EE141-Spring 2010 Digital Integrated Circuits

Lecture 23
Adders/Multipliers

Administrativia

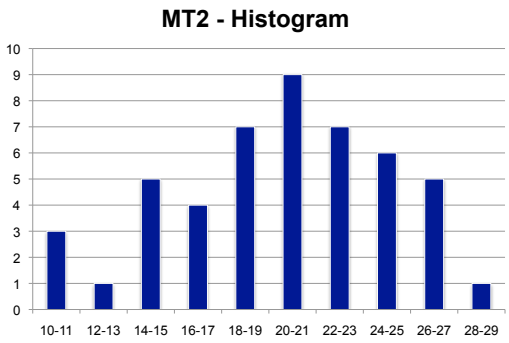
- Midterm 2 graded – More later
- Project Phase 2 Due TODAY

Class Material

- Last lecture
  - Energy
- Today's lecture
  - Adders, Multipliers
- Reading (Ch 11)

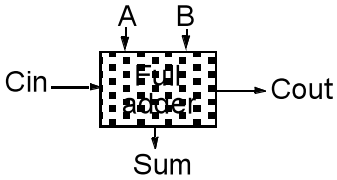
Midterm 2

- Grading Results (on 34)
  - Max: 28.3
  - Avg: 20.44
  - Median: 21.3
  - StdDev: 4.59



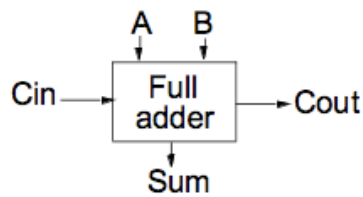
Adders

Full-Adder



A	B	C_i	S	C_o	*Carry status*	
0	0	0	0	0	delete	(kill)
0	0	1	1	0	delete	(kill)
0	1	0	1	0	propagate	
0	1	1	0	1	propagate	
1	0	0	1	0	propagate	
1	0	1	0	1	propagate	
1	1	0	0	1	generate	
1	1	1	1	1	generate	

The Binary Adder



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

$$= A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i$$

$$C_o = AB + BC_i + AC_i$$

Express Sum and Carry as a function of P, G, K

Define 3 new variable which ONLY depend on A, B

Generate (G) = AB

Propagate (P) = A ⊕ B

Kill = A B

$$C_o(G,P) = G + PC_i$$

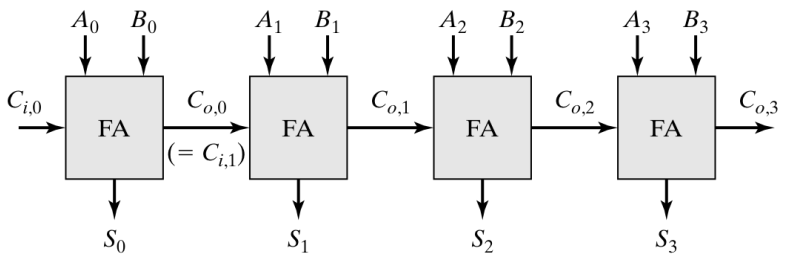
$$S(G,P) = P \oplus C_i$$

Can also derive expressions for *S* and C_o based on *K* and *P*

Note that we will sometimes use an alternate definition for

Propagate (P) = A + B

Simplest Adder: Ripple-Carry



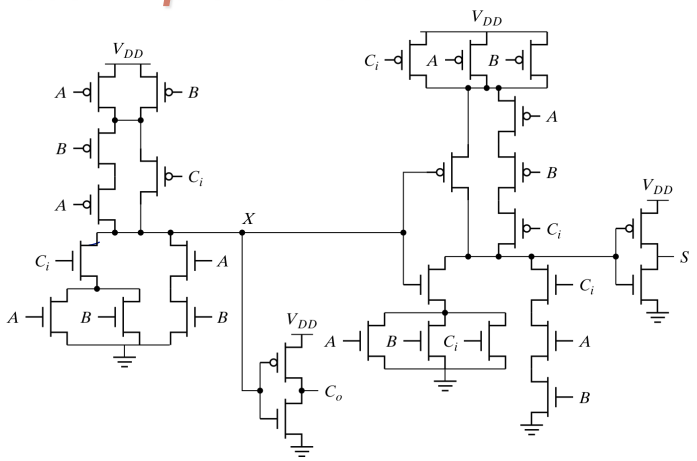
Worst case delay linear with the number of bits

$t_d = O(N)$

$t_{adder} = (N\text{-}1)t_{carry} + t_{sum}$

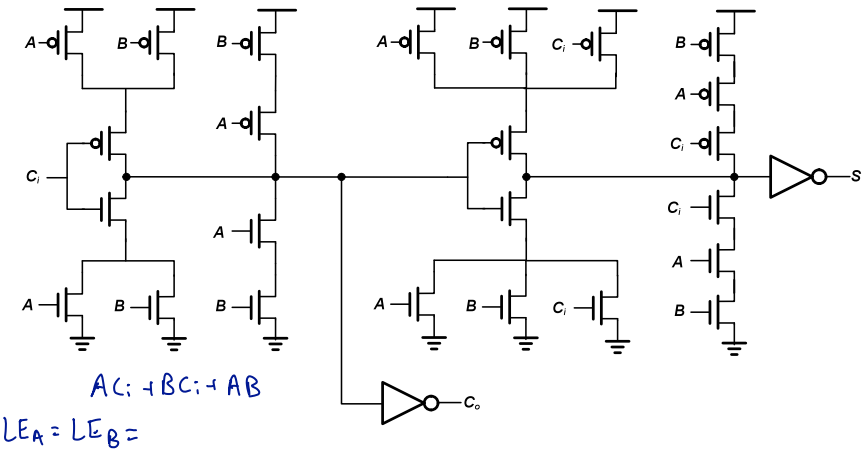
Goal: Make the fastest possible carry path circuit

Complementary Static CMOS Full Adder: "Direct" Implementation



28 Transistors

Complementary Static CMOS Full Adder



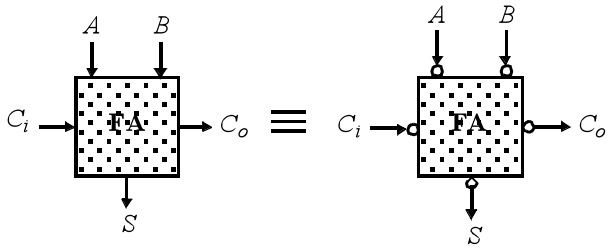
$AC_i + BC_i + AB$

$LE_A = LE_B =$

$LE_{C_i} =$

28 Transistors

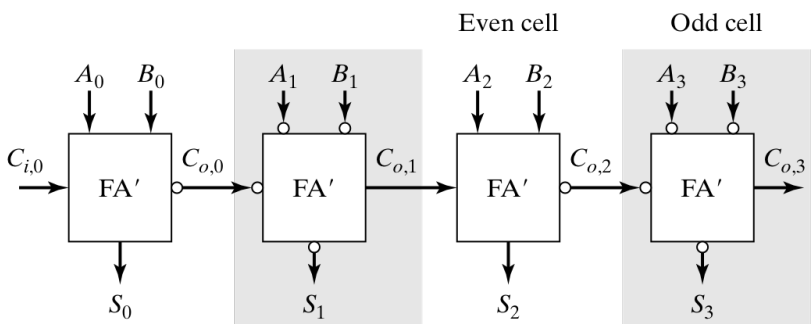
Inversion Property



$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\bar{C}_i)$$

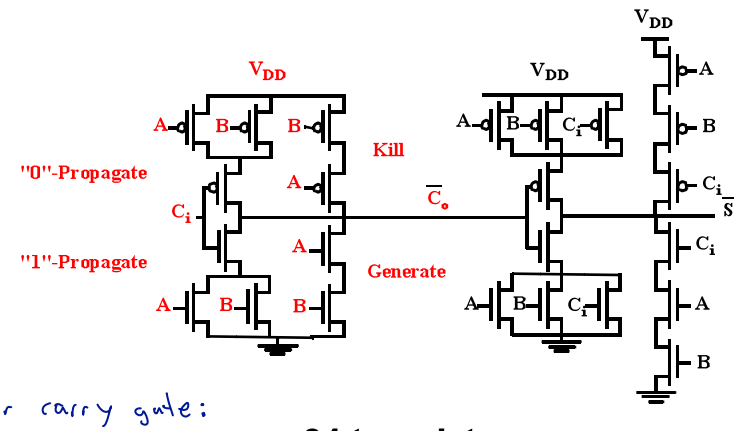
$$\bar{C}_o(A,B,C_i) = C_o(\bar{A},\bar{B},\bar{C}_i)$$

Minimize Critical Path by Reducing Inverting Stages



Exploit Inversion Property

A Better Structure: The Mirror Adder



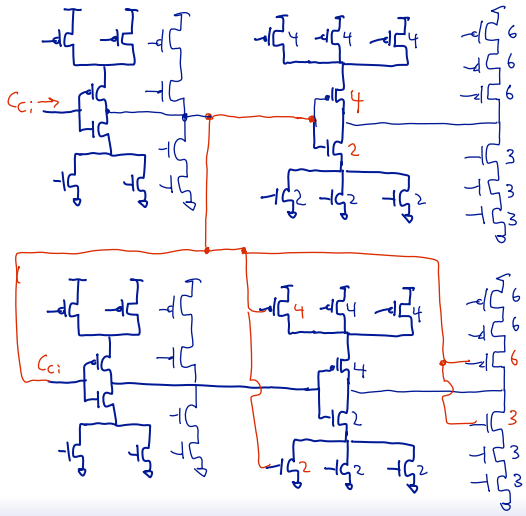
For carry gate:
$LE_{C_i} = 2$

24 transistors

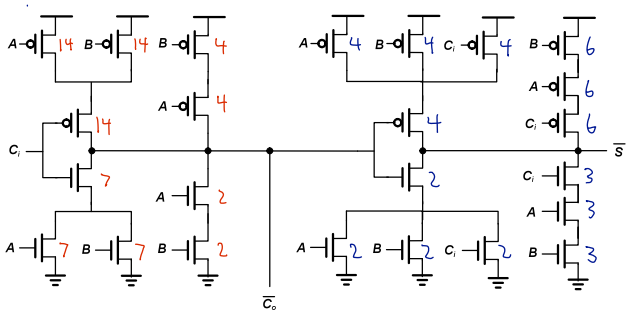
Sizing the Mirror Adder: Fanout

- Since LE of carry gate is 2, want *f* of 2 to get EF of 4
- Use min. size sum gates to reduce load on carry.
- Total load on carry gate is:

$C_{load} = C_{Ci} + (6+6+9)$
$C_{load} = 2C_{Ci}$



Sizing the Mirror Adder



- $C_{load} = C_{Ci} + (6+6+9) = 2C_{Ci}$

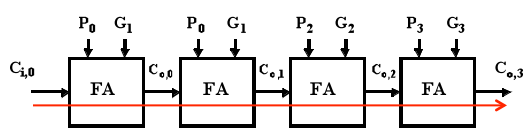
  → $C_{Ci} = 21$

- Minimum size G and K stacks to reduce diffusion loading

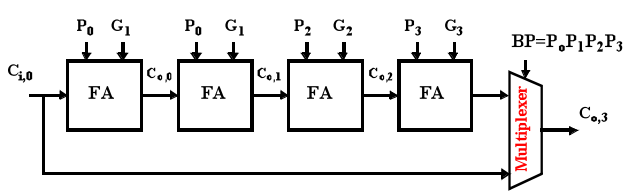
Mirror Adder Summary

- The NMOS and PMOS chains are completely symmetrical. Maximum of two series transistors in the carry-generation gate.
- When laying out the cell, the most critical issue is the minimization of the capacitance at node C_o. Reduction of the diffusion capacitances is particularly important.
- Carry signals are critical - transistors connected to C_i are placed closest to the output.
- Only the transistors in the (propagate) carry chain have to be optimized for speed. All transistors in the sum stage can be minimal size.

Carry-Bypass Adder

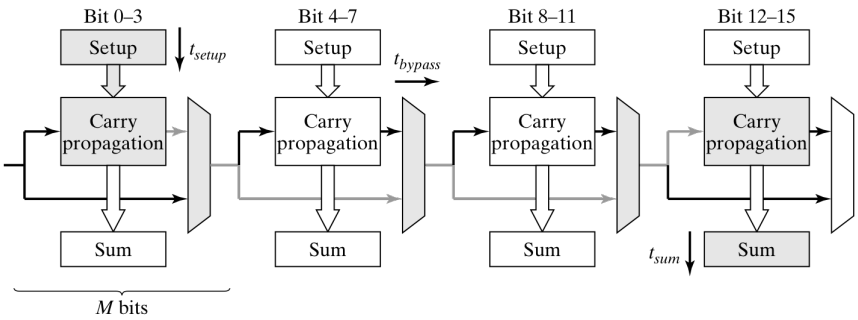


Also called Carry-Skip



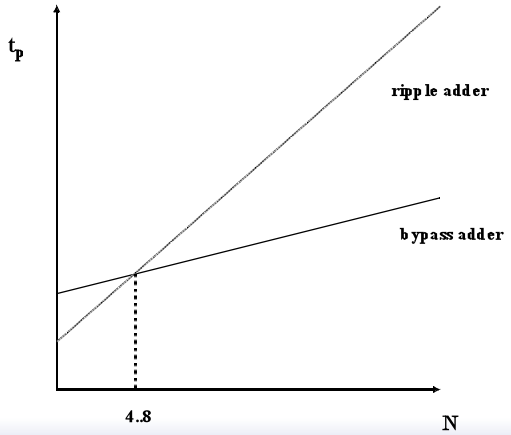
Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else "kill" or "generate".

Carry-Bypass Adder (cont.)

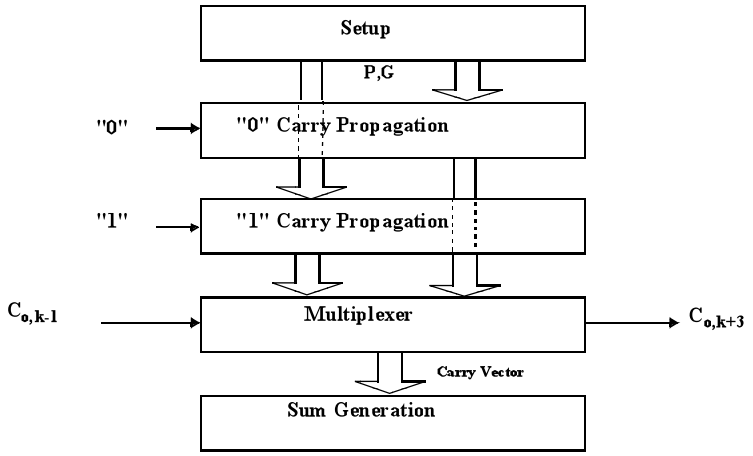


$$t_{adder} = t_{setup} + (M\text{-}1)t_{carry} + (N/M\text{-}1)t_{bypass} + (M\text{-}1)t_{carry} + t_{sum}$$

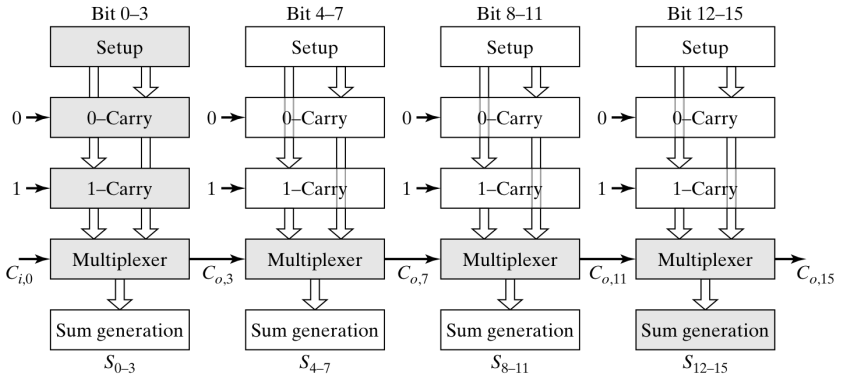
Carry Ripple versus Carry Bypass



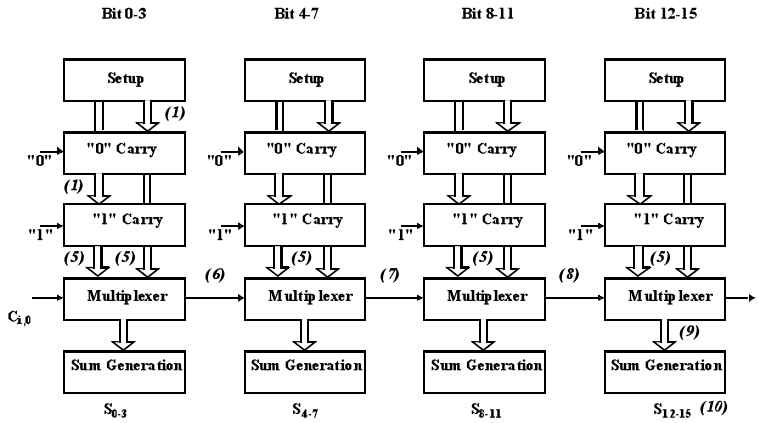
Carry-Select Adder



Carry Select Adder: Critical Path

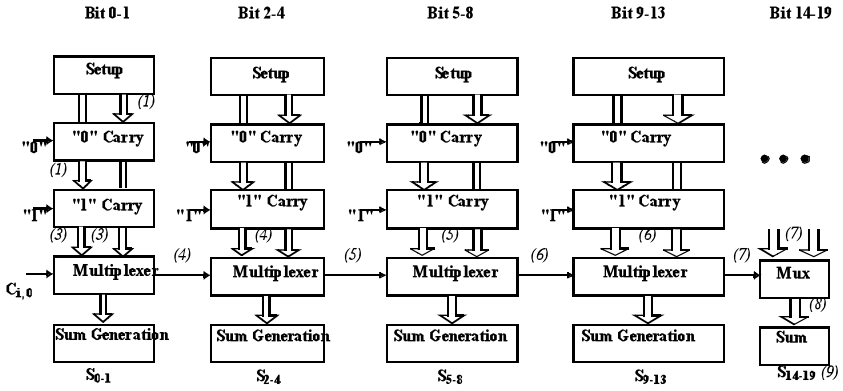


Linear Carry Select



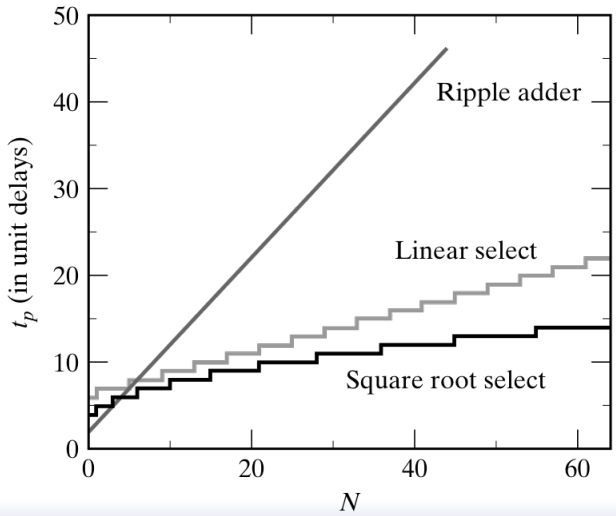
$$t_{add} = t_{setup} + \left(\frac{N}{M}\right) t_{carry} + M t_{mux} + t_{sum}$$

Square Root Carry Select



$$t_{add} = t_{setup} + M t_{carry} + (\sqrt{2N}) t_{mux} + t_{sum}$$

Adder Delays - Comparison

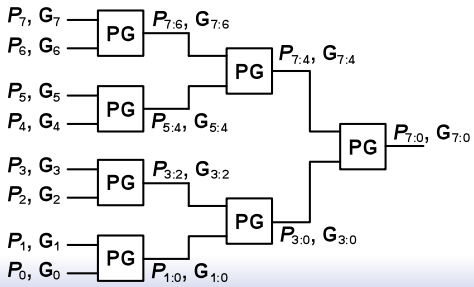


Look-Ahead Adders

Logarithmic (Tree) Adders – Basic Idea

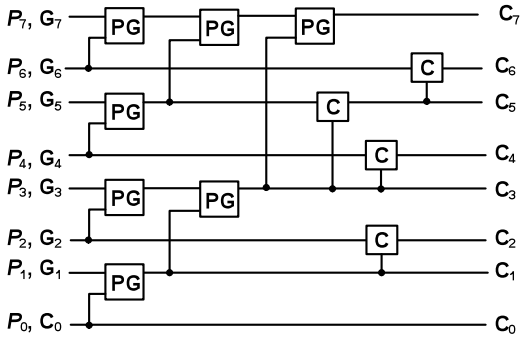
- "Look ahead" across groups of multiple bits to figure out the carry
  - Example with two bit groups:

    $P_{1:0} = P_1 \cdot P_0$, $G_{1:0} = G_1 + P_1 \cdot G_0$, → $C_{out1} = G_{1:0} + P_{1:0} \cdot C_{in}$
- Combine these groups in a tree structure:
  - Delay is now ~$\log_2(N)$
  - Instead of ~N



Rest of the Tree

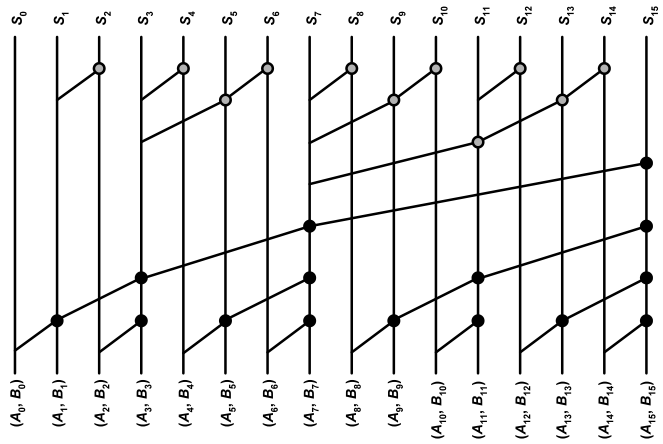
- Previous picture shows only half of the algorithm
  - Need to generate carries at individual bit positions too



Many Kinds of Tree Adders

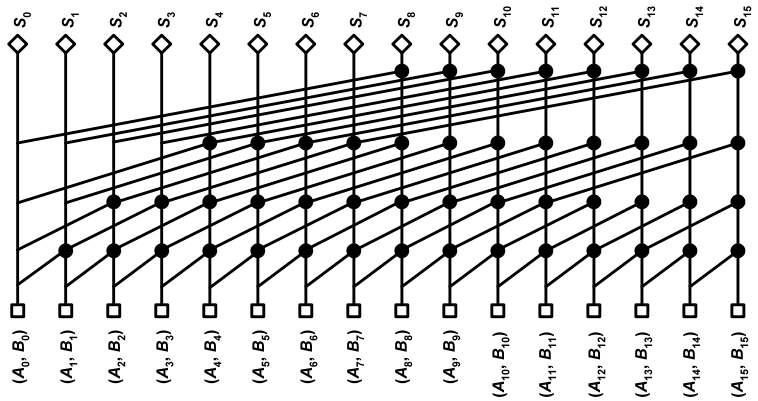
- Many ways to construct these tree (or "carry lookahead") adders
  - Many of these variations named after the people who first came up with them
- Most of these vary three basic parameters:
  - Radix: how many bits are combined in each PG gate
    - Previous example was radix 2; often go up to radix 4
  - Tree Depth: how many stages of logic you go through to get the final carry. Must be at least $\log_{Radix}(N)$
  - Fanout: Maximum logical branching in the tree

Tree Adders



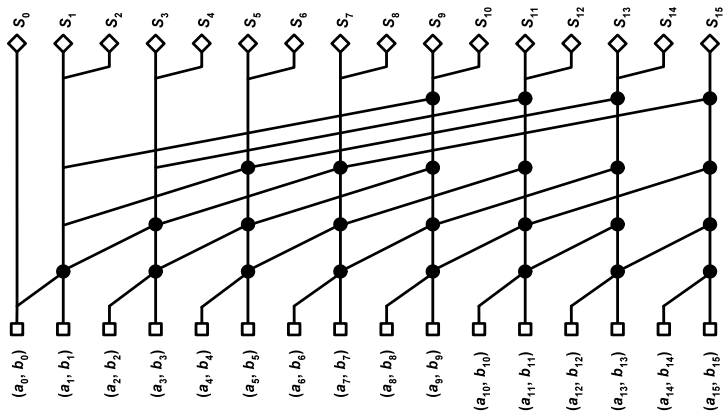
Brent-Kung Tree

Tree Adders



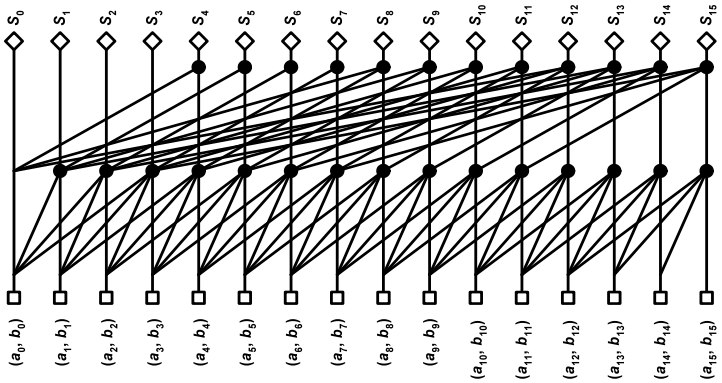
16-bit radix-2 Kogge-Stone tree

Tree Adders



16-bit radix-2 sparse tree with sparseness of 2

Tree Adders



16-bit radix-4 Kogge-Stone Tree

Multipliers

Binary Multiplication

	1 0 1 0 1 0	Multiplicand
×	1 0 1 1	Multiplier
	1 0 1 0 1 0	Partial products
	1 0 1 0 1 0	
	0 0 0 0 0 0	
+	1 0 1 0 1 0	
	1 1 1 0 0 1 1 1 0	Result

Binary Multiplication

$$Z = X \times Y = \sum_{k=0}^{M+N-1} Z_k 2^k$$

$$= \left(\sum_{i=0}^{M-1} X_i 2^i\right)\left(\sum_{j=0}^{N-1} Y_j 2^j\right)$$

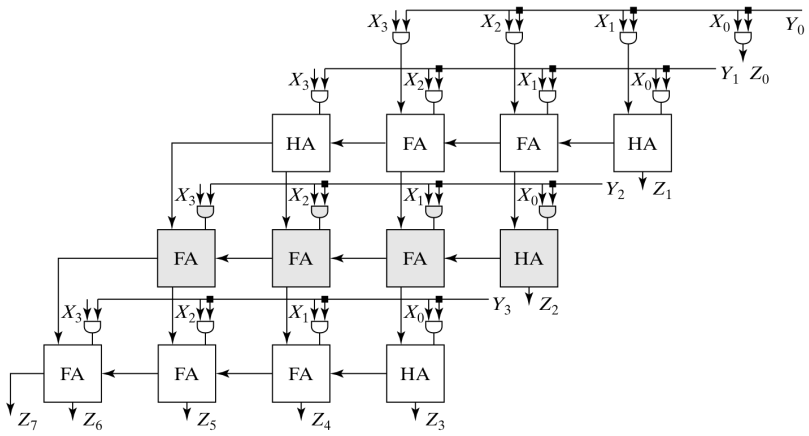
$$= \sum_{i=0}^{M-1}\left(\sum_{j=0}^{N-1} X_i Y_j 2^{i+j}\right)$$

with

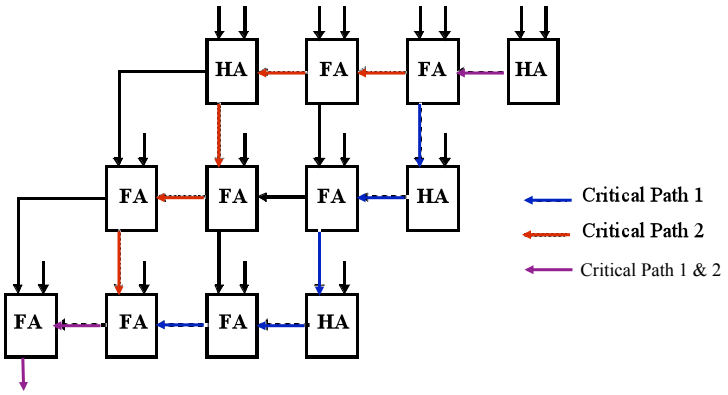
$$X = \sum_{i=0}^{M-1} X_i 2^i$$

$$Y = \sum_{j=0}^{N-1} Y_j 2^j$$

The Array Multiplier



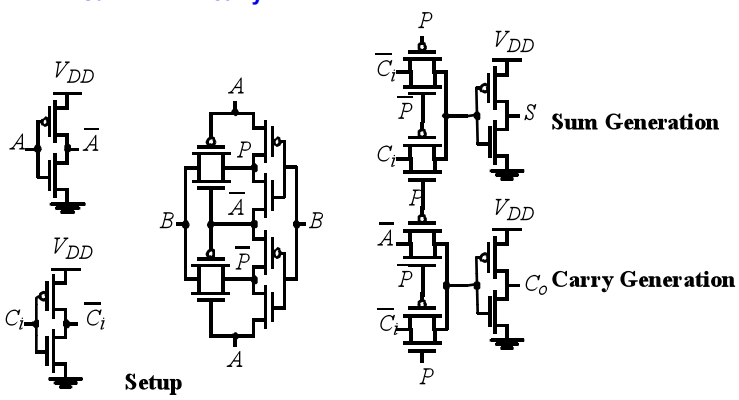
The M-by-N Array Multiplier: Critical Path



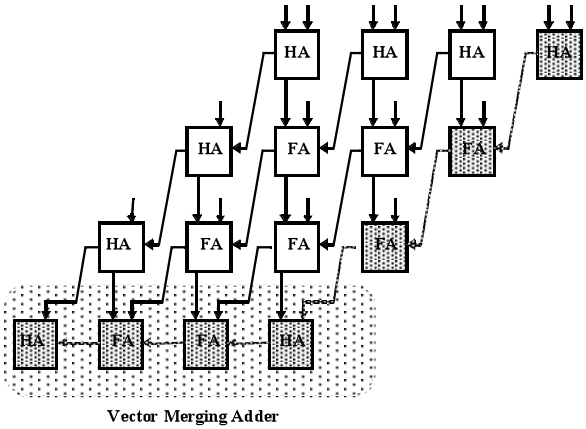
$$t_{mult} \approx \left[(M-1)+(N-2)\right]\cdot t_{carry} + (N-1)\cdot t_{sum} + t_{and}$$

Transmission-Gate Full Adder

Balanced t_{sum} and t_{carry}

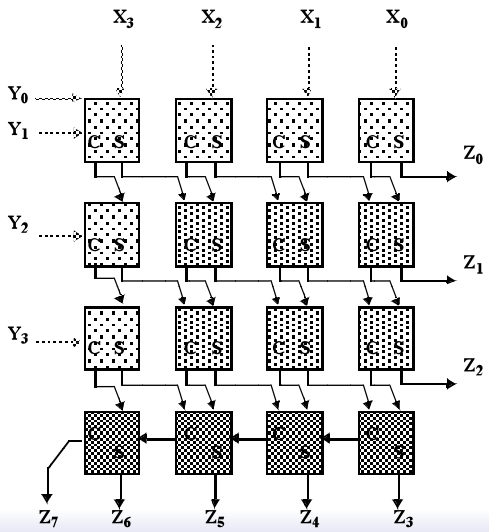


Carry-Save Multiplier



$$t_{mult} = t_{and} + (N-1)\cdot t_{carry} + t_{merge}$$

Multiplier Floorplan



HA Multiplier Cell

FA Multiplier Cell

Vector Merging Cell

X and Y signals are broadcasted through the complete array.
(.....)