When we talk about algorithm design on networks: we worry about communication and computation. When we talk about algorithm design on PRAMs: we worry about computation. In this chapter let us worry only about communication. Thompson grid model is useful for this. The key question is: Suppose computation were (essentially) free, but not communication. How much would this limit speedup? We have done this to some extent already when we considered bisection width based bounds and even diameter based bounds. In this lecture we will make the bisection width based bounds more formal, and show that communication is a bottleneck even in problems you might not suspect. For this we will use Thompson's VLSI model, with wordsize w possibly larger than 1. This means each track intersection can hold a w bit processor. Each track can carry w wires.

Here is a flavour of possible results. We can show that no matter what network is used, $AT^2 = \Omega(n^2)$ for

1. Cyclic shift of n numbers.
2. Multipliplication of n digit numbers. Each digit is w bit wide.
3. List ranking on n element lists. $w = \theta(\log n)$
4. Sorting $w = 2\log n$ bit integers.

It is also possible to prove that multiplying $n \times n$ matrices has $AT^2 = \Omega(n^4)$. This has the following interesting implication: the time to multiply matrices using $n \times n$ array is $\Omega(n)$ because for $n \times n$ arrays $A = n^2$. This shows that for the $n \times n$ array, the simple algorithm is optimal and cannot be beaten even if you use Strassen's algorithm like (or any other) ideas. As you can see, this last conclusion has nothing to do with VLSI.

In this lecture we will prove the result on matrix multiplication. The exercises will touch upon the other results.

1 Terminology

In what follows we will use several colloquial words formally. Thus it will be useful to state clearly what we mean by them.

Execution: The sequence of events starting at power up and including reading of inputs, computation and generation of outputs. We note that if a chip has m inputs (words), then it can only have 2^{mw} distinct executions.

State: The values stored in the processors. For simplicity if we assume that each processor can store at most 1 word, a P processor chip can have at most 2^{Pw} distinct states.

Behaviour: Values generated as outputs. A chip with n outputs (words) can have at most 2^{nw} behaviours. By "behaviour after time t" we will mean the values generated after time t.

Definition 1 *The communication transcript into a part P of a chip is the set of words received by it from the rest of the chip over the entire execution.*

Note that if the part has m wires coming in, and T is the exeuction time, then there can be at most mwT bits communicated, and hence at most 2^{mwT} different transcripts possible.

2 Transcript theorem

Definition 2 *Let X, Y be the inputs and outputs of certain computational problem. Let $X' = (x_0, \ldots, x_{n-1}) \subseteq X, Y' = (y_0, \ldots, y_{n-1}) \subseteq Y$. Then we will say that X' flows to Y' under control c if there exist 2^{nw} executions in which*

1. *The inputs $X - X'$ take the value c in each execution.*
2. *The inputs X' take different values in each execution.*
3. *The outputs Y' take on exactly the same values as X'.*

We will say X' flows to Y' if there exists c such that X' flows to Y' under control c.

Example: Consider $n \times n$ matrix multiplication $R = PQ$. By setting P to be the identity matrix (control), we have $Q = X'$ flowing to $R = Y'$. Likewise we have a flow from P to R.

Theorem 1 (Transcript theorem) *Suppose a cut of size $O(\sqrt{A})$ divides a chip into parts L, R. Suppose L reads inputs X' and R generates outputs Y', where X' and Y' both contains n words. Suppose X' flows to Y'. Then $AT^2 = \Omega(n^2)$.*

Proof: Let c denote the control for which X' flows to Y'. Consider executions in which all inputs besides X' are set to c, while distinct values are given to X'. Clearly there are 2^{nw} such executions.

In all these exeuctions, the values read by R from the external world are the same (held at the control c). Thus for R to have different executions, it must receive different transcipts.

But the cut has size $O(\sqrt{A})$. Thus the number of different transcripts is $2^{O(\sqrt{A}wT)}$.

Thus $2^{O(\sqrt{A}wT)} \geq 2^{nw}$. ■

3 Application to matrix multiplication

All that remains is to now show a large flow in matrix multiplication. Further, the flow must cross a small cut. For this we explore the different kinds of flow inherent in matrix multiplication.

3.1 Shift matrix

Let S_i = Matrix obtained by rotating all rows of the identity matrix by i circularly.
Example: $n = 5$, $i = 1$

$$S_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$R = S_1 Q$ = circular rotation of rows of Q one step upwards. q_{ij} flows to $r_{i-1 \bmod n,j}$
$R = PS_1$ = circular rotation of columns of P one step right. p_{ij} flows to $r_{i,j+1 \bmod n}$

But we can have shifts of any magnitude from 0 to $n-1$. Thus by chosing P to be a shift matrix we can make a row of Q to flow to any row of R; likewise by choosing Q to be a shift matrix we can make any column of P to flow to any column of R.

3.2 Proof Outline:

You give me a chip. I will examine it and I will tell you one of the following after examining it.

- Your chip contains some processor that generates at least $n^2/3$ elements of R. This will trivially prove $AT^2 = \Omega(n^4)$.
- Your chip contains a small cut on one side of which $n^2/18$ elements P' of P are read and on the other side of which $n^2/18$ elements R' of R are generated, such that P' flows to R'. I will also tell you under what control P' flows to R': this will happen by setting Q to a shift matrix.
- Your chip contains a small cut on one side of which $n^2/18$ elements Q' of Q are read and on the other side of which $n^2/18$ elements R' of R are generated, such that Q' flows to R'. I will also tell you under what control Q' flows to R': this will happen by setting R to a shift matrix.

4 Proof

If any processor generates $n^2/3$ outputs, then we are done. So assume there is no such processor.

4.1 Output partitioning

We show how to find the small cut. Assume the chip has height h length l, $h \leq l$. Number intersections in column major order. Let $N(i)$ denote the number of elements of R generated by processors in intersections $1 \ldots i$. We have $N(0) = 0$, $N(A) = n^2$. We also know that $N(i)$ increases at most by $n^2/3$. Thus there is some j such that $n^2/3 \leq N(j) \leq 2n^2/3$. This is our required cut π. Any such cut has size at most $h + 1 = O(\sqrt{A})$.

4.2 Flow graph

Now we need to find the control under which we have a large flow. We will pose this as a graph partitioning problem.

Consider a graph in which vertices are elements of P, Q, R. The edges indicate possible flow, with a colour indicating what shift matrix causes the flow as follows.

If we choose P to be a shift matrix that downshifts by s, we will have a flow from q_{ij} to $r_{i,j+s \bmod n}$. So for $s = 1, \ldots, n$ we will have an edge $(q_{ij}, r_{i,j+s \bmod n})$ for all i, j. This edge will have colour s. Likewise if we choose Q to be a shift matrix that rightshifts by s, we will have a flow from p_{ij} to $r_{i+s \bmod n,j}$. So for $s = 1, \ldots, n$ we will have an edge $(p_{ij}, r_{i+s \bmod n,j})$ for all i, j. This edge will have colour s'.

The graph will have complete bipartite graphs between the jth column of Q to the jth column of R, and complete bipartite graph between the ith row of P to the ith row of R.

Imagine the graph placed on our chip. We know that at least $n^2/3$ vertices lie on either side of the cut π of Section 4.1. If we show that $n^2/18$ edges of a single colour $s(s')$ cross π, we are done: We simply set $P(Q)$ to be a shift matrix of magnitude $s(s')$, and we have flow of size $n^2/18$ from elements of $Q(P)$ to R.

For this we will show that at least $n^3/9$ edges cross π. From this the result follows since there are only $2n$ colours.

For this we embed the complete directed graph on n^2 nodes into G with vertices placed on the elements of C. We know that both sides have at least $n^2/3$ vertices. So at least $n^4/9$ edges of the complete graph must cross the cut. If they are embedded in E edges with congestion C, then we have $EC \geq n^4/9$. We will show that the congestion is n, proving that $E \geq n^3/9$.

The embedding is as follows. The path from r_{ij} to r_{kl} is embedded through p_{ik}, r_{il}, q_{jl}.

The directed congestion on edges (r_{ij}, p_{ik}) is contributed by paths from r_{ik} to r_{kt} for any t. Thus the directed congestion is n.

The directed congestion on edges (p_{ik}, r_{il}) is contributed by paths from r_{it} to r_{kl} for any t. Thus the directed congestion is n.

The directed congestion on edges (r_{il}, q_{jl}) is contributed by paths from r_{ij} to r_{tl} for any t. Thus the directed congestion is n.

The directed congestion on edges (q_{jl}, r_{kl}) is contributed by paths from r_{tj} to r_{kl} for any t. Thus the directed congestion is n.