# Las Vegas is better than Determinism in VLSI and Distributed Computing (Extended Abstract)

Kurt Mehlhorn
Fachbereich 1o
Universität des Saarlandes
66 Saarbrücken, West-Germany

Erik M. Schmidt
Dep. of Computer Science
Aarhus University
Aarhus, Denmark

## I. Introduction

In this paper we describe a new method for proving lower bounds on the complexity of VLSI - computations and more generally distributed computations. Lipton and Sedgewick observed that the crossing sequence arguments used to prove lower bounds in VLSI (or TM or distributed computing) apply to (accepting) nondeterministic computations as well as to deterministic computations. Hence whenever a boolean function f is such that f and $\bar{f}$ (the complement of f, $\bar{f} = 1 - f$) have efficient nondeterministic chips then the known techniques are of no help for proving lower bounds on the complexity of deterministic chips.

In this paper we describe a lower bound technique (Thm 1) which only applies to <u>deterministic</u> computations. More specifically, we will show that in order to compute $f(x,y)$, $x \in X$, $y \in Y$ deterministically using two computing agents, one of which knows x and one of which knows y, $\log \operatorname{rank}_k f$ bits have to be exchanged between the two computing agents. Here $\operatorname{rank}_k f$ denotes the rank of 0-1 matrix $(f(x,y))_{x\in X, y \in Y}$ over the field k; k is arbitrary.

In the application to VLSI, we cut the chip in half with respect to the inputs, and let the two halves of the chip be the two computing agents.

The lower bound technique is strong enough to distinguish the complexity of nondeterministic and deterministic computations. Even more is true. It is strong enough to distinguish Las Vegas and deterministic computations, i.e. we will exhibit a specific function f such that

$$AT^2_{ndet}(f),\ AT^2_{ndet}(\bar{f}),\ AT^2_{Las\ Vegas}(f) \ll AT^2_{det}(f).$$

Here, $AT^2$ denotes the complexity measure area x time$^2$ of where-oblivious chips.

A Las Vegas chip uses internal randomization. However, the output does not depend on the chance events in the algorithm, i.e. Las Vegas algorithms have zero probability of error. The running time of a Las Vegas algorithm is the expected running time averaged over all possible outcomes of the chance events.

Similarly, we exhibit a language L such that

$$S\cdot T_{ndet}(L),\ S\cdot T_{ndet}(\bar{L}),\ S\cdot T_{Las\ Vegas}(L) \ll S\cdot T_{det}(L)$$

Here, S·T denotes the complexity measure space x time of multi-tape Turing machines.

## II. A Lower Bound on Deterministic Two-Way Communication Complexity

Following Yao [Yao 1 and Yao 2] we make the following definitions. Let X and Y be sets and let $f : X \times Y \to \{0,1\}$ be a 0-1 valued function. We are interested in the following problem. Let $x \in X$ and $y \in Y$ be known to persons L and R respectively. For L and R to determine cooperatively the value $f(x,y)$, they send information to each other alternately, one bit at a time, according to some algorithm. The quantity of interest, which measures the information exchange necessary for computing f, is the minimum number of bits exchanged in any algorithm.

A deterministic algorithm is given by two response functions $h_L : X \times B^* \to B$ and $h_R : Y \times B^* \to B$ and the partial output function $a : B^* \to B$, where $B = \{0,1\}$. The computation on input $(x,y)$ proceeds as follows. L starts the computation and sends bit $h_L(x,\varepsilon) = w_1$ to R; R returns $w_2 = h_R(y,w_1)$; L returns $w_3 = h_L(x,w_1w_2)$, ... until $w_1 \dots w_{k(x,y)} \in \text{dom } a$. At this point the computation stops with the result $a(w_1 \dots w_{k(x,y)}) = f(x,y)$. $k(x,y)$ is the length of the computation.
The deterministic two-way communication complexity of f is defined by

$$C_{det}(f, 1 \leftrightarrow 2) = \min_{A} \max_{\substack{x \in X \\ y \in Y}} k_A(x,y)$$

where the minimum is taken over all deterministic algorithms which compute f and $k_A(x,y)$ is the length of the computation on input x,y when algorithm A is used.

The model described above was introducted by Yao. Yao related $C_{det}$ (f, 1<->2) with the decompositon number of 0-1 matrices and determined the communication complexity of almost all functions in the deterministic and the probabilistic model. However, he obtained results about only a few concrete functions; e.g. the identity function. In theorem 1 below, we describe a more generally useful lower bound technique.

<u>Definition:</u> Let k be a ring and let $k^{(n,m)}$ be the set of n by m matrices with entries in k. The rank of $A \in k^{(n,m)}$ over k is defined by

$$\text{rank}_k(A) = \min\{p;\ \exists\ C \in B^{(n,p)},\ D \in B^{(p,m)} : A = C \cdot D\}$$

We can now state the main theorem of this section.

<u>Theorem 1:</u> Let $f : X \times Y \to \{0,1\}$ be a function, let $F = (f(x,y))_{x \in X, y \in Y}$ be the 0-1 matrix associated with f, let $\mathbb{N}$ be the ring of integers and let k be any field. Then

$$C_{det}(f, 1 \leftrightarrow 2) \geq \log \text{rank}_{\mathbb{N}}(F) \geq \log \text{rank}_k(F)$$

<u>Proof:</u> (sketch). The second inequality is based on the fact that $\text{rank}_{\mathbb{N}} F \geq \text{rank}_k F$ for any field k and any 0-1 matrix F.

For the proof of the first inequality we need one more concept : one-way nondeterministic unambiguous computations. This concept was studied previously by the second author in the context of finite automata
A nondeterministic one-way algorithm is given by a set $H_L(x) \subseteq B^*$ for every $x \in X$ and a response function $b : Y \times \{0,1\}^* \to \{0,1\}$. On input $(x,y)$, L sends some $w \in H_L(x)$ to R and then R outputs $b(y,w)$. We assume that $\bigcup_{x \in X} H_L(x)$ is prefix-free because only this will alow R to know when L has completed transmission. The algorithm computes f if $f(x,y) = 1$ iff $\exists w\ w \in H_L(x)$ and $b(y,w) = 1$.

The complexity of a nondeterministic algorithm is defined as usual by

$$\max_{\substack{x \in X, y \in Y \\ f(x,y)=1}} \min\{|w|;\ w \in H_L(x) \text{ and } b(y,w) = 1\}$$

A nondeterministic algorithm is <u>unambiguous</u> if accepting computations are unique, i.e.

for all x,y

$\exists w\ w \in H_L(x)$ and $b(y,w) = 1$

$\Rightarrow \exists!\ w\ w \in H_L(x)$ and $b(y,w) = 1$

We use $C_{unamb}(f, 1 \to 2)$ to denote the complexity of f with respect to one-way unambiguous nondeterministic computations. Our proof of the first inequality is based on the following two lemmas.

Lemma 1: $C_{det}(f, 1{<}{-}{>}2) \geq C_{unamb}(f, 1 \to 2)$

Lemma 2: $C_{unamb}(f, 1 \to 2) \geq \log \mathrm{rank}_{\mathbb{N}} F$.

Proof of lemma 1: Let $(h_L, h_R, a)$ be a deterministic algorithm for f. We simulate it by a nondeterministic one-way algorithm as follows. L sends bit $w_1 = h_L(x,\varepsilon)$, then it guesses and sends the response $w_2$ of R, then it sends $w_3 = H_L(x,w_1w_2)$, ... until $w_1 \dots w_k \in$ dom a. Upon Receiving $w_1 \dots w_k$ R checks whether L guessed correctly, i.e. $w_{2i} = h_R(y,w_1 \dots w_{2i-1})$ for $2i \leq k$, and if so R outputs $b(y,w_1 \dots w_k) := a(w_1 \dots w_k)$. If not, R outputs 0.

The simple but crucial observation is that the algorithm described above is unambiguous, since for every x and y there is at most one w, namely the deterministic computation on (x,y), such that L can send w and R outputs 1 after receiving w. □

Proof of lemma 2: Let $k = C_{unamb}(f, 1 \to 2)$ and let $(H_L(x))_{x \in X}$ and b be an unambiguous one-way algorithm for f with complexity k. Let $W = (\bigcup_{x \in X} H_L(x)) \cap (\bigcup_{i \leq k} B^i)$. Then $|W| \leq 2^k$ since $\bigcup_{x \in X} H_L(x)$ is prefix-free.

Claim 1: for all $x \in X, y \in Y$

$$f(x,y) = \sum_{w \in W} [w \in H_L(x)] \cdot b(y,w)$$

(here $[w \in H_L(x)] = 1$ if $w \in H_L(x)$ and 0 otherwise).

Proof of claim 1: If $f(x,y) = 0$ then there is no w such that $w \in H_L(x)$ and $b(y,w) = 1$. If $f(x,y) = 1$ then there is exactly one w such that $w \in H_L(x)$ and $b(y,w) = 1$. Furthermore, this w has length at most k.

The claim above immediately gives rise to a matrix equation, namely

$F = H \cdot K$

where

$H = ([w \in H_L(x)])_{x \in X, w \in W} \in B^{(|X|,|W|)}$

and

$K = (b(y,w))_{w \in W, y \in Y} \in B^{(|W|,|Y|)}$.

Hence rank $F \leq |W|$ or

$C_{unamb}(f, 1 \to 2) = k \geq \log \mathrm{rank}_{\mathbb{N}} F$. □

We note in passing that lemma 2 holds true with equality and that lemma 1 holds true with equality if $C_{det}(f, 1{<}{-}{>}2)$ is replaced by $C_{unamb}(f, 1{<}{-}{>}2)$.

The second inequality in theorem 1 is important because the standard tricks of linear algebra can be used to determine the rank of a matrix over a field k.

Finally, note that the claim in the proof of lemma 2 is false for general nondeterministic computations. It can be replaced by: for all $x \in X$

$$\{y;\ f(x,y) = 1\} = \bigcup_{w \in H_L(x)} \{y;\ b(y,w) = 1\}.$$

This equality can be used to prove

$C_{ndet}(f, 1{<}{-}{>}2) \geq \log \log \mathrm{nrow}\, F$

where nrow F is the number of different rows in matrix F.

We close this section with a brief outline of an alternate proof of theorem 1 due to M. Paterson.

Alternate proof of theorem 1: The alternate proof of theorem 1 is based on the following fact.

**Fact:** Let k be a ring, $A \in k^{(n,m)}$, $B \in k^{(n,p)}$ and $C \in k^{(q,m)}$. Then

$$\mathrm{rank}_k((A\ B)) \le \mathrm{rank}_k(A) + \mathrm{rank}_k(B)$$

and

$$\mathrm{rank}_k(\binom{A}{C}) \le \mathrm{rank}_k(A) + \mathrm{rank}_k(C),$$

here (A,B) denotes the matrix obtained by writing A and B side by side.

**Proof:** If $A = E_1 \cdot D_1$ and $B = E_2 \cdot D_2$ then

$$(A\ B) = (E_1\ E_2) \cdot \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}$$ □

Consider any two-way deterministic algorithm (g,h,a) for computing f. We define matrix $F_w$, $w \in \{0,1\}^*$, by induction of $|w|$ as follows:

1) $F_\varepsilon = F$

2) if $|w| = 2\ell$ then $F_{w0}(F_{w1})$ is obtained from $F_w$ by selecting rows x with $g(x,w) = 0$ $(g(x,w) = 1)$.

3) if $|w| = 2\ell + 1$ then $F_{w0}(F_{w1})$ is obtained from $F_w$ by selecting columns y with $h(y,w) = 0$ $(h(y,w) = 1)$.

Note that

$$\max(\mathrm{rank}_{\mathbb{N}}(F_{w0}), \mathrm{rank}_{\mathbb{N}}(F_{w1})) \ge \mathrm{rank}_{\mathbb{N}}(F_w)/2$$

by the fact above and that $w \in \mathrm{dom}\ a$ implies that $F_w$ is a constant matrix and hence $\mathrm{rank}_{\mathbb{N}}(F_w) \le 1$. Thus the complexity of any deterministic algorithm for f is at least $\log \mathrm{rank}_{\mathbb{N}}(F)$. □

## III. Las Vegas is better than Determinism for Communication Complexity

We exhibit a function f such that

$$C_{ndet}(f, 1 \leftrightarrow 2),\ C_{ndet}(\bar{f}, 1 \leftrightarrow 2),$$

$$C_{Las\ Vegas}(f, 1 \leftrightarrow 2) \ll C_{det}(f, 1 \leftrightarrow 2)$$

$\bar{f}$ denotes the complement of f, i.e. $\bar{f}(x,y) = 1 - f(x,y)$. Las Vegas algorithms use internal randomization, i.e. persons L and R use coin tosses in order to compute their responses. Las Vegas algorithms are required to always produce the correct result, i.e. the probability of error is zero. The complexity is the expected number of bits exchanged (maximized over all inputs (x,y)).

Our result shows that in the realm of distributed computing (and VLSI and TM) not only nondeterminism but even Las Vegas is provably better than determinism.

**Definition:** Let $n \in \mathbb{N}$ and let $X = Y = [0..2^n-1]^n$. For $x = (x_1,\dots,x_n) \in X$ and $y = (y_1,\dots y_n) \in Y$ let

$$f(x,y) = \begin{cases} 1 & \text{if } \exists i \quad x_i = y_i \\ 0 & \text{if } \forall i \quad x_i \ne y_i \end{cases}$$

X and Y can be thought of n numbers of n bits each. L and R are each given a list of n numbers. They have to find out whether the two lists agree in at least one position.

**Theorem 2:**

a) $C_{det}(f, 1 \leftrightarrow 2) \ge n^2$

b) $C_{ndet}(f, 1 \leftrightarrow 2) = O(n + \log n)$

c) $C_{ndet}(\bar{f}, 1 \leftrightarrow 2) = O(n \log n)$

d) $C_{Las\ Vegas}(f, 1 \leftrightarrow 2) = O(n(\log n)^2)$

**Proof:** (sketch). a) Note that $|X| = |Y| = 2^{(n^2)}$. Let $\bar{F}$ be the 0-1 matrix associated with $\bar{f}$. In view of theorem 1, it suffices to show $\mathrm{rank}_{GF(2)} F \ge 2^{(n^2)}$ where GF(2) is the field of characteristic two. We use ⊕ to denote addition mod 2. We transform $\bar{F}$ into the identity matrix by means of linear transformations.

**Lemma:** Let $w_1,\dots,w_n,y_1,\dots,y_n \in [0\dots2^n-1]$. Define $g(w_1,\dots,w_n,y_1,\dots,y_n) :=$

$$\bigoplus_{\substack{x_1 \\ x_1 \ne w_1}} \cdots \bigoplus_{\substack{x_n \\ x_n \ne w_n}} \bar{f}(x_1,\dots,x_n,y_1,\dots,y_n)$$

Then

$$g(w_1,\dots,w_n,y_1,\dots,y_n) = \begin{cases} 1 & \text{if } \forall i\ w_i = y_i \\ 0 & \text{otherwise} \end{cases}$$

Proof: Note that

$$g(w_1,\dots,w_n,y_1,\dots y_n) = |\{(x_1,\dots,x_n);\ x_1 \notin \{y_1,w_1\},\dots,x_n \notin \{y_n,w_n\}\}| \bmod 2$$
$$= \prod_{i=1}^{n} (2^n - |\{y_i,w_i\}| \bmod 2 \qquad \square$$

Matrix $G = (g(w,y))_{w\in X, y\in Y}$ is obtained from F by adding rows. Hence $\text{rank}_{GF(2)}(G) \le \text{rank}_{GF(2)} F$. Also G is the identity matrix and hence $\text{rank}_{GF(2)}(G) = 2^{(n^2)}$.

b) L guesses an $i \in [1\dots n]$ and sends i and $x_i$ to R. If $x_i = y_i$ then R outputs 1.

c) For every $i \in [1\dots n]$, L guesses $j_i \in [1\dots n]$ and sends $j_i$ and the $j_i$-th bit of $x_i$ to R; $O(n\cdot\log n)$ bits altogether. R outputs 1 if the $j_i$-th of $x_i$ and $y_i$ differ for all i.

d) The algorithm is based on the known random algorithms for inequality testing (cf. Freivalds). Whilst these algorithms have non-zero probability of error, we can achieve zero probability of error for our problem as follows. We recall the following fact.

Fact: Let $p_1,\dots,p_m$ be the primes $\le n$. Then $u,v \in [0\dots 2^n-1]$, $u \ne v$, implies

$$|\{j;\ u \not\equiv v \bmod p_j\}| \ge m/2 \qquad \square$$

The algorithm is as follows. Both L and R have the list $p_1,\dots,p_m$.

for i from 1 to n
do

(A) L selects r (to be determined later) primes $p_{j_1},\dots,p_{j_r}$ at random from the list and sends $j_1,\dots,j_r$ and $x_i \bmod p_{j_1},\dots,x_i \bmod p_{j_r}$ to R.

R checks whether $x_i = y_i \bmod p_{j_\ell}$ for all $\ell \in [1\dots r]$.

(B) If this is the case then R asks for and gets the complete $x_i$ from L. If $x_i = y_i$ then R stops and outputs 1.

od
Stop and Output 0.

It is obvious that this algorithm computes $f(x,y)$ with zero probability of error. In part (A) $O(r\cdot\log n)$ bits are sent for every $i \in [1\dots n]$, in part (B) $O(n)$ bits are sent. If $x_i = y_i$ then part (B) is always executed; but the case $x_i = y_i$ occurs only once during the execution of the algorithm. If $x_i \ne y_i$ then part (B) is reached with probability $2^{-r}$. Hence if $x_i \ne y_i$ then the expected number of bits sent in (A) and (B) is $r \log n + 2^{-r}\cdot n$. Thus the total number of bits exchanged is bounded by

$$n(r \log n + 2^{-r}\cdot n) + n$$

Choosing $r = \log n$ proves part d). $\square$

It is worth noting that the Las Vegas algorithm described above never exchanges more than $O(n^2)$ bits.

Theorem 2 is readily transfered to multitape TM under the complexity measure $S\cdot T$. Let

$$L = \{x_1 ¢\dots¢\ x_n \$^{(n^2)} y_1 ¢\dots¢\ y_n;\ |y_i| = |x_i| = n \text{ and } x_i \ne y_i \text{ for all } i\}$$

Theorem 3: Let L as above. Then

a) $S\cdot T_{det}(L) = \Omega(N^2)$

b) $S\cdot T_{\text{Las Vegas}}(L) = O(N(\log N)^2)$

, here N is the length of the input.

Proof: Omitted. $\square$

## IV. Las Vegas is Better than Determinism in VLSI

In this section we will transfer the result of the previous section to VLSI computation under the complexity measure $AT^2$. We will have to overcome one problem. Whilst the partition of the inputs into sets X and Y was predefined so far, this is not the case for VLSI computations. It is up to the chip designer where he wants to read in inputs. We overcome this difficulty by modifying our function somewhat.

Consider $f_1(z_1, \ldots, z_{2n}, s_1, \ldots, s_{2n}, k_1, \ldots k_{n/4})$ where $z_1, \ldots, z_{2n} \in \{0,1\}^n$ are bitstings of length n, $s_1, \ldots, s_{2n} \in \{0,1,X\}$ are selection inputs and $k_1, \ldots, k_{n/4} \in [0 \ldots n-1]$ are shift inputs ($f_1$ can be considered as a function of $2n \cdot n + 2n \cdot 2 + n/4 \cdot \log n = O(n^2)$ binary inputs). We will assume that exactly n/4 of the selection inputs are set to 0, say $s_{i_1}, \ldots, s_{i_{n/4}}$, and exactly n/4 are set to 1, say $s_{j_1}, \ldots, s_{j_{n/4}}$. Then

$$f_1 = 1 \text{ iff } z_{i_\ell} = \text{shift}(z_{j_\ell}, k_\ell) \quad \text{for some } \ell \in [1 \ldots n/4]$$

Here shift $(z_{f_\ell}, k_\ell)$ denotes the cyclic shift of bitstring $z_{j_\ell}$ by $k_\ell$ positions.

We use $AT^2_{XYZ}$, XYZ$\in$ {det, Las Vegas, nondet}, to denote the complexity measure $AT^2$ for where-oblivious XYZ-Chips.

**Theorem 3:** Let N be the number of binary inputs of $f_1$. Then

a) $A \cdot T^2_{\text{det}}\ (f_1) = \Omega(N^2)$

b) $A \cdot T^2_{\text{ndet}}\ (f_1) = O(N^{3/2} \text{ poly } (\log N))$

c) $A \cdot T^2_{\text{ndet}}\ (\overline{f}_1) = O(N^{3/2} \text{ poly } (\log N))$

d) $A \cdot T^2_{\text{Las Vegas}}\ (f_1) = O(N^{3/2} \text{ poly } (\log N))$

Proof: (sketch). We only sketch proofs for a) and d), b) and c) being simpler than d).

a) Consider any where-oblivious deterministic chip for $f_1$; say the chip has area A and time T. Consider a cut of length $O(\sqrt{A})$ which cuts the chip in half according to the $2n^2$ input bits which comprise the z's. Let L and R be the two halves of the chip. Let $I_1 = \{i \in [1 \ldots 2n]$; at least n/3 of the bits of $z_i$ come through ports in L$\}$ and $I_2 = \{j \in [1 \ldots 2n]$; at least n/3 of the bits of $z_j$ come through ports in R$\}$.

**Claim 3:** $|I_1|, |I_2| \geq n/2$

Choose $i_1, \ldots, i_{n/4} \in I_1$ and $j_1, \ldots, j_{n/4} \in I_2$ such that these n/2 numbers are pairwise different. Set $s_{i_1} = \ldots = s_{i_{n/4}} = 0$, $s_{j_1} = \ldots = s_{j_{n/4}} = 1$. This choice of the s-inputs will make sure that the z's which have to be compared are read in (at least partly) at different sides of the cut. We will use the shift inputs to make sure that many bits have to be transported across the cut.

For every $\ell$, $1 \leq \ell \leq n/4$, let $A_\ell (B_\ell) \subseteq [0 \ldots n-1]$ be the set of bit positions in $z_{i_\ell}$ $(z_{j_\ell})$ which come through ports in L(R).

**Claim 4:** $\exists k_\ell \in [0 \ldots n-1]$:

$$|\{t;\ t \in A_\ell \text{ and } (t+k_\ell) \bmod n \in B_\ell\}| \geq n/9$$

□

Set $k_1, \ldots, k_{n/4}$ as given by claim 4. What will that do for us? The chip has to decide whether there is an $\ell$ such that $z_{i_\ell}$ = shift $(z_{j_\ell}, k_\ell)$. But by claim 4 at least n/9 of the corresponding input bits of $z_{i_\ell}$ and shift $(z_{j_\ell}, k_\ell)$ come through ports at opposite sides of the cut. Hence $\Omega(n^2)$ bits have to flow across the cut by theorem 2 and hence $T = \Omega(n^2/\sqrt{A})$. This shows $AT^2 = \Omega(n^4) = \Omega(N^2)$.

d) We describe a Las Vegas chip for $f_1$ with area $O(n^2)$ and time $O(\sqrt{n}$ poly $(\log n))$. Thus $A \cdot T^2_{\text{Las Vegas}}(f_1) = O(n^3 \text{ poly}(\log n)) = O(N^{3/2} \text{ poly}(\log N))$.

For every $i \in [1...2n]$ the chip has a n-bit register to store $z_i$, a shifter, a divider, a table of the primes $\leq n$ and a randomizer. The randomizer generates random numbers in $[1...n]$ which are used to select primes from the table. It assumed that a randomizer with area $A = O(n)$ and $T = \sqrt{n}$ exists (This is part of the Las Vegas model of VLSI; if one assumes that a random bit can be generated in $A = O(1)$ and $T = O(1)$ then a randomizer with $A = O(\log n)$ and $T = O(1)$ would exist. So our assumption is very weak). Shifters and Multipliers can be done in $A = O(n)$ $T = O(n^{1/2})$ [Preparata/Vuillemin] and the use of Newton Iteration will turn a Multiplier into a Divider with $A = O(n)$, $T = O(n^{1/2})$. Also the table of primes has area $O(n)$ and access time $O(\log n)$.

The 2n registers are connected in two ways with one another. Firstly of all, via a permutation network with 2n inputs ($A = O(n^2)$, $T = \text{poly}(\log n)$) and second of all, the 2n registers are used as a random access memory. ($A = O(n^2)$, access time = poly $(\log n)$).

The chip operates as follows. First, the s and k inputs are read in and are distributed to the 2n registers for the z's. Time poly $(\log n)$ certainly suffices for this. Next the required shifts are carried out in parallel ($T = \sqrt{n}$). Next we go through the following cycle $(\log n)$ times. The randomizer + the table of primes generate a random prime ($T = \sqrt{n}$ poly $(\log n)$ suffices by our assumption), the divider modul is used to compute the moduli ($T = \sqrt{n}$ suffices) and the moduli are sent one bit at a time over the permutation network ($T = \text{poly}(\log n)$). Note that the n/4 subproblems can be all worked on simultaneously. At this point $O(1)$ pairs $(i_\ell, j_\ell)$ with $z_{i_\ell} \neq \text{shift}(z_{j_\ell}, k_\ell)$ will survive all the tests (on the average). Using the random access memory the pairs $z_{i_\ell}$ and shift $(z_{j_\ell}, k_\ell)$ will be compared for actual equality one after the other ($T = O(1)$ poly $(\log n)$ on average).

Altogether, the chip operates in area $A = O(n^2)$ and expected time $T = O(\sqrt{n}$ poly $(\log n))$. □

## V. Concluding Remarks

We described a new proof method for proving lower bounds on deterministic information transfer and applied this method to TM and VLSI computations. The method is strong enough to distinguish deterministic and Las Vegas computations.

Furthermore, the lower bound technique of theorem 1 can be applied to more problems. For example, it can be used to show lower bounds for matching problems in bipartite graphs. These lower bounds are of a higher order than the corresponding nondeterministic upper bounds.

## References

R. Freivalds; Probabilistic machines can use less running time, Information Processing 1977, IFIP, North Holland, 1977, 839-842.

R.J. Lipton, R. Sedgewick; Lower Bounds for VLSI, 13th ACM Symposium on Theory of Computing, 1981, 3oo-3o7.

F.P. Preparata, J.E. Vuillemin; Area-Time Optimal VLSI Networks for Computing Integer Multiplication and Discrete Fourier Transform, ICALP 81, LNCS 115, 29-4o.

E.M. Schmidt; Succinctness of Description of Context Free, Regular and Unambignous Languages, Ph. D. thesis, Cornell University, 1978.

A.C. Yao; Some complexity questions related to distributive computing, 11th ACM Symposium on Theory of Computing, 1979, 2o9-213.

A.C. Yao; The Entropic Limitations on VLSI computations, 13th ACM Symposium on Theory of Computing, 1981, 3o8-311.