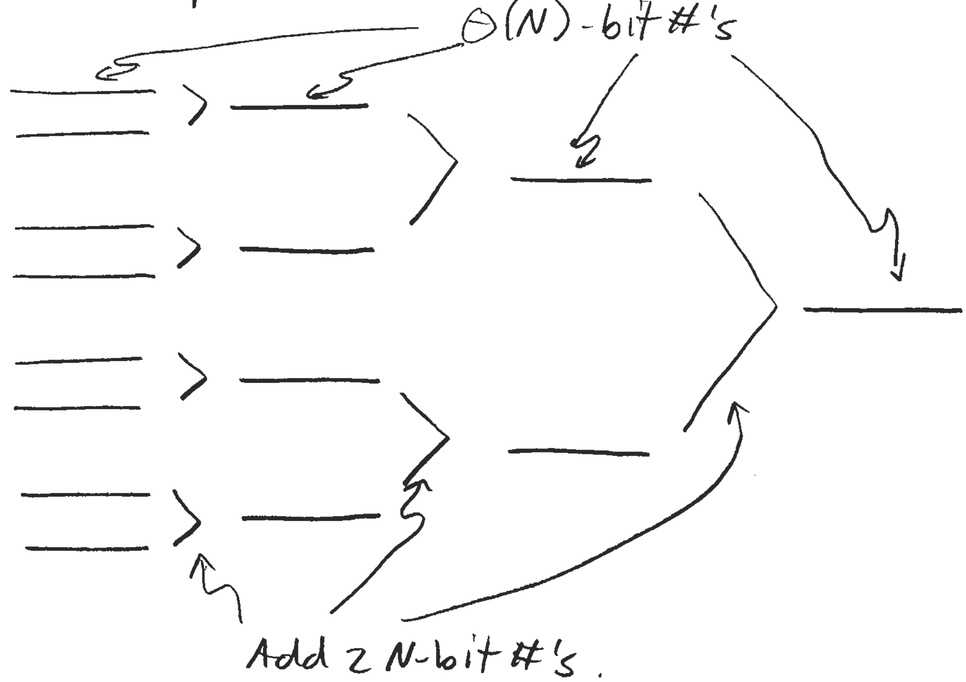
Adding N N-bit numbers

Add 2 N-bit #'s in $\Theta(\lg N)$ steps, $\Theta(N)$ HW
Add N 1-bit #'s in $\Theta(\lg N)$ steps, $\Theta(N)$ HW.

N N-bit #'s.

Break into pairs:



$\Theta(N^2)$ HW
$\Theta(\lg^2 N)$ steps
$\uparrow (\lg N)^2$

Can do better!

Carry-save addition

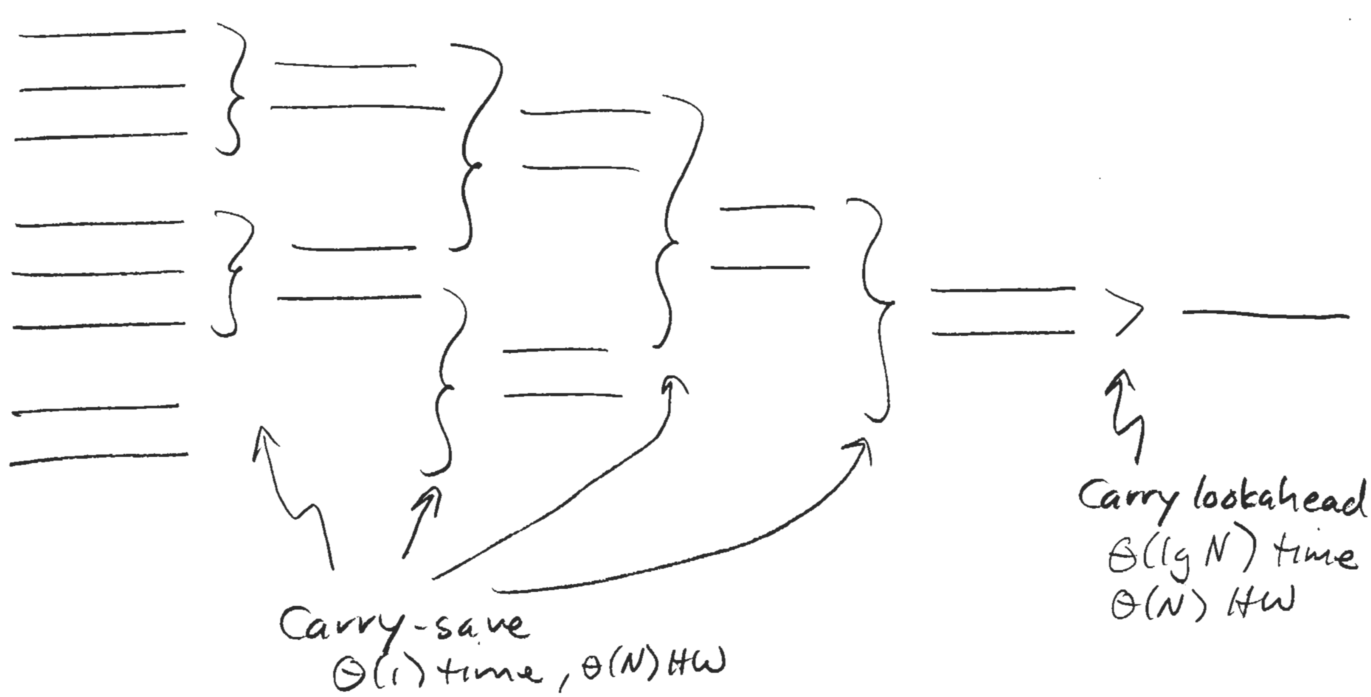
3 N-bit #'s $\rightarrow$ 2 #'s in 1 step!

Ex.

	1	0	1	1	0	1	0	
	1	0	1	0	1	1	1	
	0	1	1	1	0	1	1	
	0	1	1	0	1	1	0	← parity
1	0	1	1	0	1	1		← majority.

Array of full adders!

Wallace tree



Time $T(N) = T(\lceil 2N/3 \rceil) + \Theta(1)$ $(T(2) = 0)$

Master theorem: $T(N) = \Theta(\lg N)$

$T(N) \approx \log_{3/2} N$

HW $H^{\#}(N) = H^{\#}(\lceil 2N/3 \rceil) + \Theta(N)$ // Number of Hardware Components

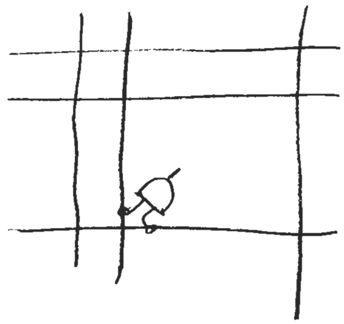
$= \Theta(N)$

$\Rightarrow H(N) = \Theta(N) \cdot H^{\#}(N) = \Theta(N^2)$

Integer multiplication

$$\begin{array}{rrrrr} & & 0 & 1 & 1 \\ & & 1 & 0 & 1 \\ \hline & & 0 & 1 & 1 \\ & 0 & 0 & 0 & \\ 0 & 1 & 1 & & \\ \hline \end{array}$$

- N #'s with $\le 2N$ bits each
- Form partial products with matrix of AND gates:



Avoid broadcast with tree.

- $\Theta(N^2)$ HW to form partial products in $\Theta(\lg N)$ time.

Wallace-tree add: $\Theta(\lg N)$ time, $\Theta(N^2)$ HW.

Convolution

$a = (a_1, a_2, \ldots, a_N)$
$b = (b_1, b_2, \ldots, b_N)$

Compute $c = (c_1, c_2, \ldots, c_{2N-1})$, where

$$c_1 = a_1 b_1$$
$$c_2 = a_1 b_2 + a_2 b_1$$
$$\vdots$$
$$c_k = a_1 b_k + \cdots + a_k b_1$$
$$\vdots$$
$$c_{2N-1} = a_N b_N$$

Polynomial multiplication

$$(a_1 + a_2 x + a_3 x^2 + \cdots + a_N x^{N-1})(b_1 + b_2 x + \cdots + b_N x^{N-1})$$
$$= c_1 + c_2 x + \cdots + c_{2N-1} x^{2N-2}$$

6.896
2/11/04
L3.4

Linear array:

$_3 \cdot a_2 \cdot a_1$ $\boxed{c_5}$ $\boxed{c_4}$ $\boxed{c_3}$ $\boxed{c_2}$ $\boxed{c_1}$ $b_3 \cdot b_2 \cdot b_1$

c_5	c_4	c_3	c_2	c_1
a_1	·	·	·	b_3
·	a_1	·	b_3	·
a_2	·	$a_1 b_3$	·	b_2
·	$a_2 b_3$	·	$a_1 b_2$	·
$a_3 \cdot b_3$	·	$a_2 b_2$	·	$a_1 b_1$
·	$a_3 b_2$	·	$a_2 b_1$	·
b_2	·	$a_3 b_1$	·	a_2

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

50% utilization
1. Solve 2 problems
2. Coalesce
3. Interlace
4. Adjust timing
5. Make multiplier slower

<u>Integer mult</u>

$$
\begin{array}{rrrrr}
 & & a_3 & a_2 & a_1 \\
 & & b_3 & b_2 & b_1 \\
\hline
 & & a_3b_1 & a_2b_1 & a_1b_1 \\
 & a_3b_2 & a_2b_2 & a_1b_2 & \\
a_3b_3 & a_2b_3 & a_1b_3 & & \\
\hline
\end{array}
$$

Idea:
- Similar to convolution
- Send carry to left.

Coarsening

<u>Def.</u> Sup. alg runs in T time on P procs.
The <u>work</u> is $P \cdot T$.

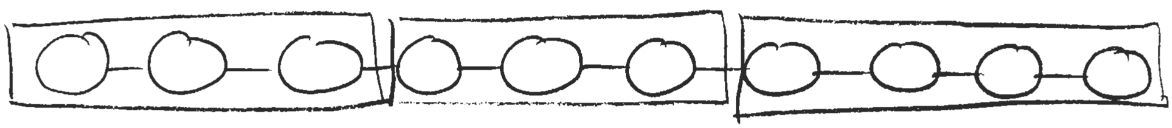
$$\underline{\text{Efficiency}} = \frac{\text{Work} \overset{\text{Time}}{=} \text{of (best) serial alg.}}{\text{Work of parallel alg.}}$$

<u>Theorem</u> A time-T, P-proc. alg can be simulated on an m-proc machine, where $m < P$, in $T \cdot \lceil \frac{P}{m} \rceil$ steps (assuming free multiplexing).

Proof. Simulate 1 P-proc step with $\lceil P/m \rceil$ m-proc steps. ⊠

Note: For fixed-connection networks, must also embed "guest" network in "host" network.

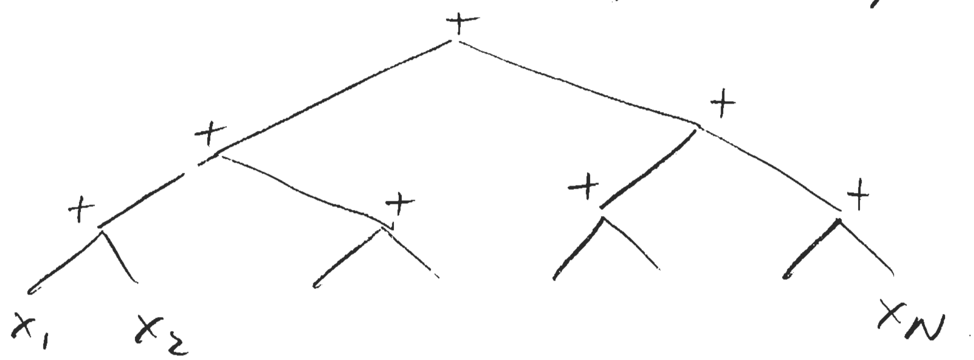
Ex. $P = 10$, $m = 3$



$$\begin{aligned} \text{Work of } m\text{-proc alg.} &= m \cdot T \lceil P/m \rceil \\ &\le m \cdot T\left(\frac{P}{m} + 1\right) \\ &= T \cdot (P + m) \\ &= \Theta(PT), \text{ since } m < P. \end{aligned}$$

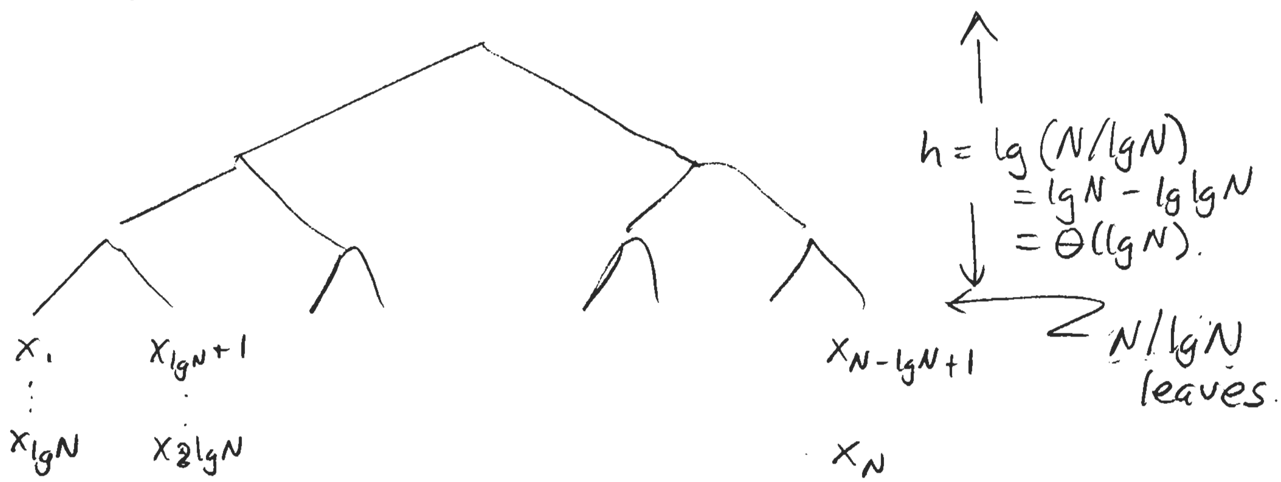
Thus, no asymptotic loss in efficiency.
Motivates study of fine-grained algs, since can always slow down for coarse grained.

Ex. Sum N numbers on complete binary tree.



Time $= \Theta(\lg N)$.
HW $= \Theta(N)$
Work $= \Theta(N \lg N)$.

More efficient:



Time $= \Theta(\lg N)$
HW $= \Theta(N/\lg N)$
Work $= \Theta(N)$.