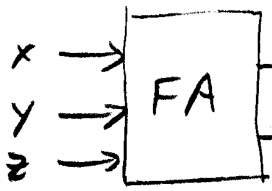
<< Sort with $\Theta(\lg N)$ procs in linear array >>
<< Firing squad – $\Theta(1)$-size state >>

Addition

Basic component: Full adder – combinational

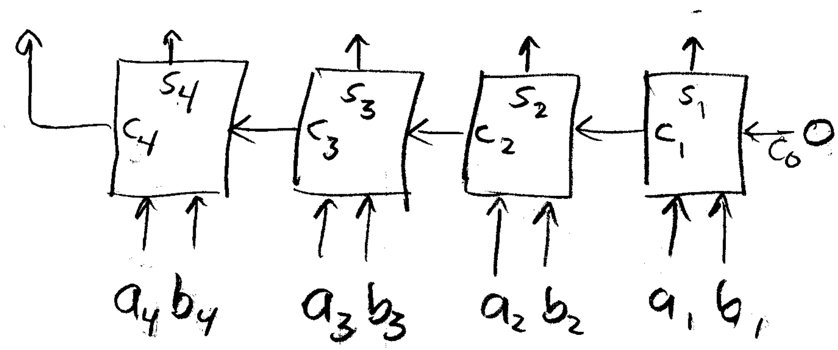


$\rightarrow c = MSB(x+y+z) = \text{majority}$

$\rightarrow s = LSB(x+y+z) = \text{parity}$

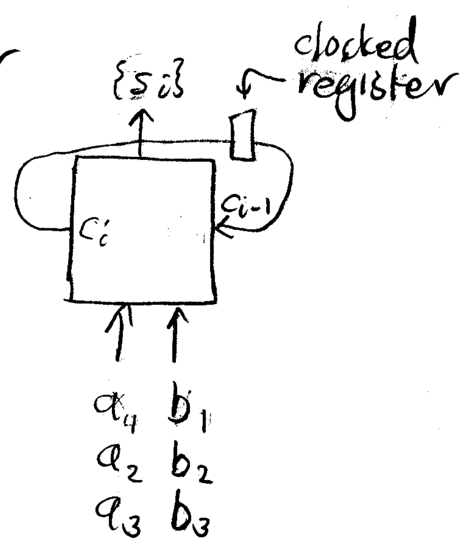
Problem:
Add 2 N-bit numbers

Ripple-carry adder



N-bit #'s $\Rightarrow$ $\Theta(N)$ time, $\Theta(N)$ HW, combinational

Serial adder



$\Theta(N)$ time, $\Theta(1)$ hardware, sequential (clocked)

Fast addition

Idea: carries are the hard part.
Know carries ⇒ compute sum in $\Theta(1)$ time
How? Array of full adders. «Show on ripple-carry adder»

	1	0	1	0	1	0	1	
	1	1	1	0	0	1	1	
1	1	0	0	1	0	0	0	
	g	p	g	k	p	p	g	(k)

Classify stages:

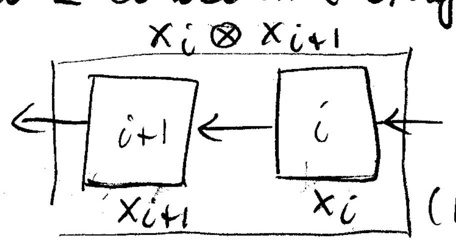
<u>kill</u>: $\substack{0\\0}$ ⇒ carry-out = 0

<u>propagate</u>: $\substack{0\\1}$ or $\substack{1\\0}$ ⇒ carry-out = carry-in

<u>generate</u>: $\substack{1\\1}$ ⇒ carry-out = 1

Carry into stage $= \begin{cases} 1 & \text{if most recent non-p is k} \\ 0 & \text{otherwise} \end{cases}$

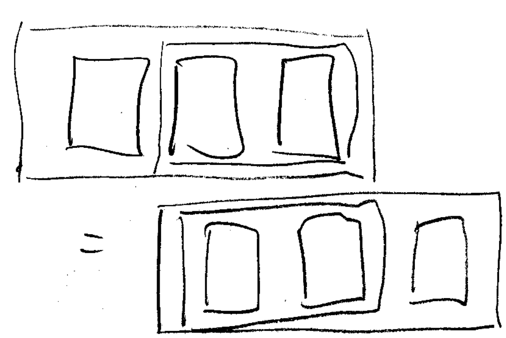
When do 2 consecutive stages kill, prop, gen?



(k, p, g)

$\otimes$		x_{i+1}: k	p	g
x_i	k	k	k	g
	p	k	p	g
	g	k	g	g

Associative!



<u>Theorem.</u> Let x_i = carry status of stage i, where $x_0 = k$. Define $y_i = x_0 \otimes x_1 \otimes \cdots \otimes x_i$.

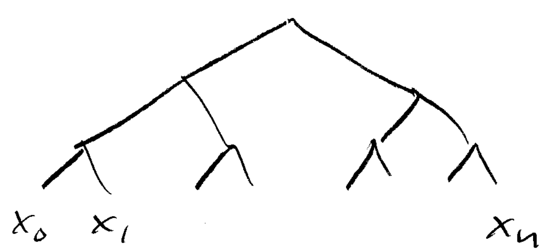
Then $y_i = k \Rightarrow c_i = 0$
$y_i = g \Rightarrow c_i = 1$
$y_i = p$ does not occur.

Proof. Induction on i. $\boxtimes$

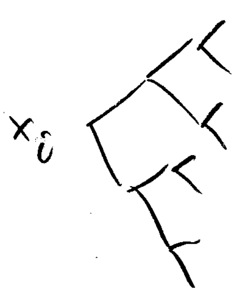
Log-time circuit:

$y_0 = x_0$
$y_1 = x_0 \otimes x_1$
$y_2 = x_0 \otimes x_1 \otimes x_2$
$y_N = x_0 \otimes x_1 \otimes \cdots \otimes x_N$

Use tree for each calculation:



Use tree to broadcast inputs (bounded-degree network):



Time = $\Theta(\lg N)$, HW = $\Theta(N^2)$.

Carry-lookahead addition

$\Theta(\lg N)$ time, $\Theta(N)$ HW.

"Parallel prefix"

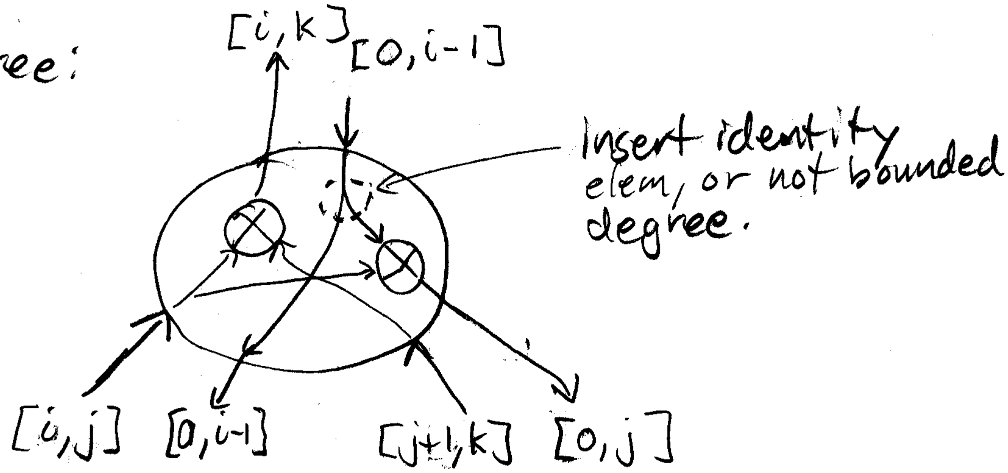
Let $[i, j]$ denote $x_i \otimes x_{i+1} \otimes \cdots \otimes x_j$

Lemma. $[i, j] \otimes [j+1, k] = [i, k]$ $\boxtimes$

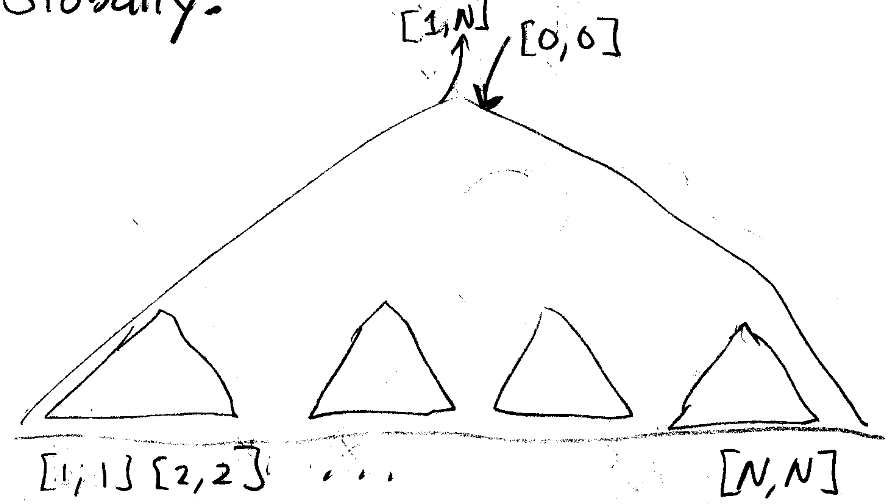
$x_i = [i, i]$

$y_i = [0, i]$

Build tree:

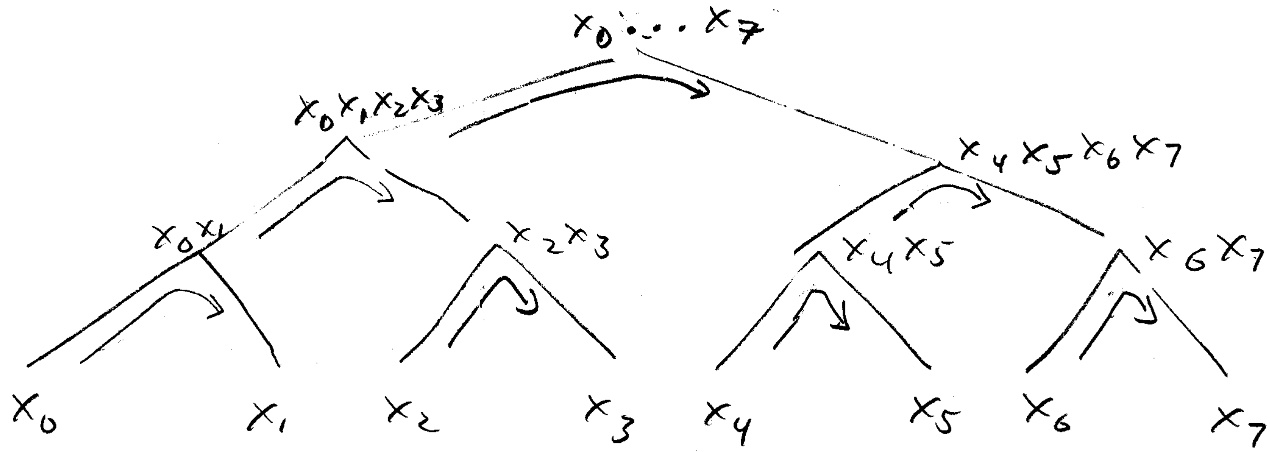


Globally:

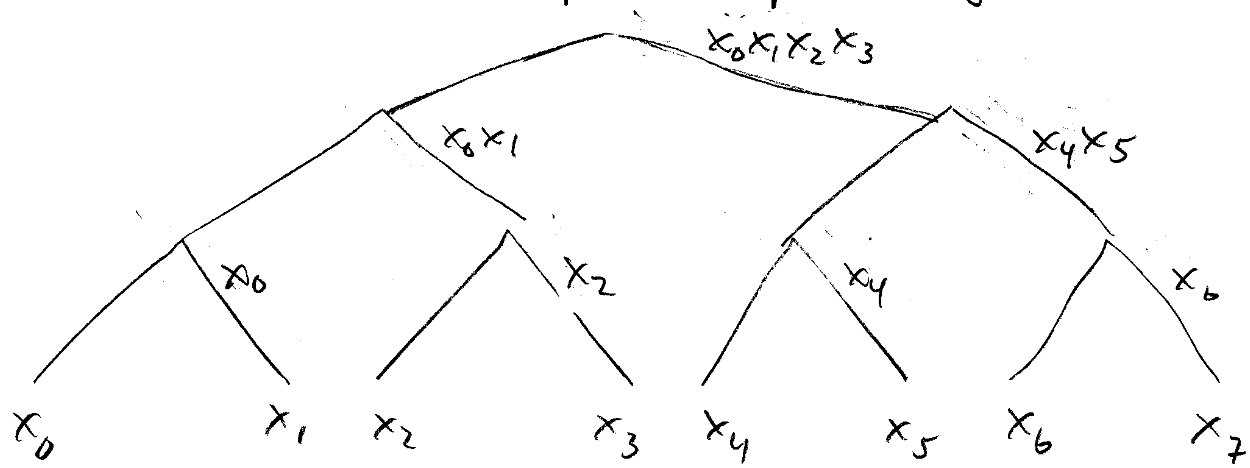


Left child values are passed up.

Similar method:



Left child values are passed up and right



Postscript Kill, propagate, generate first used in Harvard relay calculator circa mid-1940's.

$O(1)$-time addition (in their model).