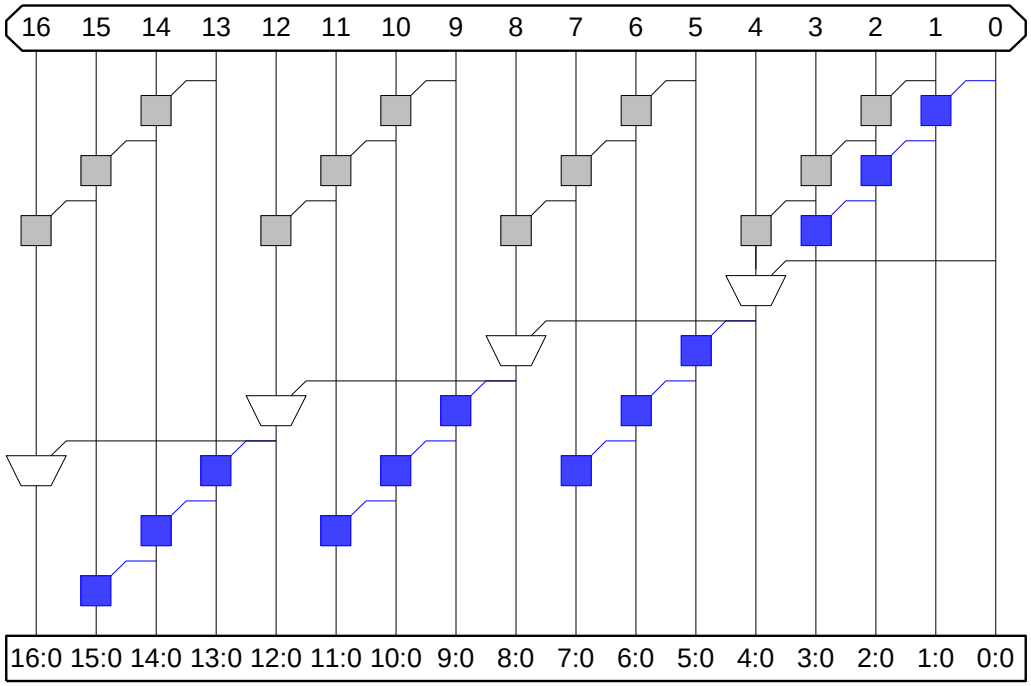
- Better delay growth rate is necessary
- Improves critical path delay



  - Red arrows: Allowed carry paths
  - Blue arrow: Non-allowed carry path

Carry Skip in PG Diagram

- For k n-bit groups (N = nk)
- Delay grows as O(√N)



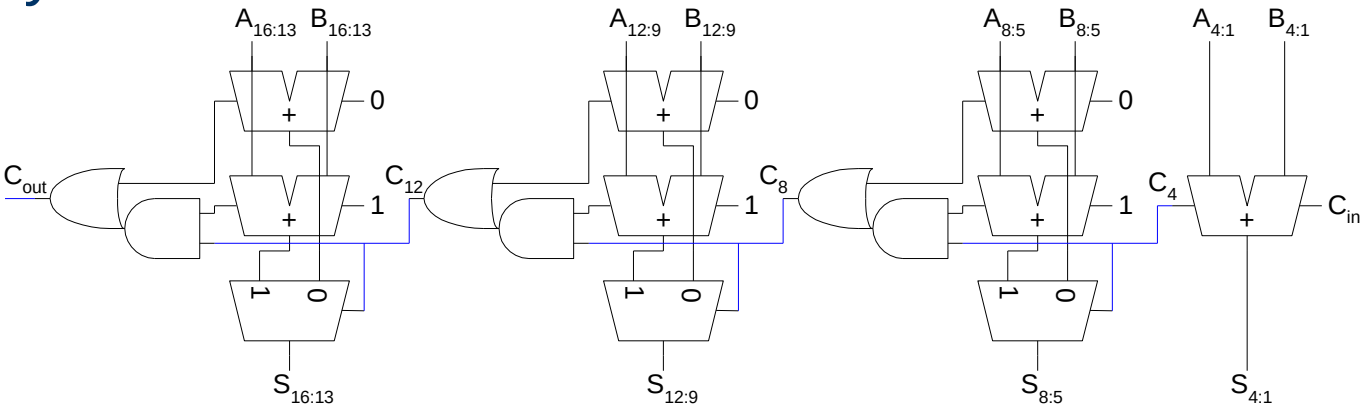
skip thru muxes

First & last group ripple

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

Carry Select Adder

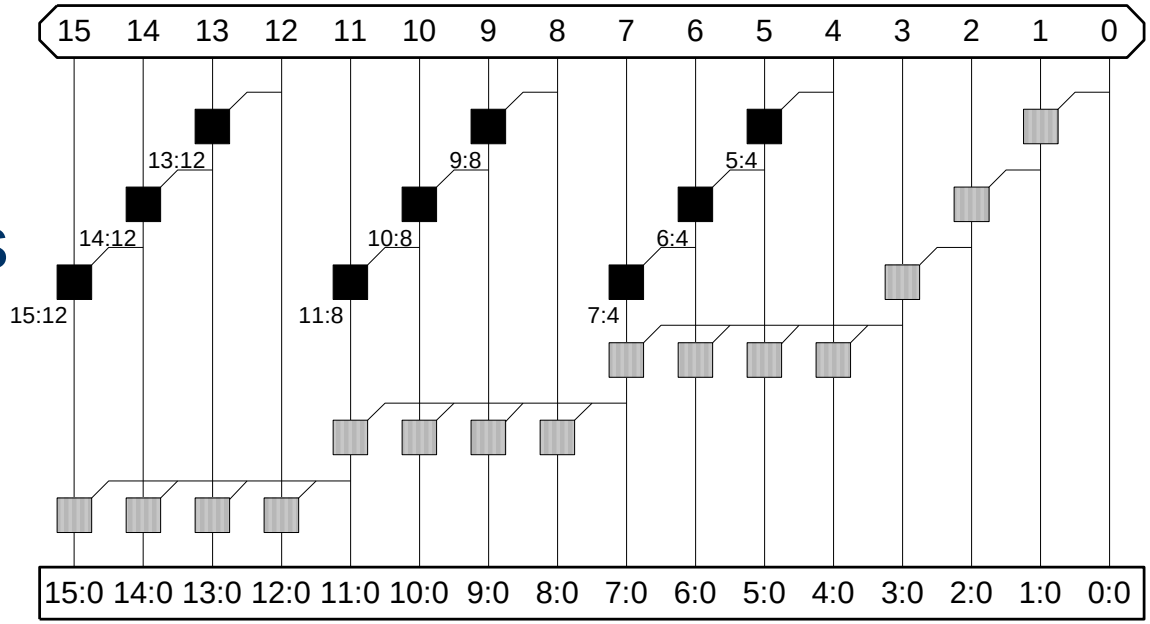
- Precomputes sum of n-bit groups for both carry conditions
- Final Mux selects the correct sum value when correct carry value arrives



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

Carry Select in PG Diagram

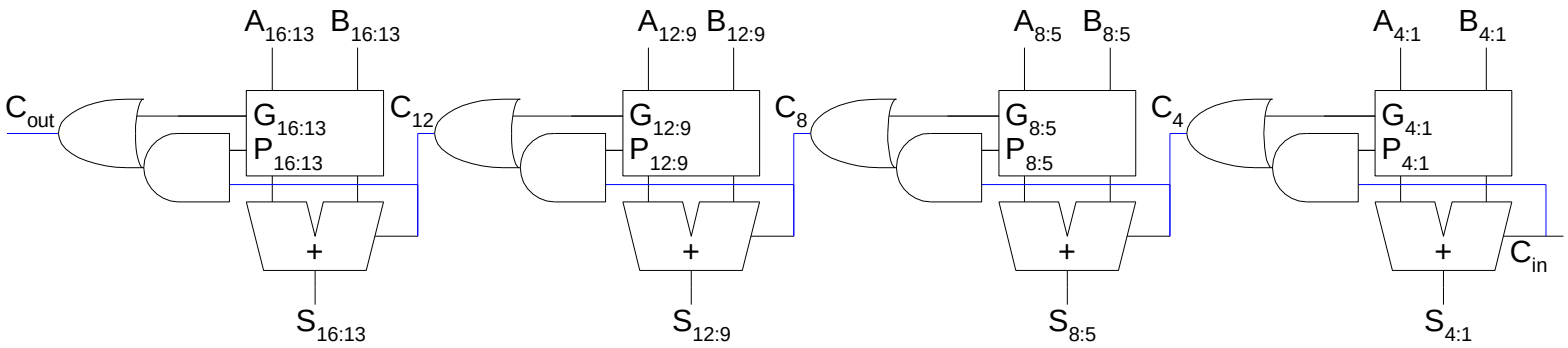
- Precomputes sum of n-bit groups for both carry conditions



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

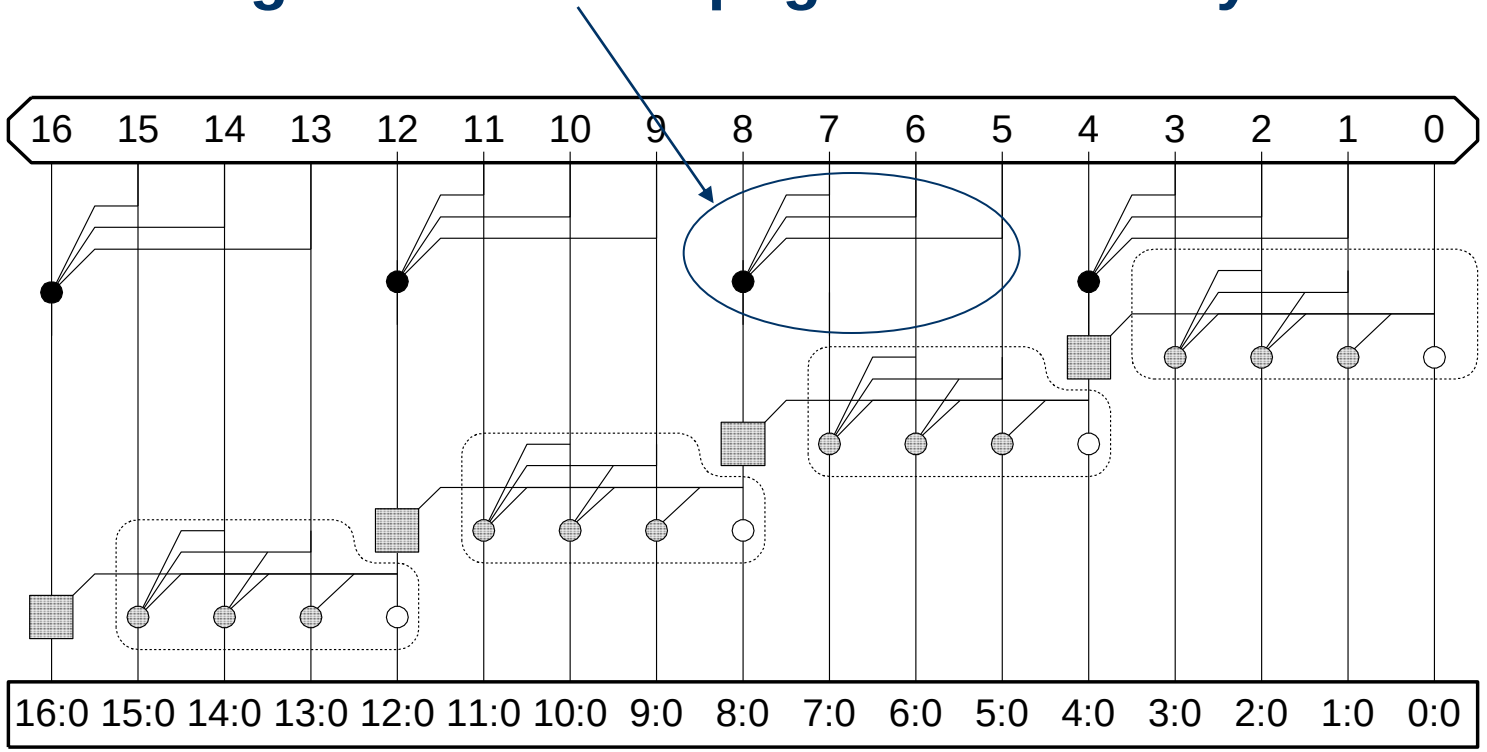
Carry Lookahead Adder

- Computes Generate bits in parallel
- Higher-valency cells are used



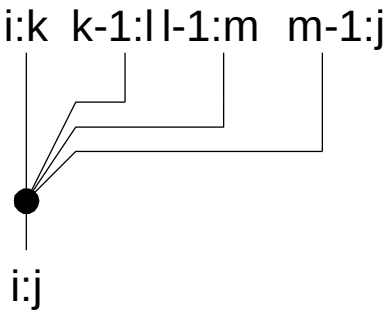
Carry Lookahead in PG Diagram

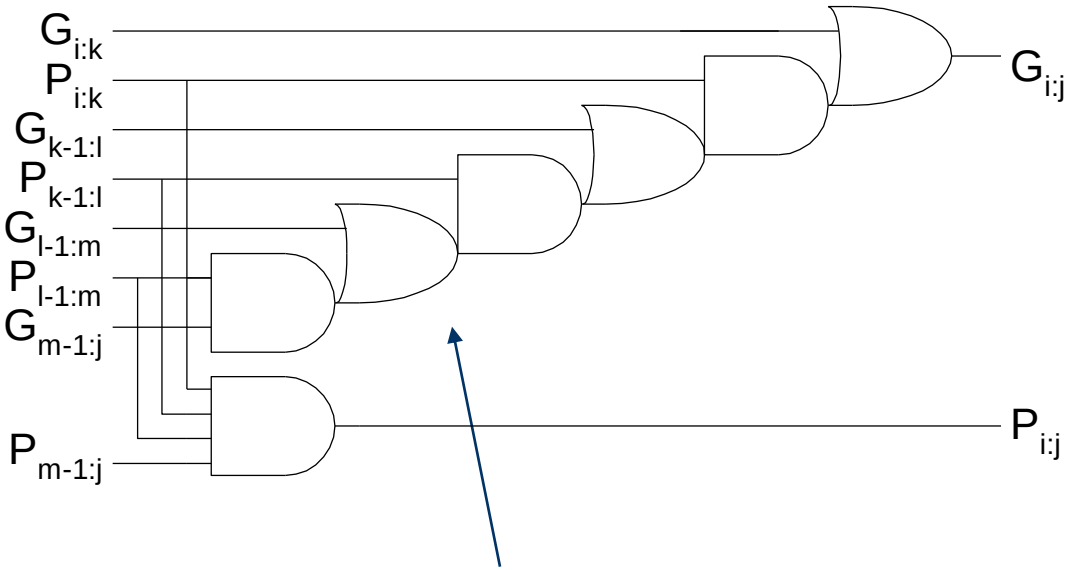
Collecting Generate/Propagate over many cells



Higher Valency Cells in CLA

- Difficult to design with static CMOS





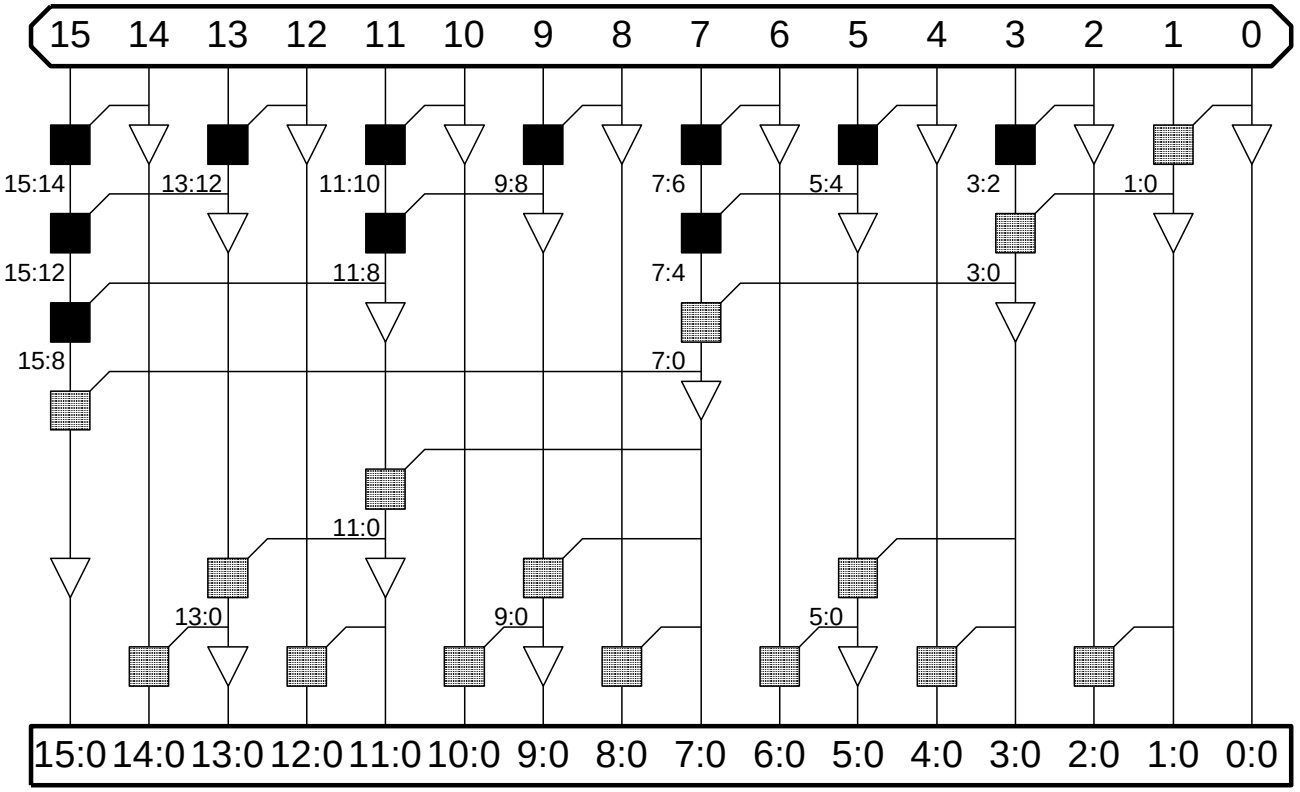
Recursive definition of Generate

Tree Adders

- Parallel PG calculation without linear propagation
- O(log N) delay
- Suitable for large-bit adders

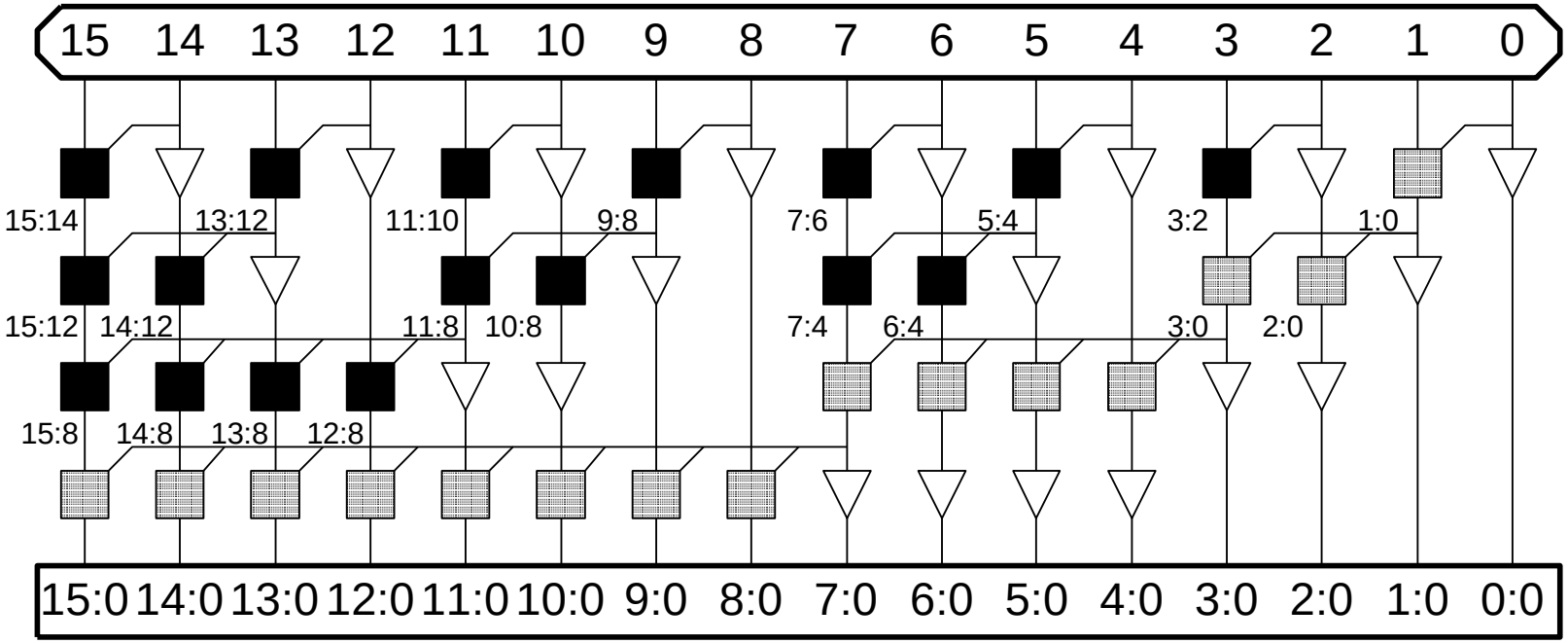
Brent-Kung

- Very First and Bad one



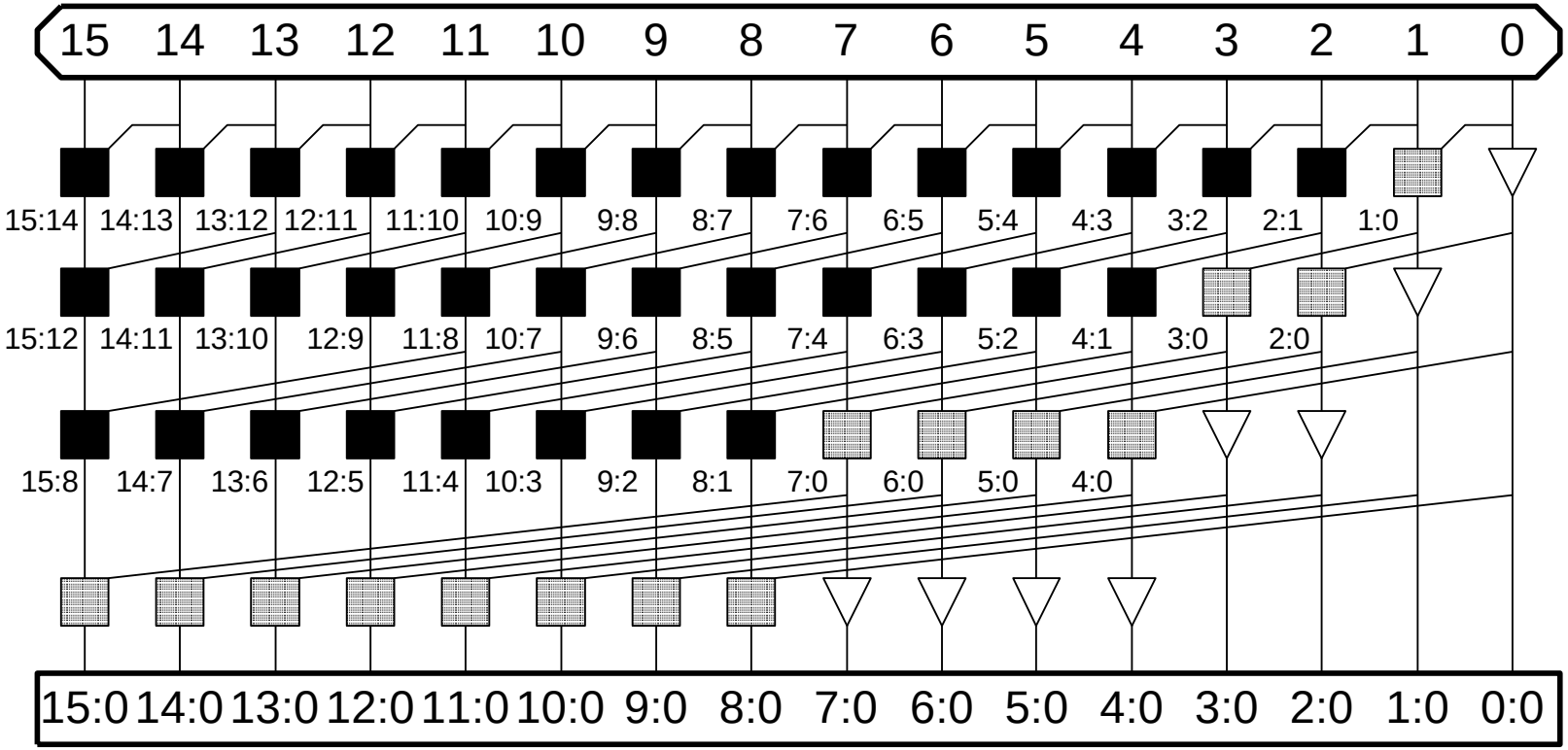
Sklansky

- Least Logic Levels
- Highest Fanout



Kogge-Stone

- Least Logic Levels
- Hard to P&R

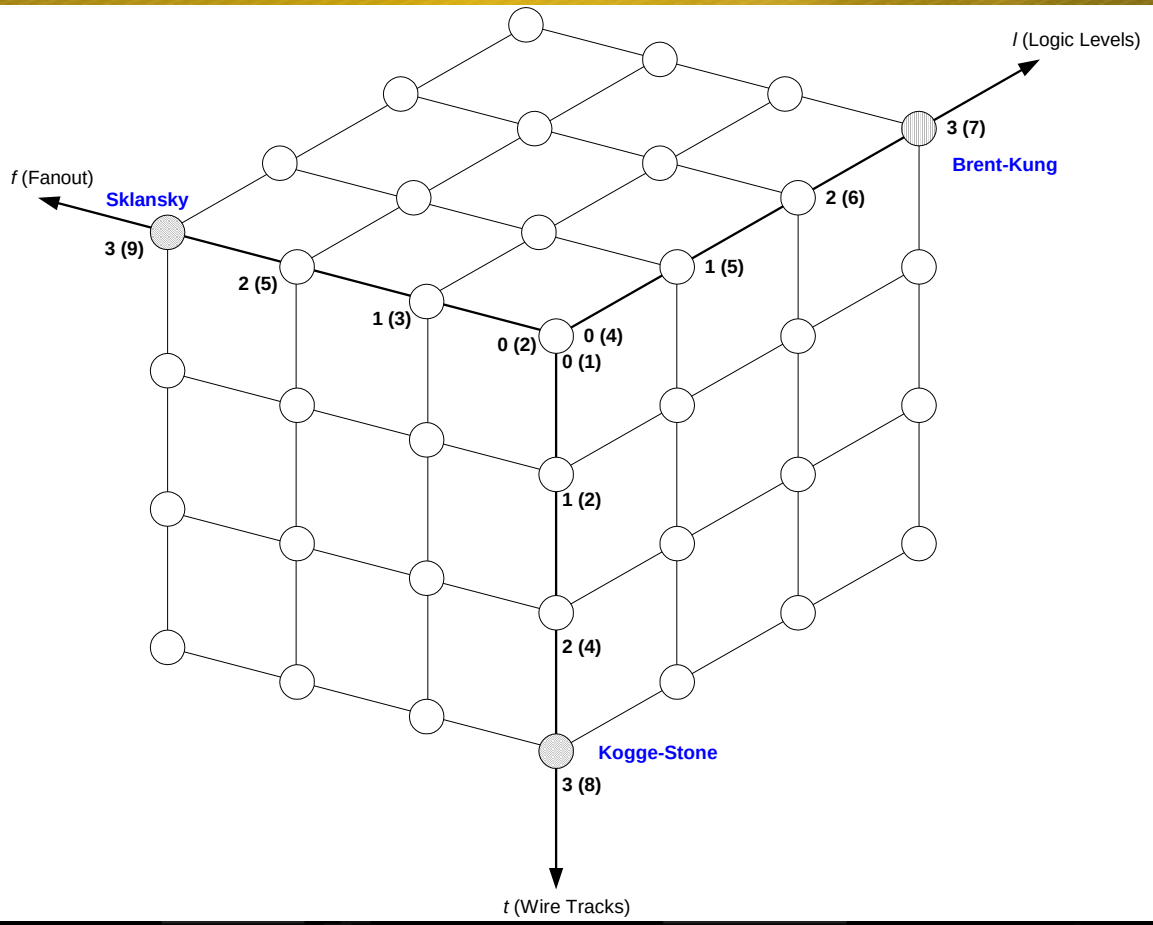


Tree Adder Taxonomy

- Ideal N-bit tree adder would have
  - $L = \log N$ logic levels
  - Fanout of 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

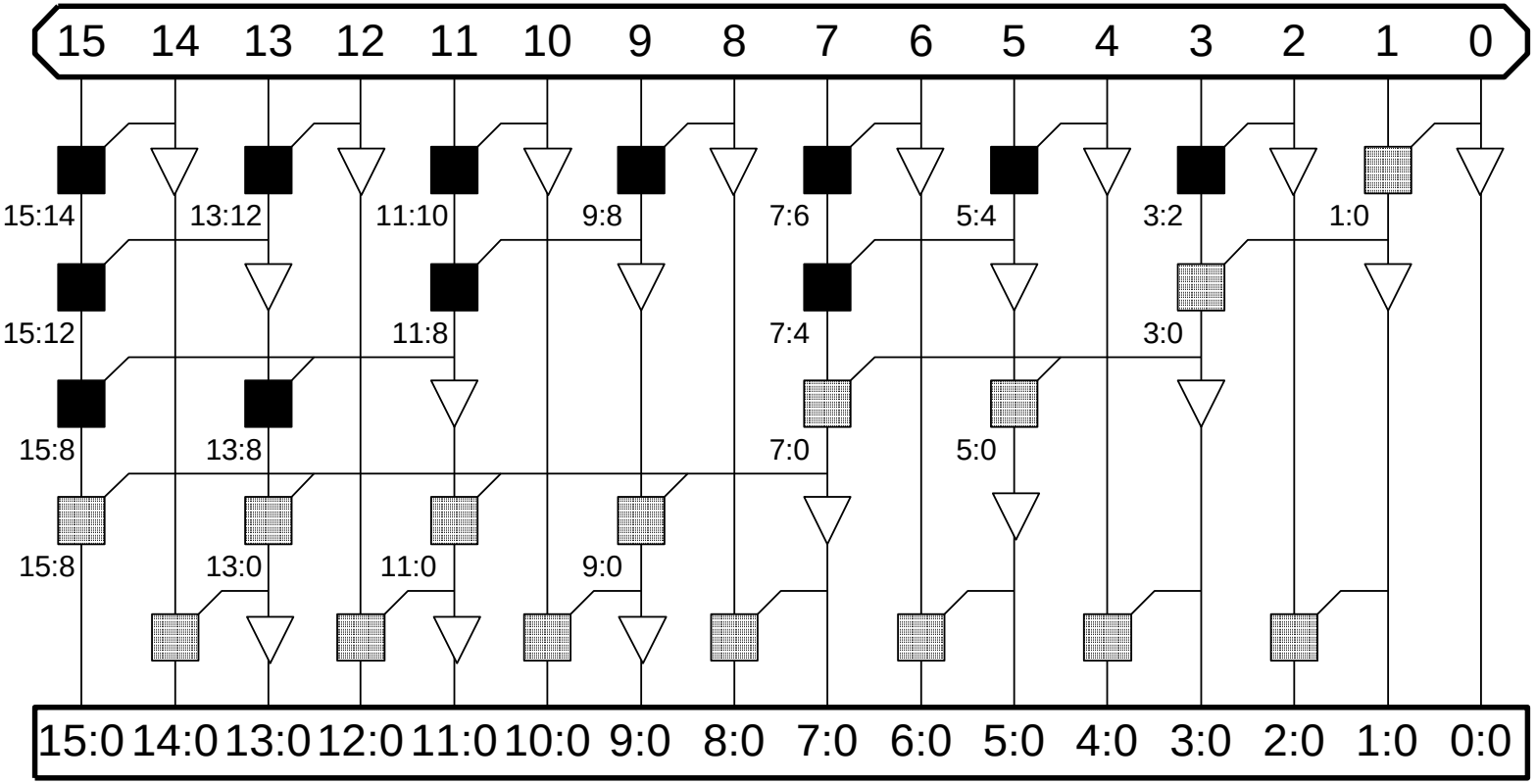
$$l + f + t = L\text{-}1$$

Tree Adder Taxonomy 2



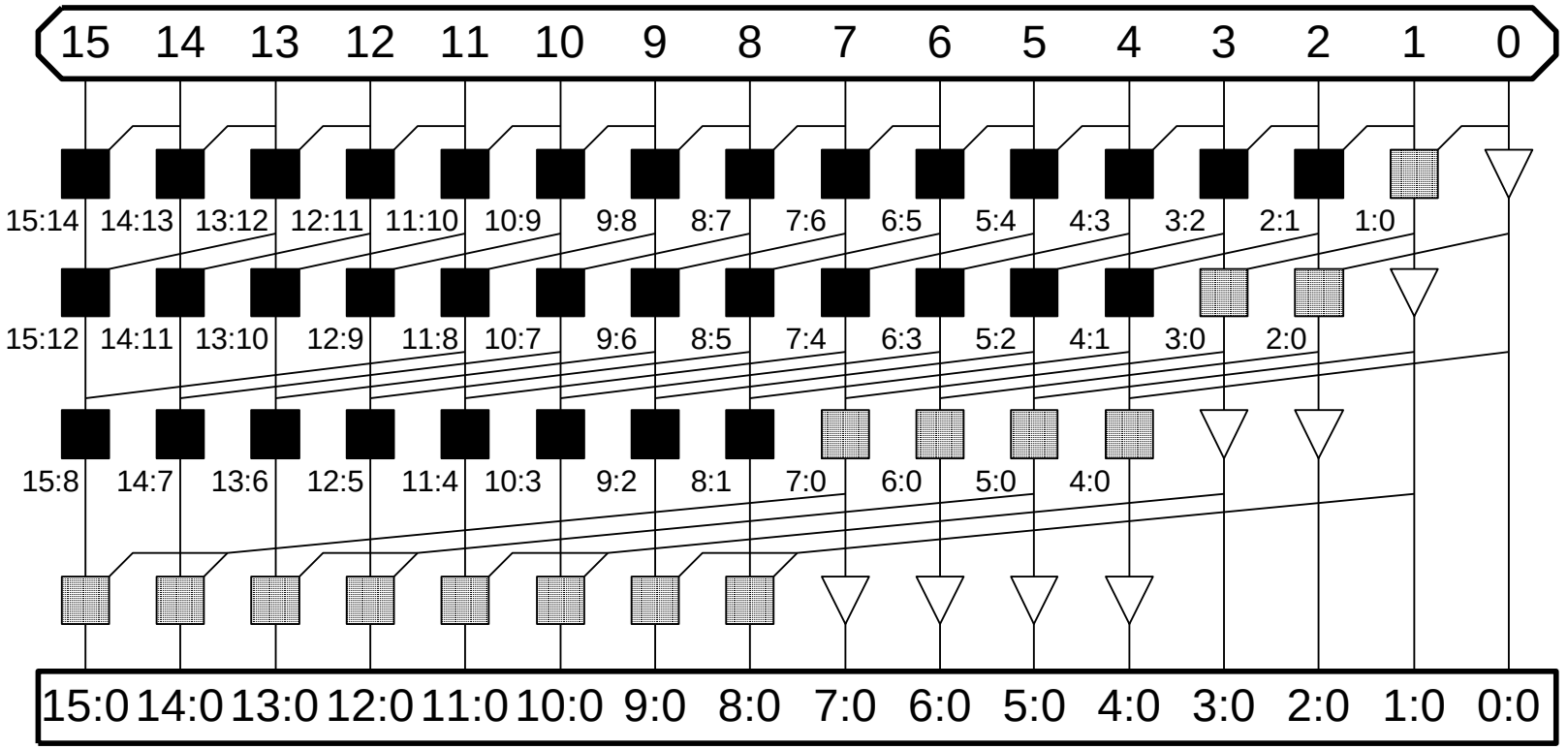
Ladner-Fischer

- A bit more logic levels
- High Fanout



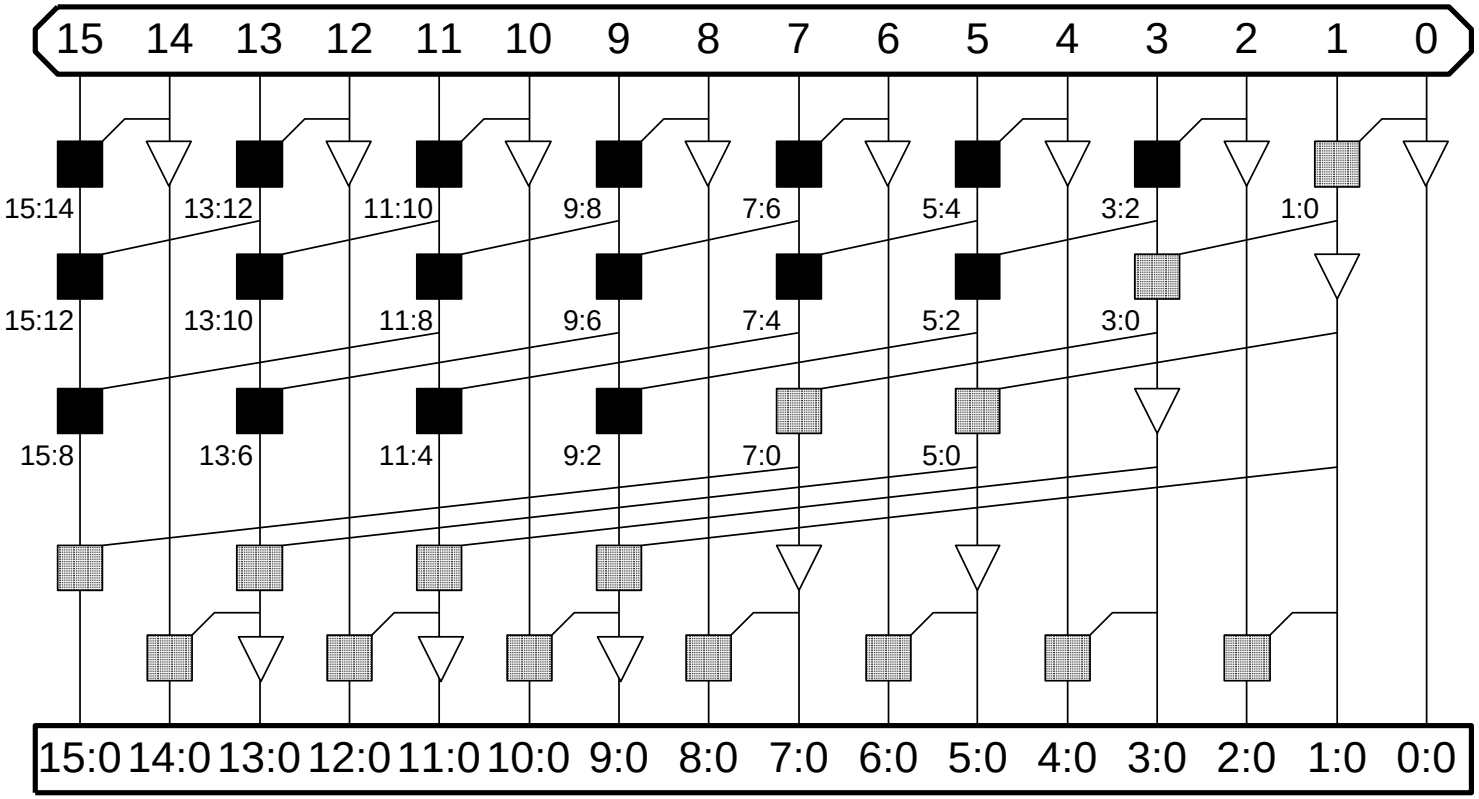
Knowles [2, 1, 1, 1]

- So many cells and wires
- Some Fanout



Han-Carlson

- A bit more logic levels
- Less cells

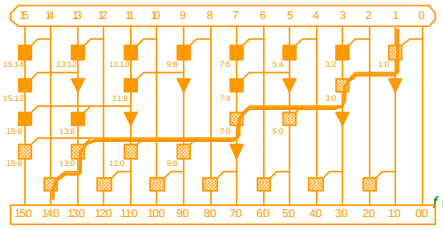


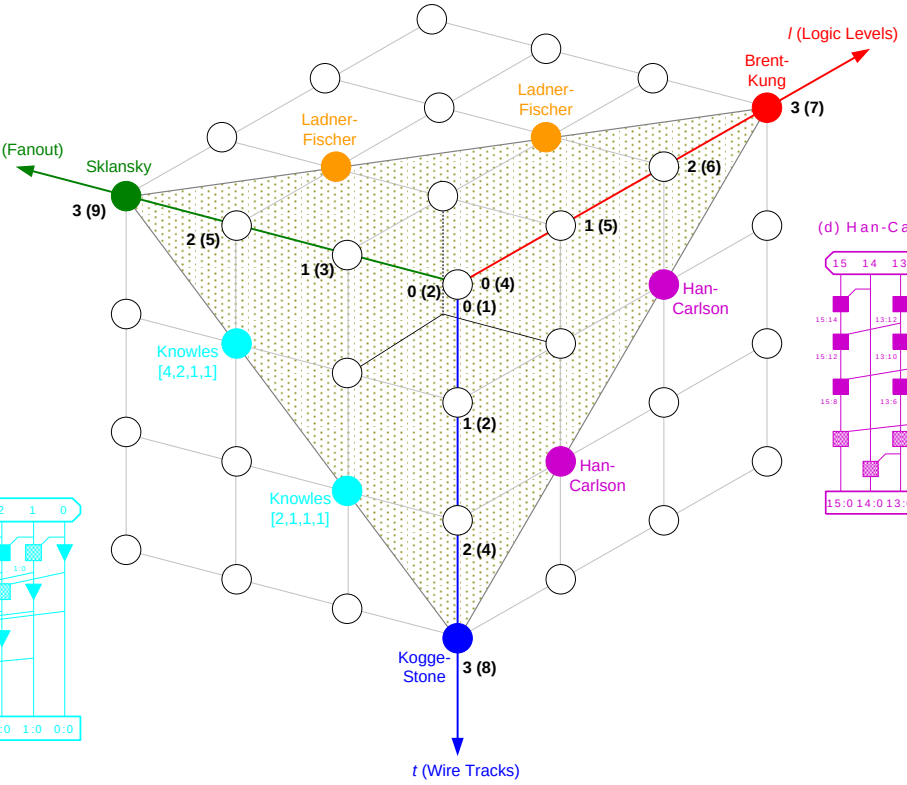
HOMEWORK

- 32-bit Sparse Tree Adder
  - Literature Search
    - What, When, Who, Where, Why, How
  - PG Diagram
    - Black cells, grey cells, buffers, muxes etc.
  - Gate Level Schematic
    - One per group
  - Delay Model wrt gate delays
    - t_{sparse}=…
  - 1 week

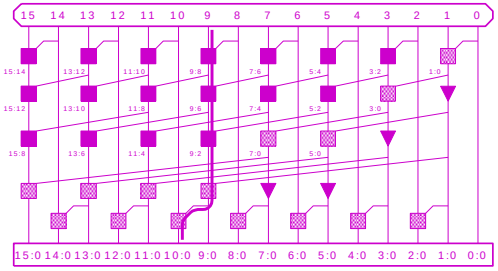
Tree Adder Taxonomy 3

(f)Ladner-Fischer

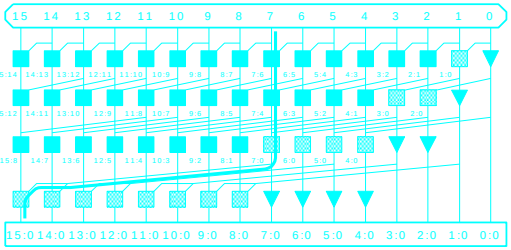




(d) Han-Carlson



(e) Knowles [2,1,1,1]



Summary

- Adders with Area-Power-Delay Tradeoffs

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-Sel. 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	$N\log_2 N$

References

- http://bwrc.eecs.berkeley.edu/icbook/Slides/chapter11.ppt
- http://www.cmosvlsi.com/lect11.pdf
- http://www.eng.utah.edu/~cs5830/Slides/addersx2.pdf
- Knowles, S. (1999) A Family of Adders