Towards a Strong Communication Complexity theory or Generating Quasi-Random Sequences from Two Communicating Slightly-random Sources.

(Extended Abstract)

Umesh V. Vazirani [1]

University of California
Berkeley, CA 94720.

1. Introduction.

Several computational applications, such as randomizing algorithms [Ra], stochastic simulation [Sc, KG] and cryptographic protocols [Bl1, GM, VV] assume a fast source of unbiased, independent coin flips. Unfortunately, the available physical sources of randomness, such as noise diodes and geiger counters are at best imperfect [Mu]. Santha and Vazirani [SV] consider a very general model for such imperfect sources of randomness: the **slightly random source**. A slightly random source is a black box which outputs 0's and 1's. The output of the source is "slightly random" in that the next output bit is 0 with probability at least δ, and 1 with probability at least δ. Thus the output of the source cannot be deterministic; it must have "some randomness" in it. On the other hand, it is clear that the assumptions on the source are not strong enough to obtain truly random bits from its output. Santha & Vazirani [SV] showed how to use $\Omega(\log n \log^* n)$ slightly-random sources working in parallel to produce n-bit **quasi-random sequences.** Such sequences have the property that they cannot be distinguished from the flips of a fair coin in a very strong sense. The strong indistinguishability allows for the substitution of quasi-random sequences in place of fair coin flips in computational applications.

[1] This research was supported by NSF-Grant MCS 82-04506, and by the IBM Doctoral Fellowship.

In this paper we show how to generate quasi-random sequences using only **two** slightly-random sources. This result is the best possible in the sense that [SV] prove that no algorithm can produce a better than $1-\delta$ biased bit given a **single** slightly-random source. To prove that two sources are enough, we must allow the sources to follow "strategies" of their choice in their attempt to make the output of the algorithm look "non quasi-random". Thus, a slightly-random source may be thought of as an **adversary** source. The adversary in each source can see the sequence generated by his source so far. The next bit output is the flip of a coin whose bias is fixed by the adversary. To ensure that the output is "slightly random", the adversary is restricted to picking a bias between δ and $1-\delta$, where δ is a constant between 0 and 1. This bias is the conditional probability that the next bit is 1 given the sequence output so far. The algorithm for generating n-bit quasi-random sequences from 2 slightly-random sources is very simple: let x and y denote $\Omega(\log n \log^* n)$ bit outputs of the two sources. Then one bit of the output is simply $b(x, y) = \vec{x} \cdot \vec{y}$, where $\vec{x}$ and $\vec{y}$ are bit vectors corresponding to x and y, and $\cdot$ denotes GF(2) inner product (i.e. all additions and multiplications are done mod 2). The simplicity of this function makes for efficient implementation in practice. Moreover, it is economical in its use of slightly-random bits: the compression factor matches the lower bound proved in [SV] to within a constant factor. This economy of slightly-random bits utilization is also an important practical factor. Proving that a function is good for extracting quasi-random bits involves showing that for every A_1 strategy and for every A_2 strategy (A_i is the adversary associated with the i^{th} source), the bit b(x,y) is almost unbiased. For this reason, even though the function f is very simple, the proof that it works is not straightforward.

The extraction of almost unbiased bits from 2 slightly-random sources is intimately related to **distributional communication complexity theory.** In section 3, we show that the current definition of distributional communication complexity [Ya2] of a boolean function is not robust. We exhibit functions whose communication complexity is high under the current definition, but whose communication complexity is drastically reduced if the input distribution is altered. On the other hand, the **inner product function** $(\vec{x} \cdot \vec{y})$ appears to have high communication complexity despite such alterations. We propose a stronger definition of communication complexity which distinguishes between the genuinely high communication complexity functions such as the inner product function, and the spurious examples. As a consequence of a Hierarchy Theorem under this definition, we prove that two **communicat-**

ing slightly-random sources can be used to generate quasi-random sequences (i.e. the "physical shielding" between the two sources need not be perfect). Finally, as a corollary to the above theorems, we answer an open question of Yao [Ya2] about the distributional communication complexity of the set intersection function.

1.2 History and Practical Implications:

The simplest model of an imperfect source of randomness is a biased coin. Von Neumann [vN] gave a very simple and fast algorithm to generate unbiased bits from the flips of biased coin, even if the bias is unknown. Blum [Bl2] models an imperfect random source as a deterministic finite state Markov process. This source may be regarded as a collection of a finite number of biased coins, each corresponding to a state of the Markov Process. Blum shows that the obvious generalization of the von Neumann procedure (i.e. apply the von Neumann procedure to each state of the Markov Process separately) does not work. He shows that surprisingly enough, by changing the order in which the bits are output, independent unbiased coin flips can be obtained! The method can be extended to generating bits in linear time at a rate almost achieving the entropy of the source [El] Santha and Vazirani [SV] proposed a very general model for an imperfect source of randomness: the slightly-random source. This is the model which is considered in this paper. They defined the notion of quasi-random sequences and showed how to obtain quasi-random sequences from the outputs of slightly-random sources. However, the number of slightly-random sources needed by their procedure grows logarithmically in n, the length of the quasi-random output sequence. Reducing the number of slightly-random sources to just two is a very important practical consideration.

Based on minimal assumptions about the physical source of randomness, the method proposed in this paper provably generates good (quasi-random) sequences. What makes this method even more exciting is that it may also speed up the generation of random bits from the physical source. This is because currently the physical source is sampled very slowly in the hope that successive bits sampled are almost independent. Instead, using our method, the physical source could be sampled more frequently (it would still remain slightly-random). This increased sampling frequency would make up for the bit-compression done by the algorithm.

2. Two Slightly-Random Sources are Enough:

2.1 Statement of the Problem:

We are given two slightly-random sources, with their associated adversaries A_1 and A_2. Each adversary knows the sequence of bits output by his source so far, and can pick the conditional probability that the next output bit is 1 to be any number between δ and $1-\delta$. Neither adversary can see the sequence generated by the other source (this corresponds to "shielding" the sources from each other. In the next section we show that even if the shielding is not perfect, one can still extract quasi-random sequences). By Theorem 3 of Santha & Vazirani [SV], it suffices to convert the outputs of the two sources into almost unbiased bits: they show that if the bias of each bit in the sequence lies in the range $[\frac{1}{2}-\varepsilon(n), \frac{1}{2}+\varepsilon(n)]$ and if $\varepsilon(n) = O(1/n^t)$ for every $t > 0$, then the resulting n-bit sequence is quasi-random, even though the adversary determines the exact bias of each bit. Let x and y be k bit strings produced by the two sources. We would like to demonstrate a boolean function b(x, y) with the property that for any $\varepsilon(n) > 0$, there is a k large enough so that the bit b(x, y) has bias in the range $[\frac{1}{2}-\varepsilon(n), \frac{1}{2}+\varepsilon(n)]$

2.2 Xor Does not Work:

Let b(x, y) be the xor of the bits of x and y. Then A_1 and A_2 have strategies such that the probability that the output b(x, y) is 0 is at least $1-2\delta$ **independent** of k (recall that |x| = |y| =k). Both adversaries follow the following simple strategy: the first k-1 bits are independent and unbiased. The adversary computes the xor, u, of the k-1 bits that he has generated so far, and for the k^{th} bit he flips a coin which has $1-\delta$ bias towards u. Each adversary succeeds in setting the xor of his bits to 0 with probability $1-\delta$. If both adversaries succeed then b(x, y) is 0 and this happens with probability exceeding $1-2\delta$.

Why did the xor function fail? The problem was that the effect of x on the function can be summed up in just one bit: the xor of the bits of x. i.e. the communication complexity of the xor function is just one bit. This relationship between extracting good bits from two sources and communication complexity is explored in Section 3.

2.3 The Inner-Product Function Works:

Let $b(x,y) = \vec{x}\cdot\vec{y}$, where $\vec{x}$ and $\vec{y}$ are just x and y represented as bit vectors, and $\cdot$ denotes GF(2) inner product. Let k be the lengths of x and y. If the number of bit positions in which both x and y have a 1 is even then b(x, y) is 0, and if it is odd then b(x, y) is 1. Intuitively, this function is good because even though each

adversary knows the positions in which he has generated 1's so far, he does not know how many of them will pair up with 1's from the other source and how many with 0's. Therefore, as k gets large, the adversary loses track of how the previously generated bits affect the inner product function, and his control over the next bit is useless.

The problem with formalizing this intuition is the possibility that the adversaries can implicitly decrease their uncertainty about the effect of the previously generated bits. For example, consider the modified scenario in which the first $k/2$ bits of each source are generated by the flips of a fair coin. The next $k/2$ bits are deterministically output by the adversary, who can of course see the first $k/2$ bits. In this setting, do the adversaries have a strategy to bias the inner-product function? Surprisingly, even though neither adversary has any idea what the inner-product of the first $k/2$ bits of x and y is, they have a strategy which **always** sets b(x, y) to 0. Let u be the $k/2$ bits that were generated by the fair coin flips. The adversary simply outputs the string u again (deterministically).

Theorem 1: Let $k = C(\delta) \log 1/\varepsilon$, where $C(\delta) = \Omega(\frac{1}{\delta^2}\log\frac{1}{\delta})$. Then for every A_1 strategy and for every A_2 strategy, b(x,y) has bias in the range $[1/2-\varepsilon, 1/2+\varepsilon]$.

Proof: Fix any strategy S for A_1. We show that A_2 has no corresponding strategy that biases the output more than ε away from 1/2.

Definition: **Label(y)** with respect to strategy S is defined as Pr[b(x, y) = 1 | x was produced using strategy S].

The proof is divided into two parts. In the first part we prove that the labels of "most" strings lie in the range $[1/2-\varepsilon/2, 1/2+\varepsilon/2]$. Call the strings with labels outside the $[1/2-\varepsilon/2, 1/2+\varepsilon/2]$ range, the "bad strings". In the second part of the proof, we show that that the bound proved in first part is strong enough that even if the adversary A_2 were allowed to decide which strings are bad, his probability of hitting a bad string is at most $\frac{\varepsilon}{2}$.

The first assertion is proved by induction on k.

Let γ denote the probability that the first bit output by A_1 is 1, $\delta \le \gamma \le 1-\delta$.

Let S_1 denote the k-1 bit strategy used if the first bit is 1,

and S_0 denote the k-1 bit strategy used if the first bit is 0.

Consider the strings x = 0z & x' = 1z, where z is a k-1 bit string.

Let the label of z with respect to the strategy S_0 be $1/2+\Delta_0$,

and let the label of z with respect to the strategy S_1 be $1/2+\Delta_1$.

Then the label of 0z w.r.t. S is

$$(1-\gamma)(1/2+\Delta_0) + \gamma(1/2+\Delta_1).$$

$$= 1/2+(1-\gamma)\Delta_0+\gamma\Delta_1.$$

And the label of 1z w.r.t. S is

$$(1-\gamma)(1/2+\Delta_0) + \gamma(1/2-\Delta_1).$$

$$= 1/2+(1-\gamma)\Delta_0-\gamma\Delta_1.$$

Notice the averaging in the deviation of the labels from 1/2 in going from k-1 bit strategies to a k-bit strategy. To capitalize on this averaging, we introduce a **potential function** of a k-bit strategy S. The potential function is a measure of the spread away from $\frac{1}{2}$ of the labels of k-bit strings with respect to S. This potential function has two useful properties:

1) There is an upper bound on the potential function that decreases exponentially with k.

2) We prove that if the potential function of strategy S is sufficiently small, there are very few "bad strings" with respect to strategy S.

Definition: Let m be the minimum positive, even, integer such that $(1-\delta)^m < 2^{H(\delta/2)}$, where H is the entropy function. The **potential** of a k-bit strategy S is defined to be $\sum_{|y|=k} \Delta(y)^m$ where $\Delta(y)$ is such that the label of the string y with respect to S is $1/2+\Delta(y)$.

We express below the the potential of the A_1 strategy S in terms of the sub-strategies S_0 and S_1:

potential(S) =

$$\sum_{|z|=k-1} ((1-\gamma)\Delta_0 + \gamma\Delta_1)^m + ((1-\gamma)\Delta_0 - \gamma\Delta_1)^m.$$

$$= 2 \sum_{|z|=k-1} \left((1-\gamma)^m \Delta_0^m + \binom{m}{2}(1-\gamma)^{m-2}\gamma^2\Delta_0^{m-2}\Delta_1^2 + \cdots + \gamma^m\Delta_1^m\right)$$

$$= 2(1-\gamma)^m \sum_{|z|=k-1} \Delta_0^m +$$

$$2\binom{m}{2}(1-\gamma)^{m-2}\gamma^2 \sum_{|z|=k-1} \Delta_0^{m-2}\Delta_1^2 + \ldots$$

$$+ 2\gamma^m \sum_{|z|=k-1} \Delta_1^m.$$

The previous step is justified since γ is independent of z. In the two extreme terms, the factors $\sum_{|z|=k-1} \Delta_0^m$, and $\sum_{|z|=k-1} \Delta_1^m$ are the potentials of the k-1 bit strategies S_0 and S_1 respectively. However, the factors in the middle terms $\sum_{|z|=k-1} \Delta_0^{m-2i}\Delta_1^{2i}$ have no nice interpretation. We use the following generalization of Schwartz's Inequality [Ru]: if m is even, then $\sum_i a_i^{m-i}b_i^i \le \max[\sum_i a_i^m, \sum_i b_i^m]$:

Thus potential(S) $\le 2\left((1-\gamma)^m + \binom{m}{2}(1-\gamma)^{m-2}\gamma^2 + \cdots + \gamma^m\right) \times$

$$max\left(\sum_{|z|=k-1} \Delta_0^m, \sum_{|z|=k-1} \Delta_1^m\right)$$

$$= \left(((1-\gamma)+\gamma)^m + ((1-\gamma)-\gamma)^m\right) \times$$

$$max\left(\sum_{|z|=k-1} \Delta_0^m, \sum_{|z|=k-1} \Delta_1^m\right)$$

Let potential(k) denote the maximum over all k-bit strategies of potential(S).

Then potential(S) $\le \left(1+(1-2\gamma)^m\right)$ potential(k-1)

$$\le 2^{H(\delta/2)} \text{ potential (k-1)}$$

The previous step follows from the definition of m, and the fact that γ lies between δ and $1-\delta$.

Since the above inequality holds for every k-bit strategy S, it follows that

$$\text{potential}(k) \le 2^{H(\frac{\delta}{2})}\,\text{potential}(k\text{-}1).$$

$$\text{Hence potential}(k) \le \left[2^{H(\frac{\delta}{2})}\right]^k.$$

Recall that a k bit sequence is bad if it has a label which differs from 1/2 by more than $\varepsilon/2$. Each bad sequence contributes atleast $\left[\frac{\varepsilon}{2}\right]^m$ to the potential function.

Therefore the number of bad sequences

$$\le \frac{potential(k)}{\left[\frac{\varepsilon}{2}\right]^m}$$

$$\le \frac{\left[2^{H(\frac{\delta}{2})}\right]^k}{\left[\frac{\varepsilon}{2}\right]^m}$$

This completes the first part of the proof of the theorem. For the second part, we must show the following:

If the adversary A_2 chooses any set, B, of (bad) k-bit strings of cardinality $\frac{\left[2^{H(\frac{\delta}{2})}\right]^k}{\left[\frac{\varepsilon}{2}\right]^m}$ and any strategy for generating a string from the set B, then he has probability at most $\frac{\varepsilon}{2}$ of generating a string from B.

Call a strategy **extreme** if each bit is output by the flip of a δ or $1-\delta$ biased coin.

Lemma 1: For every set B, the adversary A_2 has an optimal (maximizing his chance of generating a string from B) strategy which is extreme.

Lemma 2: Let S and S' be any extreme strategies. Let B be any set of strings. Then there is a set B' such that |B| = |B'|, and the probability that S' hits B' is equal to the probability that S hits B.

By the above lemmas, we may fix A_2's strategy to be any extreme strategy, without loss of generality. Fix A_2's strategy be "always bias $1-\delta$ towards 1". Then the source is simply a biased coin. It remains to show that the |B| most likely k bit sequences produced by flips of this biased coin have total probability at most $\frac{\varepsilon}{2}$. The proof is by standard probabilistic methods and uses the strong bound on binomial tails due to Bernstein [Re], and will be presented in the final paper.

qed Theorem 1.

3. Strong Communication Complexity and Two Communicating Sources:

3.1 Distributional Communication Complexity:

Suppose that two distant computers must compute a boolean function f(x, y). The first computer receives x, and the second y, each an n-bit binary string. What is the minimum number of bits that they must exchange if they alternately send each other information a bit at a time? This question was raised by Yao [Ya1], and later studied by Lipton and Sedgewick [LS], Melhorn and Schmidt [MS], and Papadimitriou

and Sipser [PS]. The strongest model for communication complexity was defined by Yao [Ya2]: Suppose that the n-bit sequences x and y were generated by the flips of a fair coin. Moreover, the boolean function need be correctly computed with probability at least $1/2+\varepsilon$. Then the minimum number of bits that must be exchanged is the **distributional** two-way communication complexity of f and is denoted by $D_\varepsilon(f)$. In the same paper, Yao exhibited a function that has provably high distributional communication complexity: the multiplication function $l_p(x,y)$. This function is defined to be the least significant bit of $(x \times y)$ mod p, where p is an n-bit prime number. Yao proved that if ε is sufficiently large, the multiplication function has a linear lower bound in communication complexity; i.e. a non-zero fraction of all the bits have to be exchanged to get the correct answer on even $1/2+\varepsilon$ of inputs.

3.2 Relating Communication Complexity and the Extraction of Good Bits From Two Non-Communicating Slightly-random Sources:

Distributional communication complexity asks for the minimum number of bits that must be exchanged to **know** f(x, y) with probability $1/2+\varepsilon$. In extracting bits from two sources, the two source adversaries attempt to **set** f(x, y) to be 1 (or 0) with probability at least $1/2+\varepsilon$. The following theorem establishes a relationship between the hardness of knowing and setting:

Theorem 2: If $D_\varepsilon(f) \leq k$, for any k ≥ 2, then two slightly-random source adversaries have strategies to set f(x, y) to be 1 (or 0) with probability at least $\frac{1}{2}+e^{-c(\delta)k\log k}$, where $c(\delta)$ is a positive constant depending on the source parameter δ.

Theorem 2 shows that high distributional communication complexity is a necessary condition for any function that extracts good random bits from two (non-communicating) sources. The proof is along similar lines as that of Theorem 3 in section 3.4.

Corollary: For every $\varepsilon>0$, $D_\varepsilon(f) = \Omega(\frac{n}{\log n})$, when f is the inner-product function.

The corollary follows immediately from Theorems 1 and 2. This is within a logn factor of the lower bound conjectured in an open problem of [Ya2]:

3.3 Pathological Functions:

Next, we exhibit a function g(x, y) that has high communication complexity in the distributional model, but whose communication complexity drops drastically if the distribution on x and y is changed.

Let n=2m. Let x = x'x'' and y = y'y'' where |x'| = |x''| = |y'| = |y''| = m. x' and y' are used to determine a logm bit pointer, which picks out a bit of x. g(x, y) is defined to be this bit. More pre-

cisely, divide x' and y' into logm blocks, each of length $\frac{m}{logm}$. Let $x_1, \ldots, x_{logm}$ and $y_1, \ldots, y_{logm}$ denote these blocks. Let l(u, v) be a high communication complexity function in the distributional sense, which takes $\frac{m}{logm}$ bit inputs u and v.

Let $b_1 = l(x_1, y_1)$.

$b_{logm} = l(x_{logm}, y_{logm})$.

Let i denote the logm bit number i = $b_1 \ldots b_{logm}$.

Definition: g(x, y) = $(m+i)^{th}$ bit of x.

The function g(x, y) has high communication complexity in the distributional sense. Intuitively, this is because at least $\frac{m}{logm}$ bits must be exchanged to know even one bit of the address i with $1/2+\varepsilon$ certainty. On the other hand since x is picked from the uniform distribution, roughly half the bits of x'' are 0 and roughly half are 1. So the chance of correctly guessing g(x, y) is less than $1/2+\varepsilon$ for n sufficiently large.

However, the high communication complexity of g(x, y) relies critically on x and y being generated by a fair coin. If instead x and y were generated by a biased coin, the communication complexity falls to 0: the computers simply guess the bit towards which the coin is biased, without exchanging any messages, and on the average, they will be correct more often than not. This drastic dependence on the input distribution contrasts sharply with the inner product function, whose communication complexity appears insensitive to the input distribution.

Of course, more general distributions on x and y may be considered, in which the successive bits of x (and y) are not independent. The construction of g(x, y) can be modified to exhibit pathological functions: for example, in the final paper, we shall exhibit a function h(x, y) whose communication complexity is high if x and y are generated by any biased coin. However, if x is generated by a 2 state Markov Process, the communication complexity falls to 0. Once again the communication complexity of the inner product function seems insensitive to this change in distribution.

3.4 A Strong Definition of Communication Complexity:

We would like to say that a function has high communication complexity in a strong sense, if the number of bit exchanges required to compute it remains high under any input distribution. We propose the following strong definition of communication complexity: x and y are generated by two slightly-random sources A_1 and A_2. They would like to set the function f(x, y) to 1 (or 0) with probability at least

$1/2+\varepsilon$. The Strong Communication complexity of the function f(x, y) is the minimum number of bits, k, that the two adversaries must exchange to bias f(x, y) by this amount. We shall denote this strong communication complexity of f(x, y) by $S_{\varepsilon,\delta}(f) = k$, where δ is the parameter associated with the slightly-random sources.

In the above terms, Theorem 1 simply says the following:

Restatement of Theorem 1: $S_{2^{-n/c(\delta)},\delta}(f) > 0$, when f is the inner product function.

Theorem 3: If $S_{\varepsilon,\delta}(f) \leq k$, then $S_{\frac{(1-\alpha)\varepsilon}{4},\alpha\delta}(f) \leq k-1$, where $0 < \alpha < 1$ and $k \geq 1$.

Sketch of Proof: Consider a pair of strategies for the two adversaries that achieves the ε bias in the value of f, using just k bits of communication. Suppose A_1 sends the first bit in the protocol. We shall show that this first bit of communication can be dispensed with, if the adversaries are given more power, to achieve a (smaller) bias in the value of f. Let T_0 be the set of (n-bit) strings on which the first adversary communicates a 0 and let T_1 be the set of strings on which he communicates 1. Clearly, A_2 can start generating his sequence after he receives the first communicated bit. Let S_0 be A_2's strategy if he receives a 0 from A_1, and S_1 if he receives a 1. Let α be the probability that A_1 generates a T_0 string (i.e. sends 0 for the first communicated bit), and $1-\alpha$, the probability that he generates a T_1 string. Let $p_{i,j}$ be the probability that the value of f is 1 given that A_1 generated a T_i string and A_2 followed strategy S_j. Then the bias in the value of f is $|\alpha p_{0,0} + (1-\alpha)p_{1,1}| \geq 1/2 + \varepsilon$. By an averaging argument, either the T_0, S_0 case contributes $\varepsilon/2$ to the bias in the value of f, or T_1, S_1 does. Suppose T_0, S_0 does. Then A_1 does not send A_2 the first bit called for by his protocol, and A_2 follows strategy S_0.

The bias in the value of f is now $|\alpha p_{0,0} + (1-\alpha)p_{1,0}|$. This number may be 1/2, even though the first term contributes at least an $\varepsilon/2$ bias. However, A_1 can use his increased power to overcome this difficulty. He can use this increased power to vary the relative probabilities of the sets T_0 and T_1, while keeping the distributions within each set the same. Then, if the second term cancelled out the bias induced by the first term in the above expression, then after this change in relative probabilities, the biases no longer cancel.

The above theorem trades off the number of bits communicated by making the adversaries (of the slightly-random sources) more powerful (smaller δ), but setting the output with smaller probability of success. This tradeoff, combined with Theorem 1 imply the following lower bound for the strong communica-

tion complexity of the inner product function.

Corollary: For every fixed $0<\varepsilon<1/2$, $S_{\varepsilon,\delta}(\vec{x}\cdot\vec{y}) = \Omega(n/\log n)$.

In other words the inner product function extracts good bits even from two communicating slightly-random sources. This is important from a practical point of view, because it says that even if the shielding between the two physical noise sources is not perfectly designed, the inner-product function will still extract good sequences from their output.

4. Open Questions.

Several interesting questions remain open:

a) Our proof that two sources are enough was tailored to the particular function, the inner product function. Is there a general characterization of functions that are good extractors?

b) Our algorithm for extracting quasi-random bits uses each block of k(n) bit sequence from each source to produce one quasi-random output bit. Consider the following scheme for reusing the blocks of slightly-random bits: denote by c(x) the cyclic shift of the bits of x by one position. We conjecture that the sequence of bits $b_0 = f(x, y)$, $b_1 = f(c(x), y)$, $b_2 = f(c^2(x), y)$, ... , is quasi-random.

c) We prove that the strong communication complexity of the inner product function is $\Omega(\frac{n}{\log n})$. We conjecture that the communication complexity is $\Omega(n)$. One approach towards improving the lower bound might be to prove a sharper hierarchy theorem.

The Bit Extraction Problem:

Consider the following question:

Input: n-bit sequence generated as follows: m of the bits are picked deterministically. The rest of the n - m bits are generated by the flips of a fair coin.

Output: *Without knowing* which bits were deterministically fixed, gererate n - m independent, unbiased bits from the sequence.

A solution to the problem in general would entail outputting $b_1(x_1,\ldots,x_n)$, $b_2(x_1,\ldots,x_n)$, ..., $b_{n-m}(x_1,\ldots,x_n)$, where the b_i's are boolean predicates. The core of the problem lies in showing that no subset of m bits can be fixed so that the n-m output bits become correlated.

The case m = 1 is easy. Let b_i be x_i xor x_n. Then the b_i's are independent, unbiased bits. However, for general m, such xor schemes perform poorly. If m = 2n/3, then we can prove that there is no xor scheme for extracting *even 2 bits;* this is independent of the value of n. What about general boolean functions b_i? We conjecture that there is always an xor scheme

which is optimal for the problem; thus in particular there is no scheme for extracting even two good bits if a third of the input bits are random!

5. Acknowledgements.

I am extremely grateful to: Manuel Blum, for his knack for asking the right questions; to Dick Karp, not only for his insights into probability theory, but also for sharing his excitement about the problem when the proof was elusive; to Gilles Brassard, who discovered a very subtle bug in a previous proof; to Sampath Kannan, for patiently listening to half-baked ideas, and offering valuable suggestions; and to Charles Bennet, Ron Rivest, Steve Rudich, Miklos Santha, Mike Sipser, Vijay Vazirani and Andy Yao for seme very fruitful discussions.

6. References.

[Bl1] M. Blum, "Coin Flipping by Telephone," IEEE COMPCON (1982).

[Bl2] M. Blum, "Independent Unbiased Coin Flips From a Correlated Biased Source: a Finite State Markov Chain," Proc. 25th Ann. Symp. on the Theory of Computing, Oct. 1984, 425-433.

[El] P. Elias, "The Efficient Construction of an Unbiased Random Sequence," Ann. Math. Statist. Vol 43, No. 3, 1972, 865-870.

[GM] S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," 1982 STOC.

[KG] W. Kennedy and J. Gentle, *Statistical Computing*, Marcel Dekker, Inc. New York.

[Kn] D. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA (second edition 1981).

[LS] R. Lipton and R. Sedgewick, "Lower bounds for VLSI," Proc. 13th Ann. ACM Symp. on Theory of Computing, May 1981, 300-307.

[Mu] H. F. Murry, "A general approach for generating natural random variables," IEEE Trans. Comput., vol. C-19, pp. 1210-1213. Dec 1970.

[vN] J. von Neumann, "Various Techniques Used in Connection with Random Digits," Notes by G. E. Forsythe, National Bureau of Standards, Applied Math Series, 1951, Vol 12, 36-38. Reprinted in von Neumann's Collected Works, Vol 5, Pergamon Press (1963), 768-770.

[PS] C. H. Papadimitriou and M. Sipser, "Communication complexity," Proc. 14th Ann. ACM Symp. on Theory of Computing, May 1982, 196-200.

[Ra] M. Rabin, "Probabilistic Algorithms," Algorithms and Complexity, J.Traub, Editor, Academic Press (1976), pp. 21-39.

[Re] A. Renyi, "Probability Theory," North-Holland Series in Applied Mathematics and Mechanics, vol 10, North-Holland Publishing Company, Amsterdam, 1970.

[Ru] W. Rudin, "Principles of Mathematical Analysis," Third Edition, McGraw-Hill Book Company.

[SV] M. Santha and U. V. Vazirani, "Generating Quasi-random Sequences from Slightly-random Sources," Proc. 25th Ann. Symp. on the Theory of Computing, Oct. 1984, 434-440.

[Sc] B. Schmeiser, "Random Variate Generation: A Survey," 1980 IEEE. Simulation with Discrete Models: A State-of-the-Art View, T.Oren, C. Shub, P. Roth (eds.).

[VV] U. V. Vazirani and V. V. Vazirani, "Trapdoor Pseudo-random Number Generators with Applications to Protocol Design," Proc. 24th Ann. Symp. on the Theory of Computing, Nov. 1983, 23-30.

[Ya1] A. C. Yao, "Some Complexity Questions related to Distributive computing," Proc. 11th Ann. ACM Symp on the Theory of Computing, April 1979, 209-213.

[Ya2] A. C. Yao, "Lower Bounds by Probabilistic Arguments," Proc. 24 Ann. Symp on the Theory of Computing, Nov. 1983, 420-428.