Functional Inversion and Communication Complexity

Shang-Hua Teng*

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

Abstract

In this paper, we study the relation between the multi-party communication complexity over various communication topologies and the complexity of inverting functions and/or permutations. In particular, we show that if a function has a *ring-protocol* or a *tree-protocol* of communication complexity bounded by H, then there is a circuit of size $O(2^H n)$ which computes an inverse of the function. Consequently, we have proved, although inverting NC^0 Boolean circuits is NP–complete, planar NC^1 Boolean circuits can be inverted in NC, and hence in polynomial time. In general, NC^k planar boolean circuits can be inverted in $O(n^{log^{(k-1)}n})$ time. Also from the ring-protocol results, we derive an $\Omega(n \log n)$ lower bound on the VLSI area to layout any one-way functions. Our results on inverting boolean circuits can be extended to invert algebraic circuits over finite rings.

One significant aspect of our result is that it enables us to compare the communication power of two topologies. We have proved that on some topologies, no one-way function nor its inverse can be computed with bounded communication complexity.

1 Introduction

One of the most fundamental questions in cryptanalysis is to characterize the class of permutations (or functions) whose inverse can be computed in polynomial time or by a polynomial size circuit [1, 10, 12, 16]. Much research in theoretically cryptography has been centered around finding the weakest possible cryptographic assumptions required in implementing major primitives [11, 4]. However, progress on characterizing permutations with small inversion circuits is very slow [12].

In this paper, we study the relation between multi-party communication complexity over various communication topologies and the complexity of inverting permutations and functions We show some nontrivial classes of permutations whose inverse can be computed efficiently.

In particular, we show that if a function has a *ring-protocol* or a *tree-protocol* of communication complexity bounded by H, then there is a circuit of size $O(2^H n)$ which computes an inverse of the function. Consequently, we have proved, although inverting NC^0 Boolean circuits is NP–complete, planar NC^1 Boolean circuits can be inverted in NC, and hence in polynomial time. In general,

*This work was supported in part by National Science Foundation grant DCR-8713489. Part of this work was done while the author was at School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

J. Feigenbaum (Ed.): Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 232-241, 1992.
© Springer-Verlag Berlin Heidelberg 1992

NC^k planar boolean circuits can be inverted in $O(n^{log^{(k-1)}n})$ time. Also from the ring-protocol results, we derive an $\Omega(n \log n)$ lower bound on the VLSI area to layout any one-way functions. Our results on inverting boolean circuits can be extended to invert algebraic circuits over finite rings.

One significant aspect of our result is that it enables us to compare the communication power of two topologies. We have proved that on some topologies, no one-way function nor its inverse can be computed with bounded communication complexity.

2 Definitions

Let $\mathcal{B} = \{0, 1\}$, i.e., the field $ZF(2)$. Let $f : \mathcal{B}^n \rightarrow \mathcal{B}^m$ be a *boolean mapping* from $\mathcal{B}^n$ to $\mathcal{B}^m$. The mapping f is a permutation if $m = n$ and f is a *bijection*. A function $g : \mathcal{B}^m \rightarrow \mathcal{B}^n$ is an *inverse of* f if for all $y \in \mathcal{B}^m$, $f(g(y)) = y$ whenever $g(y)$ is defined. In this paper, let $\mathcal{B}_{n,m}$ denote the set of all boolean mappings from $\mathcal{B}^n$ to $\mathcal{B}^m$.

A *Boolean circuit* is a directed acyclic graph whose nodes have indegree either 0 or 2. A node of indegree 0 is labeled with a variable, or with a boolean constant. A node with indegree 2 is labeled with a boolean function of 2 inputs. A node with a variable label is called an *input node* and the one with outdegree 0 is called an *output node*. We assume that each boolean circuit is *reduced* in the sense that no two input nodes share the same label. Each circuit C with n input nodes and m output nodes defines, in a natural way, a function, denoted by f_C, from $\mathcal{B}^n$ to $\mathcal{B}^m$.

A circuit C *computes* a function f if $f_C = f$. A circuit C' is an *inversion circuit* of f if C' computes an inverse of f.

The non-uniform and uniform versions of the *inversion problem* are defined as follows.

Definition 2.1 (Inversion Problem)

- **(Non-uniform)** *Is there a polynomial size circuit that computes an inverse of a given function* $f \in \mathcal{B}_{n,m}$*?*
- **(Uniform)** *Given a circuit or (a straight line program) that computes a given function* f*, construct an inversion circuit (with size bounded by a predefined function in* n *and* m*) of* f*.*

3 Communication Complexity

For each function $f \in \mathcal{B}_{n,m}$, we write $f : 2^X \rightarrow 2^Y$, where X denotes the set of n inputs and Y the set of outputs.

3.1 Two Party Communication Complexity

Suppose there are two processors in the system. For each partition (X_1, X_2) of X, processor 1 receives the values of variables in X_1 and processor 2 receives the values of variables in X_2. The *two party communication complexity* of f with respect to the partition (X_1, X_2), denoted by $C_f(X_1, X_2)$, is the number of bits the processors, using an optimal protocol, have to exchange, in the worst case, in order to jointly compute the values of all outputs of f [17, 13].

Notice that an optimal protocol for computing f with respect to the partition (X_1, X_2) also induces a natural partition of Y into (Y_1, Y_2) such that in the protocol, processor i is responsible to compute the values of all variables in Y_i, $(i \in \{1, 2\})$.

Let $\mathcal{P}(X)$ denote the set of all partitions of X and S_n the set of all permutations from $\{1, ..., n\}$ to $\{1, ..., n\}$. For each $\pi \in S_n$, let

$$\mathcal{P}_\pi(X) = \{(\{x_{\pi(1)}, ..., x_{\pi(k)}\}, \{x_{\pi(k+1)}, ..., x_{\pi(n)}\}) : 1 \leq k \leq n\}.$$

Definition 3.1 (Two Party Communication Complexity) *The* symmetric communication complexity *of a function $f \in \mathcal{B}_{n,m}$, denoted by $\mathcal{SC}(f)$, is defined to be*

$$\mathcal{SC}(f) = \max_{(X_1, X_2) \in \mathcal{P}(X)} C_f(X_1, X_2).$$

The permutational communication complexity *of a function $f \in \mathcal{B}_{n,m}$, denoted by $\mathcal{PC}(f)$, is defined to be*

$$\mathcal{PC}(f) = \min_{\pi \in S_n} \max_{(X_1, X_2) \in \mathcal{P}_\pi(X)} C_f(X_1, X_2).$$

It is easy to see that for all function f, $\mathcal{PC}(f) \leq \mathcal{SC}(f)$.

3.2 Multi-Party Communication Complexity

A *communication topology* of N nodes is a graph $G = (V, E)$ with $V = \{p_0, \ldots, p_{N-1}\}$ and $E \subset V \times V$, where V models the set of processors and G models the underlying communication network.

Each processor p_i is a Turing machine which has an input tape, an output tape and a work tape. Each edge of G models the communication channel between processors sited at its two ends. The whole system forms a computing device where each processor has some information, called *input* of the processor, and the processors want to jointly compute their respective share of outputs which are functions of all the inputs.

The computation is guided by a *distributed protocol* $\mathcal{P}$ which is a set of rules specifying the order and content of messages sent from one processor to another. We assume that all processors have unlimited computing power and the local computation is free. We only charge for the bits transmitted from one processor to the others. One major goal in the field of distributed computing is to design a distributed protocol to compute a given function that minimizes the maximum number of bits one processor has to receive, to send, or both.

For each topology G and protocol $\mathcal{P}$, let $\phi_G(\mathcal{P}, p_i)$ and $\psi_G(\mathcal{P}, p_i)$ denote, respectively, the maximum number of bits p_i has to receive and send in the worst case. Let $\omega_G(\mathcal{P}, p_i) = \phi_G(\mathcal{P}, p_i) + \psi_G(\mathcal{P}, p_i)$.

A distributed protocol $\mathcal{P}$ *computes* a function $f \in \mathcal{B}_{n,m}$ if there is an N-partition $(X_0, \ldots, X_{N-1})$ of X and $(Y_0, \ldots, Y_{N-1})$ of Y, where processor p_i receives an assignment x_i of X_i as its input, such that after running protocol $\mathcal{P}$, p_i computes y_i with $y = f(x)$, where $y = (y_0, \ldots, y_{N-1})$.

For each function, there is a trivial protocol with null communication cost, i.e., the one which assigns all inputs and all outputs to a single processor. In order to avoid this triviality, we only concern ourselves the set of *balanced protocols*,

Definition 3.2 (Balanced–Protocols) *A partition $(X_0, \ldots, X_{N-1})$ of X is H–balanced if for all $0 \leq i \leq N-1$, $|X_i| \leq H$. A protocol $\mathcal{P}$ for a function $f : 2^X \to 2^Y$ is H–balanced if its input–partition is H–balanced.*

Let $\mathcal{P}_{G,H}(f)$ denote the set of all H-balanced protocols that compute $f : 2^X \to 2^Y$,

Definition 3.3 (Balanced Communication Complexity) *For each function $f : 2^X \to 2^Y$, define*[1]

$$\Phi_{G,H}(f) = \min_{\mathcal{P} \in \mathcal{P}_{G,H}(f)} \max_i \phi_G(\mathcal{P}, p_i)$$

Notice that if the balanced communication complexity of a function f is small, then f can be computed by a circuit of small size (The proof of the following lemma will appear in the full paper).

Proposition 3.1 *For each graph G, for each $H \in \mathcal{R}^+$, each function $f : 2^X \to 2^Y$ can be computed by a circuit of size $O\left(2^{H+\Phi_{G,H}(f)}n\right)$.* □

The topology of communication networks play an important role in designing communication-efficient protocols. The set of communication topologies studied in this paper includes cliques, mesh, planar graphs, rings, and trees. The corresponding protocols are respectively called, *ideal protocols*, *mesh protocol*, *planar protocols*, *ring protocols*, and *tree protocols*.

4 Communication Topologies and Functional Inversion

In this section, we examine the communication power of various topologies including rings, meshes, trees, and cliques. We show that no one-way function (permutation) can be computed on a ring or a tree with bounded information exchange between neighbors.

4.1 Rings

We now prove that if a function f can be computed by an H-balanced ring-protocol with communication complexity $\Phi_H(f)$, then there is a circuit of size $O(2^{H+\Phi_H(f)})$ which computes an inverse of f.

Let $\mathcal{P}$ be an H-balanced ring-protocol which computes f with communication complexity $\Phi_H(f)$; let $(X_0, \ldots, X_{N-1})$ be the H-balanced partition induced by $\mathcal{P}$ on inputs and $(Y_0, \ldots, Y_{N-1})$ the partition on outputs; and let l_i and r_i be the number of bits the processor i sends to processor $i-1$ and $i+1$, respectively. For simplicity, all '+' and '-' (on index) in this section are modulo N. Let $h = H + \Phi_H(f)$. By definition, we have $l_i + r_i \leq h$.

Let $U_i = (u_{i,1}, ..., u_{i,l_i})$ and $V_i = (v_{i,1}, ..., v_{i,r_i})$ denote the set of variables whose values processor i sends to processor $i-1$ and $i+1$ respectively, in the protocol $\mathcal{P}$.

Notice that the protocol $\mathcal{P}$ defines a natural function f_i associated with processor i, from $(X_i \cup V_{i-1} \cup U_{i+1})$ to $(Y_i \cup U_i \cup V_i)$. Because f_i has only $O(h)$ bit inputs, f_i is computable by a circuit of size $O(2^h)$ (see Lemma 3.1). Let C_i be such a circuit computing f_i.

[1] We can also define:

$$\Psi_{G,H}(f) = \min_{\mathcal{P} \in \mathcal{P}_{G,H}(f)} \max_i \psi_G(\mathcal{P}, p_i)$$

$$\Omega_{G,H}(f) = \min_{\mathcal{P} \in \mathcal{P}_{G,H}(f)} \max_i \omega_G(\mathcal{P}, p_i).$$

We now define for each $y \in \mathcal{B}^m$ a digraph G_y with the property that G_y is not acyclic iff there exists $x \in \mathcal{B}^n$ such that $f(x) = y$.

For each output $y \in \mathcal{B}^m$, let M_i and T_i be $2^{l_i} \times 2^{r_{i-1}} \times 2^{l_{i+1}} \times 2^{r_i}$ matrices whose entries are defined as follows.

For each $i : 0 \leq i \leq t-1$, for each assignment $u_i, v_{i-1}, u_{i+1}, v_i$ to $U_i, V_{i-1}, U_{i+1}, V_i$.

- If there is an assignment x_i to X_i, such that $f_i(x_i, v_{i-1}, u_{i+1}) = (y_i, u_i, v_i)$, then $T_i[u_i, v_{i-1}, u_{i+1}, v_i] = X_i$ and $M_i[u_i, v_{i-1}, u_{i+1}, v_i] = 1$;
- If there is no such assignment, $M_i[u_i, v_{i-1}, u_{i+1}, v_i] = 0$;

We now define the digraph $G_y = (V, E)$ as

$$\begin{aligned} V &= \bigcup_{i=0}^{N-1} (\{i\} \times \mathcal{B}^{l_i} \times \mathcal{B}^{r_{i-1}}) \\ E &= \{((i, u_i, v_{i-1}), (i+1, u_{i+1}, v_i)) | M_i[u_i, v_{i-1}, u_{i+1}, v_i] = 1\} \end{aligned}$$

Lemma 4.1 *G_y is not acyclic iff there exists $x \in \mathcal{B}^n$ such that $f(x) = y$.*

Proof: Suppose that there exists $x \in \mathcal{B}^n$ such that $f(x) = y$. Since f can be computed by an H-balanced ring-protocol with communication complexity $\Phi_H(f)$ as above, there are x_i, u_i, and v_i such that $(y_i, u_i, v_i) = f_i(x_i, v_{i-1}, u_{i+1})$, and hence, $((i, u_i, v_{i-1}), (i+1, u_{i+1}, v_i))$ is an edge in G_y. Thus $(0, u_0, v_{N-1}), (1, u_1, v_0), \ \ldots\ , (N-1, u_{N-1}, v_{N-2}), (0, u_0, v_{N-1})$ forms a cycle in G_y.

On the other hand, since each simple cycle in G_y contains exactly one node from

$$\{i\} \times \mathcal{B}^{l_i} \times \mathcal{B}^{r_{i-1}}.$$

Hence each simple cycle of G_y is of the form

$$(0, u_0, v_{N-1}), (1, u_1, v_0), \ \ldots\ , (N-1, u_{N-1}, v_{N-2}), (0, u_0, v_{N-1}).$$

By definition of G_y, there exists x_i, such that $(y_i, u_i, v_i) = f_i(x_i, v_{i-1}, u_{i+1})$, and therefore $f(x) = y$. □

We now show how to invert f, given C_i.

Inputs: $y \in \mathcal{B}^m$.

- compute the matrices M_i and T_i for $0 \leq i \leq N$ using C_i;
- construct the digraph G_y, using M_i's;
- if G_y has no cycle, then output that there is no x such that $f(x) = y$;
- otherwise, compute a cycle of G_y $(0, u_0, v_{N-1}), \ldots, (N-1, u_{N-1}, v_{N-2}), (0, u_0, v_{N-1})$, and output $x = (x_0, \ldots, x_{N-1})$, where $x_i = T_i[u_i, v_{i-1}, u_{i+1}, v_i]$.

It is easy to check that the above algorithm runs in time $O(2^h n)$. Let C be a circuit that simulates the above algorithm, we have,

Theorem 4.1 *For each $f \in \mathcal{B}_{n,m}$, if there is an H-balanced ring-protocol computing f with communication complexity $\Phi_H(f)$, then there is a circuit of size $O(2^{H+\Phi_H(f)} n)$ which computes an inverse of f.* □

We say a function $f \in \mathcal{B}_{n,m}$ is *h-ring-partitionable* if it can be computed by an H-ring protocol such that $H + \Phi_H(f) \leq h$.

Corollary 4.1 *If f is $O(\log n)$-ring-partitionable, then there is a polynomial size circuit computing an inverse of f. Hence, there is no one-way function which is $O(\log n)$-ring-partitionable.* □

4.2 Trees

We now show that if a function f can be computed by an H-balanced tree-protocol with communication complexity $\Phi_H(f)$, then there is a circuit of size $O(2^{H+\Phi_H(f)}n)$ which computes an inverse of f.

Without loss of generality, we can assume that trees are rooted. But in the tree-protocol, each node can communicate with any of its neighbor (children and parent). For simplicity of the presentation, we further assume that the trees are binary, i.e., each node has at most two children. Our results can be extended to any bounded degree tree.

Let $\mathcal{P}$ be an H-balanced tree-protocol computing f with communication complexity $\Phi_H(f)$; let $(X_0, \ldots, X_{N-1})$ be the H-balanced partition induced by $\mathcal{P}$ on inputs and $(Y_0, \ldots, Y_{N-1})$ be the partition on outputs. For each node i in a given tree, let $p(i)$, $lc(i)$, $rc(i)$, be its parent, left child, and right child, respectively. Let $c(i)$ be a child of i. Let p_i, l_i and r_i be the number of bits the processor i sends to processors $p(i)$, $lc(i)$, and $rc(i)$, respectively. Let $h = H + \Phi_H(f)$. By definitions, we have $p_i + l_i + r_i \leq h$.

We now reduce the inversion problems to the following *consistency problem* on trees.

A *labeled tree* is a 3-tuple (T, F, Z) where T is a rooted tree with N nodes $\{0, .., N-1\}$, $Z = \{Z_0, \ldots, Z_{N-1}\}$, and $G = \{g_0, \ldots, g_{i-1}\}$. Each node i in T is associated with a set Z_i of k_i boolean variables and a $(k_i + k_{p(i)} + k_{lc(i)} + k_{rc(i)})$-place boolean function g_i which only depends on variables with node i and with neighbors of node i. An assignment to variables in Z *satisfies* g_i, if the value of g_i is 1 under this assignment.

Definition 4.1 (Consistency Problem on Trees) *Given a labeled tree (T, G, Z), compute an assignment of Z which satisfies all functions g_i, $0 \leq i \leq N-1$.*

The consistency problem on trees can be solved in $O(2^{\max_i(k_i)}n)$ time by RAKE operation [9, 8, 2]. Recently, the author gave an optimal $O(\log n)$ time algorithm for this problem when $(\max_i k_i)$ is a constant [14].

Let $U_i = (u_{i,1}, ..., u_{i,l_i})$, $V_i = (v_{i,1}, ..., v_{i,r_i})$, and $W_i = (w_{i,1}, ..., w_{i,r_i})$ denote the set of variables of whose values processor i sends to processors $p(i)$, $lc(i)$, and $rc(i)$, respectively, in the protocol $\mathcal{P}$. Notice that the protocol $\mathcal{P}$ defines a natural function f_i associated with processor i, from $X_i \cup U_{p(lc(i))} \cup U_{p(rc(i))} \cup W_{c(p(i))}$ to $Y_i \cup U_i \cup V_i \cup W_i$. Because f_i has only $O(h)$ bits inputs, f_i is computable by a circuit of size $O(2^h)$ (see Lemma 3.1). Let C_i be such a circuit computing f_i.

Now, let $Z_i = X_i \cup U_i \cup V_i \cup W_i$ and let g_i be a function from $(X_i \cup U_{p(lc(i))} \cup U_{p(rc(i))} \cup W_{c(p(i))} \cup U_i \cup V_i \cup W_i)$ to $\mathcal{B}$ such that g_i has value 1 if

$$(y_i \cup U_i \cup V_i \cup W_i) = f_i(X_i \cup U_{p(lc(i))} \cup U_{p(rc(i))} \cup W_{c(p(i))}).$$

¿From the above discussion, we have the following lemma (the proof will appear in the full paper).

Lemma 4.2 *for each $y \in \mathcal{B}^m$, for each assignment x to X, there is an assignment, u_i to U_i, v_i to V_i, and w_i to W_i satisfying all g_i iff $f(x) = y$.*

Therefore, for each $y \in \mathcal{B}^m$, in $O(2^h n)$ time, we can, using the algorithm for consistency problem on trees, compute an $x \in \mathcal{B}^n$, such that $f(x) = y$ (if such an x exists). Let C be a circuit that simulates the above algorithm, we have,

Theorem 4.2 *For each $f \in \mathcal{B}_{n,m}$, if there is an H-balanced tree-protocol computing f with communication complexity $\Phi_H(f)$, then there is an inversion circuit of size $O(2^{H+\Phi_H(f)}n)$ for f.* □

We say a function $f \in \mathcal{B}_{n,m}$ is *h-tree-partitionable* if it can be computed by an H-tree-protocol such that $H + \Phi_H(f) \leq h$.

Corollary 4.2 *If f is $O(\log n)$-tree-partitionable, then there is a polynomial size circuit computing an inverse of f. Hence, there is no one-way function which is $O(\log n)$-tree-partitionable.* □

4.3 Cliques

In the above two subsections, we show that if the balanced communication complexity (on trees or rings) of a function is small, then there exists a small size circuit computing an inverse of the function. Of course, the results depend critically on the topology of the communication networks. To what topology can our results be extended? We first observe that our result can be extended to $O(\log n)$ by n meshes (the proof will appear given in the full paper).

Theorem 4.3 *For each $f \in \mathcal{B}_{n,m}$, if there is an H-balanced mesh-protocol on the $O(\log n) \times n$ mesh computing f with communication complexity $\Phi_H(f)$, then there exists a circuit of size $O(2^{H+\Phi_H(f)}n^2)$ which computes an inverse of f.* □

It is remain open whether the similar result exists for an n by n mesh.

We now show, it is unlikely to extend the results to all topologies with bounded degree. We say a function $f \in \mathcal{B}_{n,m}$ is *k-partitionable* if there is a k-balanced protocol on an n-clique with communication complexity $\Phi_H(f)$ bounded above by k.

Theorem 4.4 *If[2] $NP \neq P$, there is a 6-partitionable function f such that no polynomial size circuit computes an inverse of f.* □

Proof: This theorem follows simply from the following lemma.

Lemma 4.3 (Garey and Johnson) *The SAT problem, in which each clause contains at most 3 variables or the negation of variables and each variable or its negation is in at most three clause, is NP-complete.* □

Corollary 4.3 *If $NP \neq P$, there is a 6-partitionable function f which is neither $O(\log n)$-ring partitionable, nor $O(\log n)$-tree partitionable.* □

Similarly, we have,

Corollary 4.4 *If $NP \neq T(2^{polylog})$, there is a 6-partitionable function f which is neither polylog-ring partitionable nor polylog-tree partitionable.* □

[2] In this paper, the notation of $NP \neq P$ denote that there is an NP function which can not be computed by a polynomial size circuit.

5 Inverting Planar Circuits

A boolean circuit C is *planar* if (1) the underlying graph of C is planar and (2) all inputs are on the same face of the underlying graph, (this face is called the *input face*).

We now show the relationship between the depth of a planar boolean circuit and the balanced communication complexity (on ring) of the function it computes.

Lemma 5.1 *If a function $f \in \mathcal{B}_{n,m}$ can be computed by a planar circuit of depth d, then f is $O(d)$-ring-partitionable.*

Proof: Without loss of generality, we can assume that C is embedded on the surface of a cylinder with the input face at the bottom if the cylinder (see Figure 1).

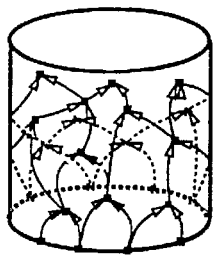

Figure 1: Embedding Planar Circuits on a Cylinder

Since the height of C is d, C has a $(1/3, 2/3)$-separator of size d [6, 7] that $(1/2, 2/3)$-splits the inputs (See Figure 2).

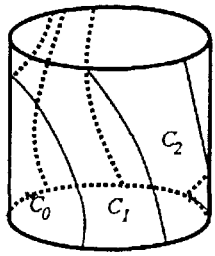

Figure 2: Partition of a Circuit by a h-separator

By recursively applying the separator partition, we can partition the circuit into n' components, $C_0, \ldots, C_{n'-1}$, each contains at most h-inputs. We say two components are neighbors, if they share some nodes which are removed during the partition. From the construction above, it follows that each component has at most two neighbors. Moreover, no pair of neighbors share more than d nodes which are removed, and hence, without loss of generality, we assume C_i has neighbor C_{i-1} and C_{i+1}. Therefore, we have an d-balanced ring-protocol with n' nodes, where the processor on node i evaluates the component C_i and communicates with processor on node $i-1$ and $i+1$ to

evaluate the nodes on the separator. Clearly, the communication complexity $\Phi_H(f)$ is bounded by $O(d)$. □

Theorem 5.1 *If f can be computed by a planar circuit of depth d, then an inverse of f can be computed by a circuit of size $O(2^d n)$.* □

A function $f \in \mathcal{B}^{n,m}$ is NC^k-computable if f can be computed by an $O((\log n)^k)$-depth circuit of polynomial size. It is NC^k-planar-computable if it can be computed by a $O((\log n)^k)$-depth planar circuit of polynomial size.

Corollary 5.1 *If $NP \neq P$, then there is an NC^0 function which can not be computed by an NC^1 planar circuit.* □

Combining with a result of Hastad [3],

Corollary 5.2 *if $P \neq NC^1$, then there is an NC^0 permutation which can not be computed by an NC^1 planar circuit.* □

Note that in the definition of the planar circuit, it is crucial to impose the restriction all inputs are on the same face. When this restriction is removed, the class of resulting circuits is called *general planar circuits*. The computational power of general planar circuit is greater than planar circuit (the proof will appear in the full paper).

Theorem 5.2 *If $NP \neq P$, there is a function f computable by a general planar circuit with constant depth and polynomial size, whose inverse is not computable by any polynomial size circuit.* □

6 Area Requirement of One-way Functions

In this section, we prove a lower bound on the VLSI area requirement of one-way functions in Thompson model [15, 5] where all inputs and outputs are on the boundary of the Thompson grid. We can prove (the proof will be in the final version)

Theorem 6.1 *If a function f has a circuit with layout area A, then there is a circuit of size $O(2^{A/n})$ computing an inverse of f.* □

Corollary 6.1 *For all one-way functions f, the area required to layout f is at least $\Omega(n \log n)$.* □

7 Final Remarks

All results presented in the above section are stated in the non-uniform form. Similar uniform version of the results can be proven. We also consider the parallel complexity of inverting boolean circuits. Those results will be included in the full paper. One interesting question remain open is to what topology can our upper bound result be extended. In particular, it is interesting to know whether similar results can be obtained on n by n meshes.

Finally, we have also obtained some results that relates the two-party communication complexity to the complexity of inverting permutations. Those results will be included in the final version of the paper.

Acknowledgments We would like to thank Alan Frieze, Merrick Furst, Hillel Gazit, Manpreet Khaira, Zhi-Li Zhang for helpful discussions.

References

[1] R. Boppana and J. Lagarias. One-way functions and circuit complexity. In *Proc. Struc. in Compl. Theory, Lect. Notes. in Computer Science.*, 1986.

[2] H. Gazit, G. L. Miller, and S.-H. Teng. Optimal tree contraction in the EREW Model, In *Current Computations edited by S. K. Tewsburg, B. W. Dickinson, and S. C. Schwartz"*, pages 139–156. 1988.

[3] J. Hastad. *Computational Limitations for Small Depth Circuits.* The MIT Press, 1986.

[4] R. Impagliazzo and S. Rudich. Limits on the provable consequence of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61. 1989.

[5] Frank Thomson Leighton. *Complexity Issues in VLSI.* Foundations of Computing. MIT Press, Cambridge, MA, 1983.

[6] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM J. of Appl. Math.*, 36:177–189, April 1979.

[7] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 376–382, 1984.

[8] G. L. Miller and J. H. Reif. Parallel tree contraction and its applications. In *26th Symposium on Foundations of Computer Science*, pages 478–489, 1985.

[9] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison Wesley, 1984.

[10] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digitial signatures and public–key cryptosystems. *CACM*, 21(2):120–126, 1978.

[11] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, pages 387–394. 1990.

[12] C. Sturtivant and Z.-L. Zhang. Efficiently inverting bijections given by straight line programs. In *31st Annual Symposium on Foundations of Computer Science*, pages 327–334. 1990.

[13] M. Szegedy. Functions with bounded symmetric communication complexity and circuit with mod m gates. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, pages 278–286. 1990.

[14] S. H. Teng. Fast parallel algorithms for tree-based constraint satisfaction problems. Manuscript, Carnegie Mellon University, 1990.

[15] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Carnegie-Mellon University, Department of Computer Science, 1980.

[16] A. C.-C. Yao. Theory and application of trapdoor functions. In *23th Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE, 1982.

[17] A. C.-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11st Annual ACM Symposium on Theory of Computing*, pages 209–213. ACM, 1979.