

6.896  
2/11/04  
L3.1

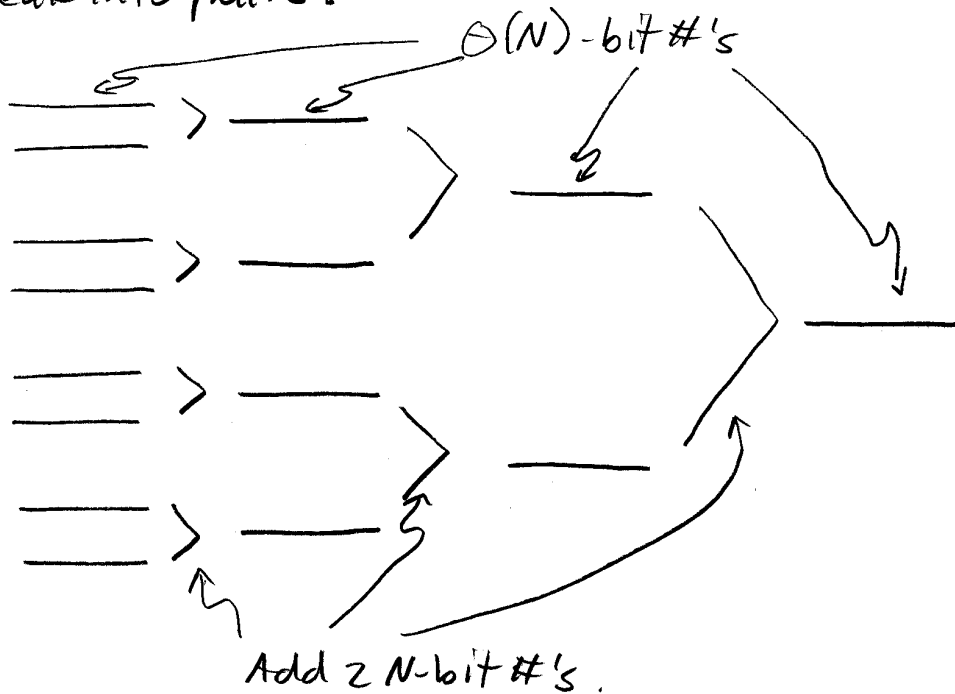
### Adding $N$ $N$ -bit numbers

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!

6.896  
2/11/04  
L3.2

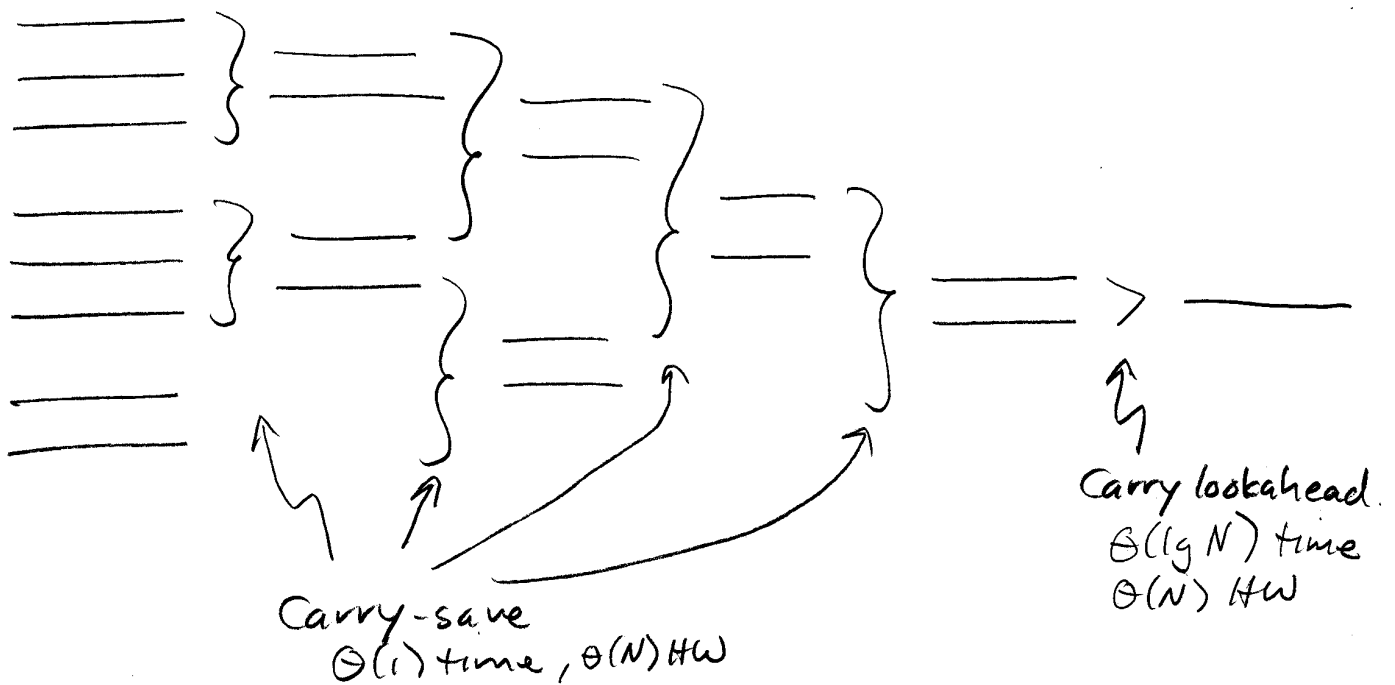
## Carry-save addition

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

Ex. 
$$\begin{array}{r} 1011010 \\ 1010111 \\ 0111011 \\ \hline 0/1/1/0/1/1/0 \leftarrow \text{parity} \\ 1/0/1/1/0/1/1 \leftarrow \text{majority} \end{array}$$

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  
 $= \Theta(N)$

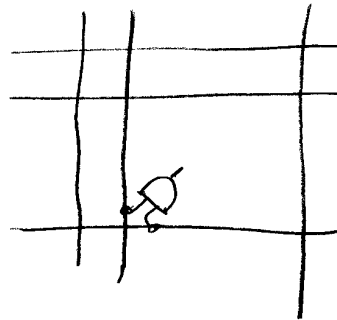
$\Rightarrow H(N) = \Theta(N) \cdot H^{\#}(N) = \Theta(N^2)$

6.896  
2/11/04  
L3.3

## Integer multiplication

$$\begin{array}{r} 011 \\ 101 \\ \hline 011 \\ 000 \\ 011 \\ \hline \end{array}$$

- $N$  #'s with  $\leq 2N$  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

$$a = (a_1, a_2, \dots, a_N)$$

$$b = (b_1, b_2, \dots, b_N)$$

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

$$c_1 = a_1 b_1$$

$$c_2 = a_1 b_2 + a_2 b_1$$

$\vdots$

$$c_k = a_1 b_k + \dots + a_k b_1$$

$\vdots$

$$c_{2N-1} = a_N b_N$$

## Polynomial multiplication

$$(a_1 + a_2 x + a_3 x^2 + \dots + a_N x^{N-1})(b_1 + b_2 x + \dots + b_N x^{N-1})$$

$$= c_1 + c_2 x + \dots + c_{2N-1} x^{2N-2}$$

6,896  
2/11/04  
L3.4

Linear array:

$a_3 \cdot a_2 \cdot a_1$	$C_5$	$C_4$	$C_3$	$C_2$	$C_1$	$b_3 \cdot b_2 \cdot b_1$
	$a_1$	$\cdot$	$\cdot$	$\cdot$	$b_3$	
	$\cdot$	$a_1$	$\cdot$	$b_3$	$\cdot$	
	$a_2$	$\cdot$	$a_1 b_3$	$\cdot$	$b_2$	
	$\cdot$	$a_2 b_3$	$\cdot$	$a_1 b_2$	$\cdot$	
	$a_3 \cdot b_3$	$\cdot$	$a_2 b_2$	$\cdot$	$a_1 b_1$	
	$\cdot$	$a_3 b_2$	$\cdot$	$a_2 b_1$	$\cdot$	
	$b_2$	$\cdot$	$a_3 b_1$	$\cdot$	$a_2$	

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

50% utilization

1. Solve  $Z$  problems
2. Coalesce
3. Interlace
4. Adjust timing

5. Make multiplier slower

Integer mult.

$$\begin{array}{r}
 a_3 \ a_2 \ a_1 \\
 b_3 \ b_2 \ b_1 \\
 \hline
 a_3 b_1 \ a_2 b_1 \ a_1 b_1 \\
 a_3 b_2 \ a_2 b_2 \ a_1 b_2 \\
 a_3 b_3 \ a_2 b_3 \ a_1 b_3 \\
 \hline
 \end{array}$$

Idea:

- Similar to convolution
- Send carry to left.

6.896  
2/11/04  
L3.5

## Coarsening

Def. Sup. alg runs in  $T$  time on  $P$  procs.

The work is  $P \cdot T$ .

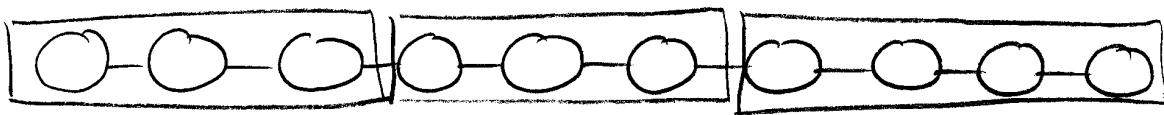
$$\text{Efficiency} = \frac{\text{Work}^{\text{Time}} \text{ of (best) serial alg.}}{\text{Work of parallel alg.}}$$

Theorem 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.  $\square$

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

Ex.  $P=10, m=3$



$$\text{Work of } m\text{-proc alg.} = m \cdot T \lceil P/m \rceil$$

$$\leq m \cdot T \left( \frac{P}{m} + 1 \right)$$

$$= T \cdot (P + m)$$

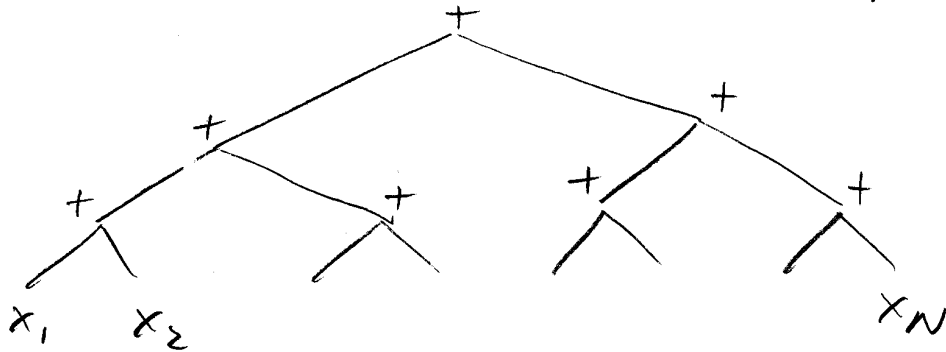
$$= \Theta(PT), \text{ since } m < P.$$

Thus, no asymptotic loss in efficiency.

Motivates study of fine-grained algs, since can always slow down for coarse grained.

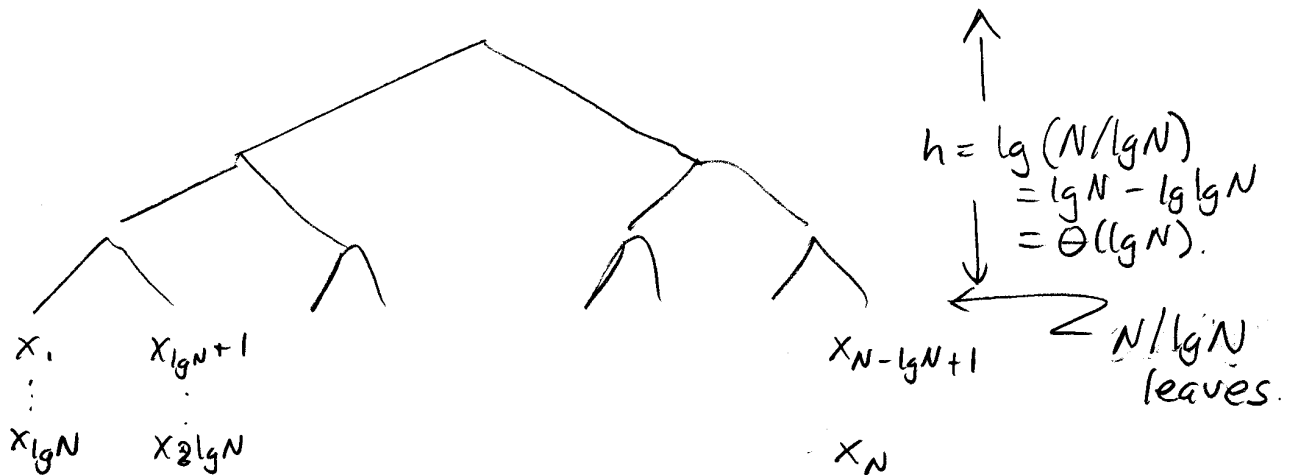
6.896  
2/11/04  
L3.6

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 =  $\Theta(N)$ .