CS273 Foundations of Parallel and Network Computing **Spring 2003**

Lecture 14: March 12, 2003

Lecturer: Satish Rao *Scribe: Kaushik Datta*

Note: These notes borrow heavily from the previous year's notes on VLSI chip design in addition to the lecture.

14.1 Overview

This lecture continues our discussion of VLSI chip design. We will cover the basics and lower bounds of the circuit model, area lower bounds and AT^2 bounds of various algorithms, and the Partition Lemma.

14.2 Introduction- The OR operation

If we wanted to take the OR of several bits, there are several ways to do it:

1. A simple way to do so would be to create a binary tree with the input bits at each leaf. The width of the tree is $N/2$, while the height is $\log N$, resulting in an area of $O(N \log N)$. In addition, the time $T = O(\log N)$. If we consider AT to be the total work, then $AT = O(N \log^2 N)$ in this case.
2. A better layout can be produced by recursing using the divide-and-conquer strategy. This results in the recurrence relation $S(N) = 2S(N/4) + O(1)$, which can be solved via Master's Theorem to be $O(\sqrt{N})$. Therefore area is $O(N)$. In terms of our resources, area $A = O(N)$ while time $T = O(\log N)$, so our total work is $AT = O(N \log N)$.
3. Another way to take the OR of many bits is to start at the root of the binary tree and go down to level $\log \log N$. Input $\log N$ bits into one gate, and proceed from there. In this case, $A = O(N/\log N)$ and $T = O(\log N)$, resulting in the total work of $O(N)$.

14.3 The Circuit Model

14.3.1 Basics

- Each processor can store only one word (ω bits) at a time.
- Each processor can read/write one word per time step.
- Each processor can send one word on each link per time step.
- Each input word is read only once.
- The I/O is where/when oblivious.

Definition 14.1 *Suppose we have:*

INPUT: $x_1, x_2, \ldots, x_m$

OUTPUT: $y_1, y_2, \ldots, y_n$.

Where oblivious *means that the processor that reads x_i (or generates y_j) is fixed before execution begins.* When oblivious *means that the time at which x_i is read (or y_i is generated) is fixed before execution begins.*

14.3.2 Lower Bounds

- **Memory:** $A \geq f$

  If f words must be remembered at some point in the computation, $A \geq f$ in order to store all the words.

- **I/O:** $AT \geq f$

  If the I/O consists of f words, $AT \geq f$ since a chip of area A can read/write at most A words per time step.

- **Bisection:** $AT^2 \geq f^2$

  Assume that a cut of length $\sqrt{A}$ has divided the chip in half, and f words needs to be communicated across the cut in T time steps. Since $\sqrt{A}$ words can be communicated per time step, then $f < T\sqrt{A}$, or $AT^2 > f^2$.

- **Chip Area:** $A \geq \frac{\log N}{\omega}$

  Definition 14.2 *A* chip state *corresponds to a particular set of values stored in the processors. A* chip behavior *corresponds to a particular set of output values.*

  A chip with P processors can have $2^{P\omega}$ chip states, since each processor can store one word (ω bits). Similarly, a chip that outputs n words can have $2^{n\omega}$ behaviors. Note that the chip behavior is a function of both the original chip state and the inputs.

  Suppose the chip goes through a number of executions $E_1, E_2, \ldots, E_N$, where at least t time steps are performed in each execution so that all the inputs are read in. Observe that if E_i and E_j produce different behaviors given the same input, their original chip states must have been different. Therefore, if the chip state at time t is different in all executions given the same inputs, then all the original chip states must have been different. Thus we can say that $2^{P\omega} \geq N$ or $P \geq \frac{\log N}{\omega}$. This implies that if a chip has area A, then $A \geq \frac{\log N}{\omega}$.

14.4 Area Lower Bounds

14.4.1 Cyclic Shift

INPUT 1: $x_{n-1}, x_{n-2}, \ldots, x_0$ where $\omega \geq \log n$.

INPUT 2: s (shift amount)

OUTPUT: $y_{n-1}, y_{n-2}, \ldots, y_0$ where $y_i = x_{(i+s) \ mod \ n}$.

Lemma 14.3 *Cyclic shift requires $A = \Omega(n)$.*

Claim 14.4 *All x_i's have to be read in before any y_j is generated.*

Proof: Suppose y_j is generated before x_i is read in for some execution. Then according to "when obliviousness", this must be true for all executions. Let us set $s = (i - j) \bmod n$. The chip must then output $y_j = x_{(j+s) \bmod n} = x_i$. Since x_i has not been read in yet, we can choose a value for x_i that will cause the chip to make an error. Thus, a contradiction occurs. ■

Cyclic shift is "when oblivious", since each input does not depend on the the others. Therefore we can choose that x_i is read in at an arbitrary time t_i. It is NOT legal to read in a specific x_i as a function of y_i. The number of behaviors after reading in all the x_i's is $f = 2^{n\omega}$, since there are 2^{ω} output values for each y_i. Then, according to the lower bound on chip area given above, $A = \Omega(\frac{\log 2^{n\omega}}{\omega}) = \Omega(n)$.

Another way to interpret this argument is to realize that there is a time instant t at which the chip must have all n data words in memory. Thus, n processors are necessary, and hence an area of n.

14.4.2 Integer Multiplication

INPUTS: $a_{n-1}, a_{n-2}, \ldots, a_0$ and $b_{n-1}, b_{n-2}, \ldots, b_0$

OUTPUT: $c_{2n-1}, \ldots, c_0$

where a_i, b_j, c_k are all bits.

Theorem 14.5 *$A = \Omega(n)$ in the bit model.*

Proof: Let:

$X' = b_{n-1}, \ldots, b_{n/2}$

$Y' = c_{3n/2-1}, \ldots, c_n$

Note that every bit in Y' is dependent on every bit in X' (with the bits of a acting as powers of 2). Therefore, all of X' must be read before any of Y' is outputted. Thus, in order to store X', the chip area must at least $n/2$. ■

14.4.3 Summary of Area Lower Bounds

Word model: $\omega = \theta(\log n)$

Cyclic shift	$\Omega(n)$
Sorting	$\Omega(n)$

Bit model:

OR	$\theta(1)$
Addition mod 2^n	$\theta(1)$
Addition mod $2^n - 1$	$\Omega(n)$
Integer Multiplication	$\Omega(n)$
Convolution	$\Omega(n)$
Sorting n integers each $(1 + \log n)$ bits wide	$\theta(n)$
Sorting n integers each $2 \log n$ bits wide	$\Omega(n \log n)$

Exercise: Find matching upper bounds.

14.5 AT^2 Bounds

Definition 14.6 *The* transcript *of C is a concatenation of all the words transmitted across C over time. In other words, the transcript keeps track of all the communication across cut C.*

The behavior of L can be considered to be a function of L's input values and C's transcript. We are given that all inputs in L are fixed. Therefore, L will have different behaviors only if the transcript varies (i.e. varying information crosses over C into L). If there are f different behaviors for a fixed set of inputs, then $f \leq 2^{\omega|C|T}$, where $|C|$ = number of wires crossing cut C and T = execution time.

Theorem 14.7 *The Transcript Theorem*

Suppose that $|C| \leq h+1 \leq \sqrt{A}+1$, where h is the height of the cut C. Then, $\frac{\log f}{\omega} \leq |C|T \leq T\sqrt{A}$, or $AT^2 \geq \frac{\log^2 f}{\omega^2}$.

Lemma 14.8 *The Partition Lemma*

Assume no processor outputs more than $n/3$ values. Then there exists a small cut C and constant α such that $X' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{n/6}}\}$ are input on one side and $Y' = \{y_{i_1-\alpha}, y_{i_2-\alpha}, \ldots, y_{i_{n/6}-\alpha}\}$ are output on the other side.

Claim 14.9 *There exists a small cut C s.t. at least $n/3$ outputs are generated on each side. In other words, $n/3 \leq$ number of $y_i's \leq 2n/3$ for both sides of the cut.*

Proof of Claim: Use the modified sliding argument, where the cut C is slid across the chip from left to right until the claim is met. Note that the cut can cover only part of a column. ■

Proof of Partition Lemma: Assume that the small cut C in the above claim exists. WLOG assume side L reads in at least $n/2$ inputs, so that:

Inputs on side L: $x_{i_1}, x_{i_2}, \ldots, x_{i_{n/2}}$

Outputs on side R: $y_{j_1}, y_{j_2}, \ldots, y_{j_{n/3}}$

Connect each input with each output and vice versa so that a complete bipartite graph is formed. There will be $(n/2)(n/3) = n^2/6$ edges in the graph. Now, assign colors to the edges according to the formula $(x_p, y_q) = (p-q) \bmod n$, where $0 \leq color < n$. There will be at least $n/6$ edges of each color present. In addition, $color(x_p, y_q)$ is equal to the shift count which will let y_q equal x_p. If we assign X' and Y' to be the endpoints of the edges with the most popular color, the conditions of the Partition Lemma will be met. ■

14.5.1 Cyclic Shift

INPUT 1: $x_{n-1}, x_{n-2}, \ldots, x_0$ where $\omega \geq \log n$.

INPUT 2: s (shift amount)

OUTPUT: $y_{n-1}, y_{n-2}, \ldots, y_0$ where $y_i = x_{(i+s) \bmod n}$.

Lemma 14.10 *Cyclic shift requires $AT^2 = \Omega(n^2)$.*

Proof: We will apply the Partition Lemma. WLOG let X' be the input for R and let Y' be the output for L. Now, consider the $2^{n\omega/6}$ executions in which:

- $s = \alpha$.
- All $x_i \notin X'$ are given value 0.
- X' has distinct values in each execution. Since X' has a total of $n\omega/6$ bits, this is possible.

This means that $y_{i_j - \alpha} = x_{i_j}$ in all executions. Furthermore, Y' must be distinct in all executions. Now, we know that $n\omega/6$ bits cross cut C, so we apply the Transcript Theorem to get that $AT^2 = \Omega(\frac{\log^2 2^{n\omega/6}}{\omega^2}) = \Omega(n^2)$.

■

Cyclic Shift Upper Bounds

$\sqrt{n} \times \sqrt{n}$ **Mesh:**

$A = O(n)$

$T = O(\sqrt{n})$

$AT^2 = n^2$ Optimal Network

Area cannot be reduced without increasing time and vice versa.

n **-node Hypercube:**

$A = O(n^2)$

$T = O(\log n)$

$AT^2 = O(n^2 \log^2 n)$ Suboptimal Network

Improvement possible in time or area without increasing the other.

n **-node Butterfly:**

$A = O(n^2/log^2 n)$

$T = O(\log n)$

$AT^2 = O(n^2)$ Optimal Network

Area cannot be reduced without increasing time and vice versa.

14.5.2 Boolean Matrix Multiplication

INPUTS: $n \times n$ matrices A, B.

OUTPUT: $n \times n$ matrix C where $C = AB$.

Theorem 14.11 $AT^2 = \Omega(n^4)$.

Proof: Assume no processor generates $\geq n^2/3$ outputs. There exists a cut π such that $\geq n^2/3$ bits of C are generated on both sides. Then, either:

- $n^2/6$ bits of A are generated on one side of π which can be forced to move across the cut by setting B to a shift matrix.
- $n^2/6$ bits of B are generated on one side of π which can be forced to move across the cut by setting A to a shift matrix.

Finally, use the Transcript Theorem. ■

Matrix Multiplication Upper Bounds

$n \times n$ **Mesh:**

$A = O(n^2)$

$T = O(n)$

$AT^2 = O(n^4)$ Optimal Network

n^3 **Processor Hypercube:**

$A = O(n^6)$

$T = O(\log n)$

$AT^2 = O(n^6 \log^2 n)$ Suboptimal

Using specialized networks can produce optimal AT^2 even for small T.