Depth-Bounded Communication Complexity for Distributed Computation

ABSTRACT

We study, in a VLSI design perspective, the communication complexity between two physically separated computational components. The optimization objective is to minimize amount of information exchange between these two parties. We argue that the notion of communication depths should be captured to reflect the system performance. In particular, we will focus on depth-1 communication, which has the most practical applications in VLSI design. We show that some existing techniques can be applied in the context of depth-1 communication for combinational problems. To extend to sequential cases, we show that dynamic cofactoring should be used to take advantage of state information for communication minimization. Theoretical analysis shows that the exact depth-1 solution is statistically very close to the lower bound for any-depth communication complexity.

1. INTRODUCTION

In VLSI design, long distance communication among local computational blocks should be avoided because long distance communication may introduce congestion in physical design, degradation in system performance, unreliability in noise analysis, etc. These problems are getting more and more serious partly due to the deep sub-micron (DSM) effects, and partly due to the growing complication of system design. In future DSM designs, it is predicted that a signal will take more than ten clock cycles to traverse the entire chip area [10]. Therefore, in addition to minimizing the amount of communication, it is necessary to bound the depth of communication through distant transmissions. Fortunately, due to the advances in semiconductor manufacturing, it becomes viable to trade the overhead of local computation for the reduction of distant communication.

Also it may often happen that there exist constraints on the locations of inputs and outputs. For instance, in system-on-chip design methodologies, inputs and outputs may be confined to some IP (intellectual property) blocks. As another example, in embedded system design, which may have inter-chip communication, the system can interact with an environment that spans a large area. Environmental parameters and the required interaction should be distributed locally. In these circumstances, the locations of inputs and outputs are fixed. In this paper we consider depth-bounded communication complexity where inputs and outputs are constrained to some location. In particular, we focus on the depth-1 case, which has the most practical applications on high-performance VLSI design.

The contributions of this paper include:

1. Formulate the communication complexity problem in the VLSI design perspective. In particular, we consider depth-bounded communication with fixed input-output constraints.

2. Relate the computation of depth-1 communication complexity with some existing techniques. Generalize the computation for sequential problems. By introducing dynamic cofactoring, the communication complexity can be minimized using state information.

3. Theoretically analyze the statistically relation between the exact depth-1 solution and the lower bound of any-depth communication complexity. Show that exact depth-1 solutions are very close to the optimum lower bound for a vast class of instances.

The remainder of this paper is organized as follows. We relate this paper with previous work in Section 2. After formulating the problem in Section 3, we discuss the depth-bounded communication complexity in Section 4. Section 5 includes our theoretical analysis relating the exact depth-1 solution with the general lower bound. We outline the experiments in Section 6, and will present the results in the final version of the paper. Conclusions are given in Section 7.

2. PRIOR WORK

Communication complexity has been intensively studied in theoretical computer science, e.g. see Yao's seminal paper [15]. A good survey on this topic can be found in [8]. Previous theoretical work has focused on proving lower bounds of communication complexity for combinational functions. Here we repeat the lower bound result of [11] as follows. Let $rank(M)$ denote the linear rank of matrix M over a finite field. (In this paper we are considered with a binary field.) Then

THEOREM 1 (RANK LOWER BOUND [11]). *Given a function f with a matrix representation M_f, then the communication complexity of M_f is at least* $\log_2 rank(M_f)$.

The result of Theorem 1 will be used in our later analysis on the optimality of depth-1 solution in Section 5. (The matrix representation of a function will be defined later.)

The most relevant formulation to ours is [12]. In [12], bounded round communication complexity was first discussed. The formulation is the same as our notion of depth-bounded concept except some minor technicalities. A factor-2 difference may arise in counting the amount of information

exchange. This is due to the fact that [12] assumed *prefix-freeness* to eliminate the "end of transmission" symbol. In the context of hardware design, this assumption is unnecessary since hardware itself has the nature of concurrency and transmission directions. Here we use *depth*, instead of *round*, to more adequately reflect hardware performance.

The previous work was only concerned with communication for combinational functions. However, most VLSI systems are inherently sequential. The communication complexity for sequential functions has to be addressed to reflect our need. Although the decomposition of finite-state machines has been studied [7], the optimization problems were centered in area or state minimization. Therefore the previous work is not directly applicable to our considered problem. Also as we analyze the problem from a hardware perspective, the notion of resource sharing, which was missing in the literature, remains to be captured.

The application of communication complexity in the VLSI-CAD field is not new. Hwang et al. exploited communication complexity for multilevel logic synthesis in [2, 3]. In their formulation there was no input-output constraint. The optimization problem became finding good input separations for a given (combinational) function. Also depths were not considered.

3. PROBLEM FORMULATION

We explore the solution to the problem formulated as follows. We are given a computation task $T(I, O)$ (either combinational or sequential) with sets I and O as inputs and outputs respectively. Suppose T should be executed by two parties A and B such that A owns inputs I_1 and outputs O_1 and B owns inputs I_2 and outputs O_2. Assume I_1 and I_2 form a partition of I, and O_1 and O_2 form a partition of O. (To prevent trivial solutions, assume I_1 and I_2 have similar sizes.) Because the communication between A and B has high cost and long delay (compared to local computation), it is required to finish T using the least amount of information exchange within a specified communication depth. (The depth counts the times of necessary back-and-forth communication between A and B.)

4. DEPTH-BOUNDED COMMUNICATION COMPLEXITY

We first study depth-bounded communication complexity for combinational instances, and then generalize the discussion for sequential cases.

4.1 Combinational Computation

Before analyzing depth-bounded communication complexity, we discuss Yao's model [15]. Let X, Y, Z be finite sets and $f : X \times Y \mapsto Z$ be the output function. Two parties, A and B, wish to evaluate $f(x, y)$, for some inputs $x \in X$ and $y \in Y$. However, A only knows x and B only knows y. Thus information needs to be exchanged. Suppose communication is the only concern. Then the cost of a communication protocol $\mathcal{P}$, which depends only on f, is the maximum number of bits exchanged for all possible inputs. To capture the essence of communication complexity, a communication protocol can be represented as a tree structure (a binary tree) reflecting the search for *monochromatic rectangles* [15] in the functional matrix of f, where rows are indexed by x and columns indexed by y. Each leaf in the protocol tree is a constant value in Z. When a monochromatic rectangle has been reached, A or B can determine the value of f. Figure 1 shows an example. The height (maximum depth to a leaf) of the protocol tree corresponds to the cost of $\mathcal{P}$.

We argue that the protocol tree does not well characterize the communication depth. This is because the functional dependencies in the tree might not follow a strict order of alternating communications. To obtain the communication depth of a protocol tree, one should rearrange the tree such that the functional dependency follows an alternating order starting from the root. The communication depth is then (1 + the number of alternating roles of the sender and the receiver). To avoid such a rearrangement, later we will modify the protocol tree directly taking depths into account.

4.1.1 Outputs on one side.

In the following analysis, we assume that only A provides the output for $f(x, y)$. To simplify the discussion, assume that $f(x, y)$ is a binary-output function. The generalization to multi-valued functions is straightforward. Analyzing the communication complexity of $f(x, y)$ is equivalent to analyzing $f'(x', y')$, a simplified function of f. This simplification procedure is provided in Observation 1. A function f can be represented by a matrix M: we use the local input variables (located on the same side as f), x, to index the rows of the functional matrix, and use other input variables, y, to index the columns.

OBSERVATION 1. *Given a functional matrix M of $f(x, y)$, let M' be a matrix derived from M by merging identical rows and columns. Then an optimum protocol $\mathcal{P}'$ for M' corresponds to an optimum protocol $\mathcal{P}$ for M with the same cost. Also permuting the order of rows (and/or columns) of the matrices is immaterial.*

Figure 2 shows the communication between A and B under some depth constraints. In the case of depth equal to 1, the communication complexity has lower bound $\lceil \log_2 k \rceil$, where k is the number of distinct column vectors in the functional matrix of $f(x, y)$, i.e. the number of columns in the reduced matrix from Observation 1. The protocol tree in this case can be represented as a k-ary tree with only one node, the root. Also we can view the functional matrix of f as having been vertically sliced into k sub-matrices, each has identical column vectors. This is similar to the decomposition chart used in functional decomposition [13]. In fact, the lower bound is always achievable. The corresponding protocol can be quickly derived using BDD-based functional decomposition [9]. In addition, the computation can be generalized to a set of functions $\vec{f}(x, y)$ by functionally decomposing their *hyperfunction* as in [5].

On the other hand, when the communication depth is relaxed to 2, it turns out that there is much more freedom in the protocol and the exact solution is computationally much harder. The solution space becomes exponentially much larger than the depth-1 case: In the first round of communication, although the depth-1 solution can be used as a communication upper bound, we still need to decide how many bits to transmit. Suppose k bits are decided. We need to decide how to partition the rows or columns of the functional matrix into 2^k portions. Once we have decided all of these, the second round communication is straightforward as the depth-1 case for each sliced sub-matrix. Thus searching an exact solution for depth-2 communication is in

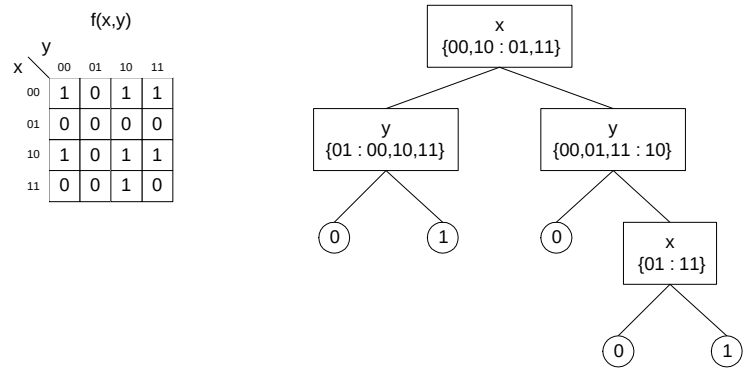

Figure 1: The functional matrix of $f(x, y)$ and a protocol tree for $\mathcal{P}$. The value of $\mathcal{P}$ on input (x, y) is the label of the leaf reached by starting from the root, and walking on the tree. Each internal node has two branches representing a bi-partition, denoted as "{left : right}", of the input space. The length of a path from the root to a leaf equals the number of bits exchanged under a particular instance of (x, y).

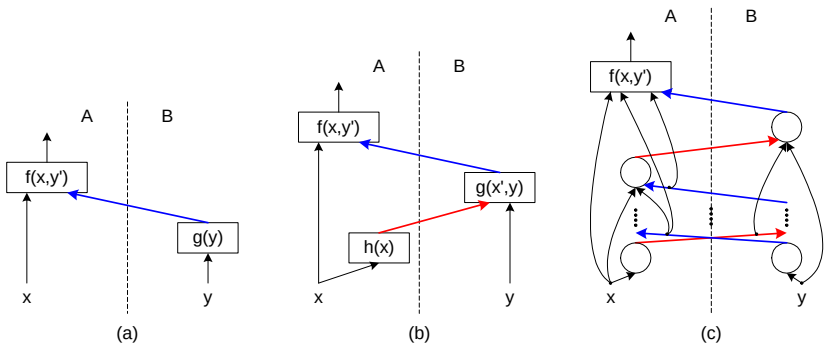

Figure 2: (a) Depth-1 communication. (b) Depth-2 communication. (c) General depth communication.

general intimidating.

The corresponding protocol tree in the depth-2 case can be represented as a 2-level (multi-branching) tree. The branches at the first level reflect the conditional information flowing from A to B and those at the second level reflect the information flowing from B back to A. Accordingly, the functional matrix of f is vertically sliced first into several sub-matrices, each of which are then horizontally sliced into finer sub-matrices. (As a result of Observation 1, after a slicing, each sub-matrix is free to have row and/or column permutation independent of other sub-matrices.) Similarly, one can generalize the relation between the protocol tree and the slicing of functional matrices. Figure 3 shows the relation between the slicing of a functional matrix and its corresponding protocol tree for a depth-3 communication scheme. In our modified protocol trees, in any two consecutive levels the roles of the sender and the receiver are switched. In addition, at the bottom level of the tree, B always sends information to A. Given a modified protocol tree, the required number of bits exchanged equals

$$\sum_i \log_2(\text{the max number of branches of a node at level } i).$$

As discussed earlier, for communication with depth ≥ 2 the exact solution appears to be formidable to compute. Heuristics need to be used.

4.1.2 Outputs on both sides.

We now consider a more general case, where A and B provide outputs $f_A(x, y)$ and $f_B(x, y)$ respectively. To analyze the communication complexity of computing $f_A(x, y)$ and $f_B(x, y)$, it is equivalent to analyze that of computing $f'_A(x', y')$ and $f'_B(x', y')$, which are simplified functions of f_A and f_B respectively. This simplification procedure is provided in Observation 2.

Observation 2. *Given two functional matrices M_a of $f_A(x, y)$ and M_b of $f_B(x, y)$, let $M = \begin{bmatrix} M_a & M_b^T \\ M_b^T & M_b^T \end{bmatrix}$ and its reduced matrix $M' = \begin{bmatrix} M'_a & M_b'^T \\ M_b'^T & M_b'^T \end{bmatrix}$, obtained by merging identical rows and columns. (The matrix transposes are due to the fact the entries of M_a are indexed by (x, y) and those of M_b by (y, x).). An optimum protocol $\mathcal{P}'$ for M'_a and M'_b corresponds to an optimum protocol $\mathcal{P}$ for M_a and M_b with the same cost.*

With the combined matrix M in Theorem 2, the matrix slicing can be performed as for the outputs-on-one-side case. Therefore we can combine two protocol trees into one. However, in computing the communication depth, we should analyze A and B separately because the bottom level of information flow may increase the depth for only one of them. Also simulated annealing algorithms may be applied in this case.

To see how mutual information can be shared between two parties, Figure 4 shows an example where a combi-

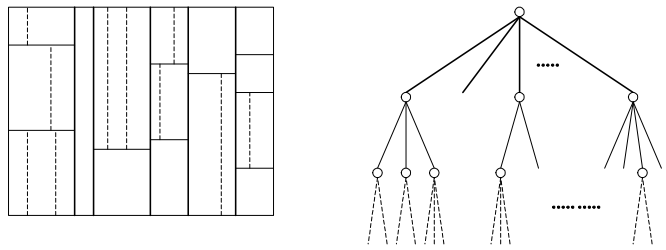

Figure 3: The slicing of a functional matrix and its corresponding protocol tree.

national cycle is necessary to achieve a minimum amount of communication. (Assume depth-2 communication is allowed.) It can be checked that breaking the combinational cycle increases the communication at least by one. Cyclic combinational circuits [6] are known to save area for certain circumstances. However, it was not known if this could result in a savings in communication. Here such a savings is demonstrated with a concrete example. Also the concept of information sharing was not addressed in previous studies of communication complexity. Although information sharing is useful from a design perspective, it does not affect the lower bound analysis for communication complexity.

4.2 Sequential Computation

We consider sequential instances, where the computation task $T(I, O)$ is history dependent. In particular, we assume that $T(I, O)$ can be modelled as a finite state machine (FSM).

Given a physical separation of inputs and outputs, the original FSM can be viewed as divided into two sub-FSMs, say M_A and M_B. Now the communication between M_A and M_B differs from the combinational case in that it should provide input information for correct next state transition in addition to primary output computation. Notice that because different outputs can induce different state equivalence relations, M_A and M_B can be reduced differently from the original FSM due to the separation of the set O of outputs.

4.2.1 Communication with full state information.

To minimize the amount of communication from M_A to M_B, M_A must have full information about the current state of M_B at every time instant. Using this state information, M_A can possibly condense the input information and only transmit what is necessary for M_B at a specific current state. A similar situation holds for communication from M_B to M_A. Hence a good strategy is to let M_A and M_B have the capability of simulating the state transition of the other. To do so, the simplest way is to let both M_A and M_B have *equivalent* state transition graphs as the original FSM, and let M_A and M_B keep in corresponding equivalent states at any time instant. As a result, each machine knows the other's current state by reading its own current state. Although this might seem wasteful from the computation point of view, it is worthwhile from the communication aspect.

Given M_A and M_B with equivalent transition functions $\delta_A : I_1 \times I_2' \times S \mapsto S$ and $\delta_B : I_1' \times I_2 \times S \mapsto S$ respectively, the computation of the exact solution for depth-one communication is quite similar to the combinational case. Assume M_A and M_B are in a current state $s \in S$. For s, the minimum amount of communication from M_A to M_B can be computed using functional decomposition over the hyperfunction of $f_B|_s$ and $\delta_B|_s$, where $f_B|_s$ and $\delta_B|_s$ are respectively the cofactored output and transition functions of M_B with respective to the current state s. A similar computation applies for the communication from M_B to M_A. Therefore the only difference with the combinational case is that f_B and δ_B change dynamically depending on the current state.

Since M_A and M_B have only subsets of the outputs, the output-induced state equivalence relations of M_A and M_B could be much simpler than that of the original FSM [4]. In such cases, the transition functions of M_A and M_B can be simplified separately. In general, this simplification may result in different transition functions, and thus reduce the resolution of state information that one knows about the other. The difference in transition functions introduces the notion of communication with partial state information.

4.2.2 Communication with partial state information.

Let S be the set of states of the original FSM. Suppose M_A and M_B have different transition functions $\delta_A : I_1 \times I_2' \times S_A \mapsto S_A$ and $\delta_B : I_1' \times I_2 \times S_B \mapsto S_B$ respectively. To simplify the discussion, we view $S_A = \cup_i E_{A_i}$ as having the same set of states as S, but with equivalence classes E_{A_i} of states induced by the outputs i of M_A. Similarly $S_B = \cup_j E_{B_j}$ has the same set of states as S, but has equivalence classes E_{B_j} of states induced by the outputs j of M_B. Then given a current state $s_a \in S_A$, the minimum amount of communication from M_A to M_B can be computed from functional decomposition over the hyperfunction of $f_B|_s$ and $\delta_B|_s$, $\forall s \in S^\dagger \subseteq S_B$, where

$$S^\dagger = \{s \in \bigcup_k E_{B_k} \mid E_{B_k} \cap S_A^* \neq \emptyset\},$$

$$S_A^* = \{E_{A_i} \mid E_{A_i} \cap E_B(s_a) \neq \emptyset\},$$

where $E_B(s_a)$ denotes the equivalence class of M_B containing the corresponding state of s_a.

Therefore, from the analysis of communication with full and partial state information, the sequential instance differs from combinational instance only in the cofactoring process. Given the physical separation constraints on inputs and outputs, and transition functions of the two separated sub-FSMs, the cofactoring computation gives the exact solution to the minimization of communication complexity.

5. ANALYSIS

In practical applications, depth-1 communication is of primary interest. It would be interesting to theoretically analyze the expected difference between the exact depth-1 solution and the communication lower bound. In the following

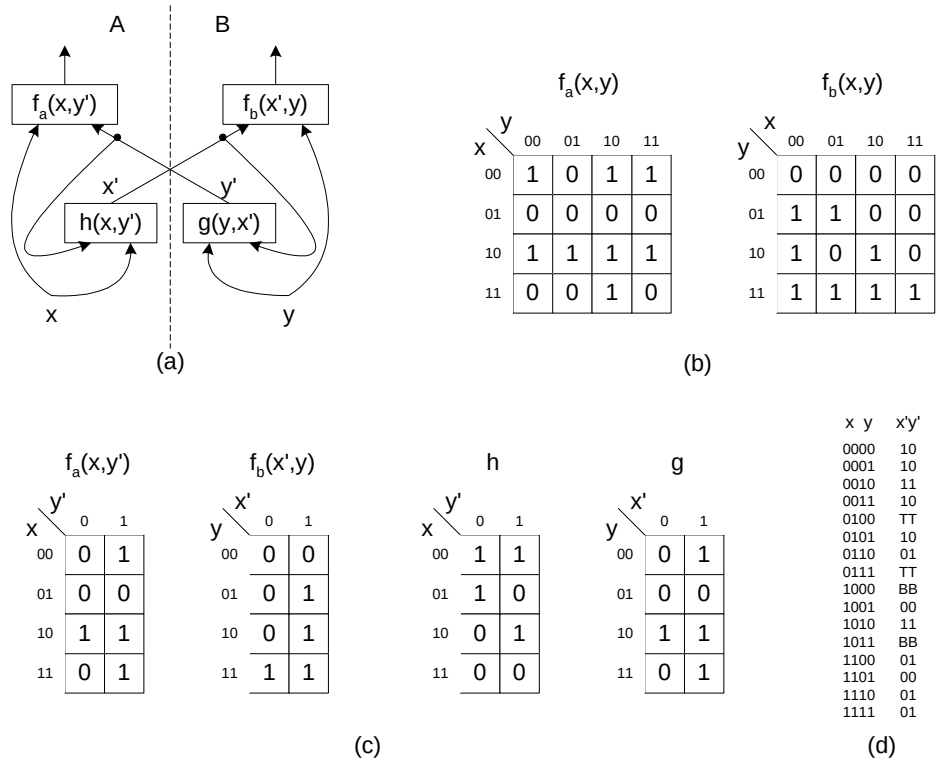

Figure 4: (a) The block diagram for functional dependencies. (b) The functions $f_a(x,y)$ and $f_b(x,y)$ under computation. (c) The functions for the components in (a). (d) The functions of x' and y' in terms of x and y. Here "T" and "B" denote the oscillation and bi-stable behaviors respectively. However, note that the outputs of f_a and f_b are independent of x', y' when such behaviors occur.

analysis, for the sake of simplicity we assume that the inputs of a given Boolean function f is partitioned into two equal-sized subsets. The analysis can be easily adjusted for general cases. With the bi-partitioning of inputs, f can be represented as a 0-1 matrix M.

THEOREM 2. *To color n balls with m colors ($m \geq n$), one out of the m colors is chosen uniformly at random when coloring each ball. The expected number of colors used is*

$$E_c(n,m) = \frac{1}{m^n} \sum_{k=1}^{n} k \cdot k! \cdot \binom{m}{k} \cdot S(n,k), \tag{1}$$

where $S(n,k)$ is a Stirling number of the second kind [14], *i.e.,*

$$S(n,k) = \frac{1}{k!} \sum_{i=0}^{k-1} (-1)^i \cdot \binom{k}{i} \cdot (k-i)^n \tag{2}$$

$$= \sum_{x_0=1}^{1} \sum_{x_1=1}^{k} \sum_{x_2=x_1}^{k} \cdots \sum_{x_{n-k}=x_{n-k-1}}^{k} \prod_{i=0}^{n-k} x_i \tag{3}$$

PROOF. (First proof) Let the n balls to be distinct, and thus there are m^n possible ways of coloring. We count the number of instances when k colors are used. By the fact that $S(n,k)$ counts the number of ways of partitioning a set of n elements into k nonempty subsets, there are $S(n,k)$ ways of grouping n balls into k subsets. Associating each of the k subsets with a distinct color, we have $\binom{m}{k} \cdot k!$ ways of doing so. Hence the probability of using k colors is

$$\frac{1}{m^n} \binom{m}{k} \cdot k! \cdot S(n,k).$$

The expectation value, $E_c(m,n)$, of Equation 1 follows. □

PROOF. (Second proof) Without the knowledge of the Stirling number of the second kind, we show another proof, which may also lead to the proof of Theorem 3.

We introduce $n-1$ binary variables, $\alpha_1, \cdots, \alpha_{n-1}$. Let us design the coloring process as follows. Start coloring from Ball 0 to Ball $(n-1)$. Initially let Ball 0 have a color. When coloring Ball i ($1 \leq i \leq n-1$), we consult α_i. If $\alpha_i = 0$, then we use a previously-used color. Otherwise, we use a not-previously-used color. Therefore $1 + \sum_{i=1}^{n-1} \alpha_i$ is the number of colors used. Also given a 0-1 vector of $(\alpha_1, \cdots, \alpha_{n-1})$, the corresponding coloring probability can be easily calculated:

$$\prod_{i=1}^{n-1} \frac{m^{\alpha_i} + (-1)^{\alpha_i} (\sum_{j=1}^{i} \alpha_j)}{m}$$

Observe that all the probabilities of vectors with $\sum_{i=1}^{n-1} \alpha_i = k-1$ (i.e. k colors used) have the common factor

$$\frac{1}{m^{n-1}} \frac{(m-1)!}{(m-k)!}.$$

Summing over these probabilities, we get the probability of

using k colors:

$$\frac{1}{m^{n-1}}\frac{(m-1)!}{(m-k)!}\cdot\sum_{x_0=1,1\le x_1\le\cdots\le x_{n-k}\le k} x_0x_1x_2\cdots x_{n-k}$$

Hence $E_c(n,m)$ is as Equation 1 with $S(n,k)$ represented as Equation 3. □

For an n-bit vector, there are 2^n possible valuations. By substituting 2^n for m in $E_c(n,m)$, it follows that

COROLLARY 1. *For an $(n\times n)$ matrix M with entries in the binary field $\mathcal{B}$ uniformly distributed over field elements $\{0,1\}$, the expected number of distinct columns is*

$$E_c(n,2^n)=\frac{1}{2^{n^2}}\sum_{k=1}^{n}k\cdot k!\cdot\binom{2^n}{k}\cdot S(n,k). \tag{4}$$

Let $Pc(n,k)$ denote the probability that the $(n\times n)$ matrix M has k distinct column vectors. $Pc(n,n)$ is the only dominating probability in $E_c(n,2^n)$. $Pc(n,n)$ approaches 1 extremely fast with respect to the increase of n. This is because the sample space of choosing column vectors is too large (2^n) to have much chance of repeating previously chosen column vectors. Hence $E_c(n,2^n)$ almost equals n. How about the rank lower bound for the optimal solution? Would it be close to n or much less than n? Below we manage to answer these questions.

It is well known that

PROPOSITION 1. *For any $(m\times n)$ matrix, its column rank equals its row rank.*

In the following discussion, we are concerned with the binary field.

LEMMA 1. *Given a binary matrix with rank r, the number of distinct rows (columns) is at most 2^r.*

PROOF. Given r linearly independent row vectors, we show that they can generate 2^r distinct row (column) vectors. Hence the lemma will follow.

Given two (distinct) non-zero row vectors $\vec{v}=(v_1,\ldots,v_n)$ and $\vec{u}=(u_1,\ldots,u_n)$, there are 4 possible linear combinations: $(\vec{0}\cdot\vec{v}+\vec{0}\cdot\vec{u})$, $(\vec{0}\cdot\vec{v}+\vec{1}\cdot\vec{u})$, $(\vec{1}\cdot\vec{v}+\vec{0}\cdot\vec{u})$, and $(\vec{1}\cdot\vec{v}+\vec{1}\cdot\vec{u})$, where $\vec{0}\equiv(0,0,\ldots,0)$, $\vec{1}\equiv(1,1,\ldots,1)$, "$\cdot$" denotes a positional-wise inner product, and "+" denotes a positional-wise XOR. It is clear that only the last one produces a new non-zero vector. Observe that $\vec{v}+\vec{u}$ produces a new row vector different from $\vec{v}$ and $\vec{u}$. Also use the fact positional-wise XORing any non-empty subset from a set of linearly independent row vectors generates a unique vector (different from others generated by other subsets). For a set of r linearly independent row vectors, $\binom{r}{0}+\binom{r}{1}+\binom{r}{2}+\cdots+\binom{r}{r}=2^r$ distinct row vectors can be generated.

Now consider the number of distinct column vectors. Let $\vec{w}=\vec{v}+\vec{u}$. When the three vectors $\vec{v}$, $\vec{u}$ and $\vec{w}$ are places in the form of a matrix, $\vec{w}$ cannot increase the number of distinct column vectors. On the other hand, consider adding a new row vector to a matrix. Suppose this row is linearly independent of all other original rows in the matrix. Then the number of distinct columns in the matrix is at most doubled by adding such a new row. Therefore given a binary matrix with rank r, the number of distinct columns is at most 2^r. □

THEOREM 3. *Given an $(n\times n)$ binary matrix M, which has each entry filled in as either 0 or 1 uniformly at random, the expected rank of M is*

$$E_r(n)=\frac{1}{2^{n^2}}\sum_{k=1}^{n}k\cdot\prod_{i=0}^{k-1}(2^n-2^i)\cdot T(n,k), \tag{5}$$

where

$$T(n,k)=\sum_{x_0=0}^{0}\sum_{x_1=0}^{k}\sum_{x_2=x_1}^{k}\cdots\sum_{x_{n-k}=x_{n-k-1}}^{k}2^{\sum_{i=0}^{n-k}x_i} \tag{6}$$

PROOF. Since the row rank is the same as the column rank by Proposition 1, without loss of generality, we consider the row rank of M. Let M be constructed from Row 0 to Row $(n-1)$. Each time a $(1\times n)$ row vector is added to M. (There are 2^n choices for such a row vector.) We apply a similar technique as used in the second proof of Theorem 2, and introduce n binary variables, $\alpha_0,\ldots,\alpha_{n-1}$. Variable α_i is used to indicate whether or not the rank is increased when appending Row i to M. Hence $\sum_{i=0}^{n-1}\alpha_i$ is the rank of M. From the proof of Lemma 1, we know that 2^r distinct row vectors can be generated from r linearly-independent row vectors. Therefore given a 0-1 valuation of $(\alpha_0,\cdots,\alpha_{n-1})$, its corresponding probability that M can be constructed this way is

$$\prod_{i=0}^{n-1}\frac{\left(2^n-2^{(\sum_{j=0}^{i}\alpha_j-1)}\right)^{\alpha_i}\cdot\left(2^{(\sum_{j=0}^{i}\alpha_j)}\right)^{1-\alpha_i}}{2^n}$$

Observe that, for all possible valuations of $(\alpha_0,\ldots,\alpha_{n-1})$ with $\sum_{i=0}^{n-1}\alpha_i=k$ (i.e. rank $=k$), there is a common factor,

$$\frac{1}{2^{n^2}}\prod_{i=0}^{k-1}(2^n-2^i).$$

Summing over these probabilities, we derive the probability that M has rank k:

$$\frac{1}{2^{n^2}}\prod_{i=0}^{k-1}(2^n-2^i)\cdot\sum_{x_0=0,0\le x_1\le\cdots\le x_{n-k}\le k}2^{x_0}\cdot 2^{x_1}\cdots 2^{x_{n-k}}$$

It follows that the expected rank $E_r(n)$ is as Equation 5. □

Let $Pr(n,k)$ denote the probability that the $(n\times n)$ matrix M has rank k. Figure 5 shows the probability distributions of $Pc(10,k)$ and $Pr(10,k)$, $k=0,1,\ldots,10$; Figure 6 plots the same probability distributions using a logarithmic scale. For sufficiently large n ($n>6$), there are three dominating probabilities, i.e. $Pr(n,n-2)\approx 0.13$, $Pr(n,n-1)\approx 0.58$ and $Pr(n,n)\approx 0.29$. Although the peak probability of $Pr(n,k)$ is not as sharp as that of $Pc(n,k)$, $E_r(n)$ is still very close to $E_c(n,2^n)$, more precisely $E_r(n)\approx E_c(n,2^n)-0.85\approx n-0.85$. This result indicates that, given a random $(n\times n)$ binary matrix, it would be almost full-ranked. Figure 7 shows the relation between $E_c(n)$ and $E_r(n)$ for $n=1,2,\ldots,20$.

Even though there exists an exponential gap [1] between depth-(k–1) and depth-k communication complexity, the expected number of distinct columns (rows) can be quite close to the expected rank. Therefore the result implies that the exact depth-1 solution is statistically quite close to global optimal. After taking the logarithm of the number of distinct column vectors, the amount of required information

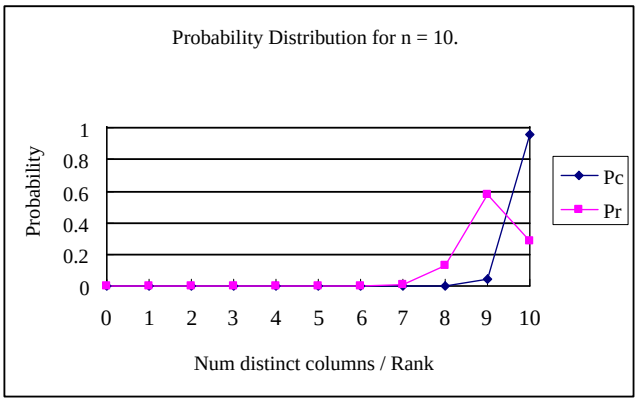

Figure 5: Probability distributions of $Pc(10, k)$ and $Pr(10, k)$.

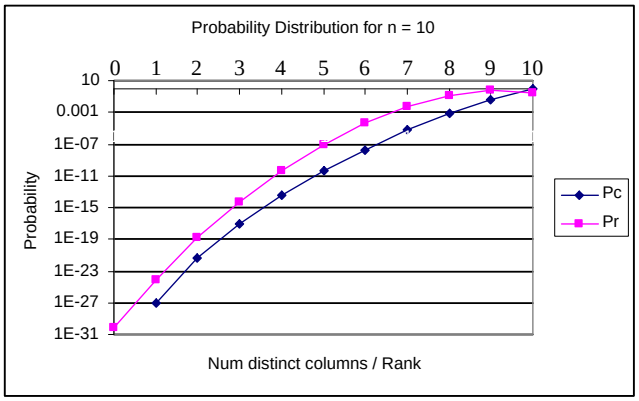

Figure 6: Probability distributions of $Pc(10, k)$ and $Pr(10, k)$ (logarithmic scale).

exchange for the depth-1 solution would be statistically almost the same as the global optimal solution. Notice that the rank provides a lower bound for the optimum solution; however, the optimum solution of a given instance could be much larger than the lower bound. In Example 1, we show an infinite family of examples whose communication complexity is exponentially larger than the rank lower bound. (To the best of our knowledge, it was not previously known if there exists an exponential gap between the rank lower bound and the optimum solution.)

EXAMPLE 1. *We construct a $(2^r \times 2^r)$ matrix X of rank r as follows.*

1. *Start from an $(r \times r)$ identity matrix I_r.*
2. *Build a $(2^r - 1 \times r)$ matrix X'' by adding $\binom{r}{2} + \binom{r}{3} + \cdots + \binom{r}{r}$ rows to I_r, where the $\binom{r}{i}$ rows are derived by adding (bitwise* XOR*ing) i row-vectors of I_r, $i = 2, \ldots, r$.*
3. *Build a $(2^r - 1 \times 2^r - 1)$ matrix X' by adding $\binom{r}{2} + \binom{r}{3} + \cdots + \binom{r}{r}$ columns to X'', where the $\binom{r}{i}$ columns are derived by adding (bitwise* XOR*ing) i column-vectors of X'', $i = 2, \ldots, r$.*
4. *Construct a $(2^r \times 2^r)$ matrix X from X' by adding a row and a column of all 1's.*

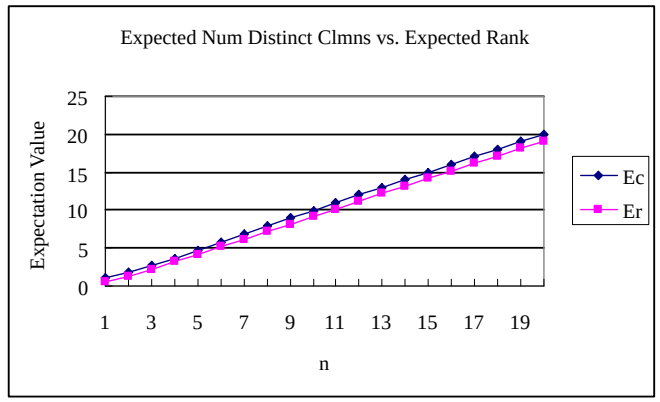

Figure 7: Distributions of $Ec(n)$ and $Er(n)$.

CLAIM 1. *The communication complexity of matrix X is of $\Theta(r)$.*

PROOF. Since the proof would be quite involved, we only give a sketch for the proof.

Matrix X has the property that any (3×3) rectangle in the matrix has rank greater than or equal to 2. For any communication protocol to reach a monochromatic rectangle, the maximal rectangle must have at least one dimension of size less than 3. It implies that at least $\Omega(\log 2^r) = \Omega(r)$ bits of information must be exchanged. On the other hand, 2^r bits of information suffice to determine the function represented by X. Hence the communication complexity of matrix X is of $\Theta(r)$. $\square$

COROLLARY 2. *The rank low bound can be exponentially loose away from the actual communication complexity.*

The above statistical analysis is based on the assumption that the considered binary matrix has entries either 0 or 1 purely at random. This might not be adequate for real VLSI designs. In addition to the design instance, the input-output separation constraints may have decisive effects on the minimum amount of information exchange. However, the statistical analysis may suggest that there exists a vast design class that the depth-1 solution is quite close to the global optimal.

6. EXPERIMENTS

We are currently in the middle of experimenting, and will answer the following questions in the final version of this paper.

1. In real-world circuit designs, would the rank and the number of distinct column vectors be far from the theoretic expectations?
2. Would BDD-based approach be practical in computing the exact depth-1 solution? (The BDD size would blowup if the number of distinct column vectors is too large.)
3. Even though depth-1 communication has the most practical application, it would be interesting to know how much information can be reduced by increasing the communication depth by one. The results might be useful in the tradeoff between speed and area. (Since

finding an exact solution for depth-2 problems is intimidating, heuristics must be applied here.)

7. CONCLUSIONS

In this paper we studied depth-bounded communication complexity, which has applications in the design of real-time embedded systems and VLSI design.

When we restrict the solution to depth-1 communication (which has the most practical application), exact solutions exist for both combinational and sequential computations. We show a procedure of dynamically calculating the necessary information exchange for sequential computation, such that at a particular current state combinational computation is used. Symbolic techniques may be applied in the computation.

Also we theoretically analyze the difference between the expected solution for depth-1 communication and the expected lower bound for any-depth communication. The result indicates that the exact depth-1 solution would statistically quite close to the lower bound, in spite of the existence of exponential gaps between the exact solutions of depth-$(k-1)$ and depth-k communications.

Although the experiments are not finished yet, we manage to show some results in the practical side of communication complexity problems. Thus we can compare them against our theoretical results.

8. REFERENCES

[1] P. Duris, Z. Galil, and G. Schnitger. Lower bounds on communication complexity. In *Proc. of the ACM Symposium on Theory of Computing*, pages 81–91, 1984.

[2] T.-T. Hwang, R. M. Owens, and M. J. Irwin. Exploiting communication complexity for multilevel logic synthesis. *IEEE Trans. on Computer-Aided Design*, pages 1017–1027, October 1990.

[3] T.-T. Hwang, R. M. Owens, and M. J. Irwin. Efficiently computing communication complexity for multilevel logic synthesis. *IEEE Trans. on Computer-Aided Design*, pages 545–554, May 1992.

[4] J.-H. Jiang and R. K. Brayton. On the verification of sequential equivalence. *Proc. of the Int'l Workshop on Logic and Synthesis*, June 2002.

[5] J.-H. Jiang, J.-Y. Jou, and J.-D. Huang. Unified functional decomposition via encoding for FPGA technology mapping. *IEEE Trans. on VLSI*, pages 251–260, April 2001.

[6] W. H. Kautz. The necessity of closed loops in minimal combinational circuits. pages 162–164, 1970.

[7] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, 1978.

[8] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[9] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *Proc. of the Design Automation Conf.*, pages 642–647, 1993.

[10] D. Matzke. Will physical scalability sabotage performance gains? *IEEE Computer Magazine*, pages 37–39, 1997.

[11] K. Mehlhorn and E. Schmidt. Las-Vegas is better than determinism in VLSI and distributed computing. In *Proc. of the ACM Symposium on Theory of Computing*, pages 330–337, 1982.

[12] C. Papadimitriou and M. Sipser. Communication complexity. In *Proc. of the ACM Symposium on Theory of Computing*, pages 196–200, 1982.

[13] J. P. Roth and R. M. Karp. Minimization over boolean graphs. *IBM Journal of Research and Development*, pages 227–238, 1962.

[14] R. P. Stanley. *Enumerative Combinatorics. I.* Cambridge University Press, 1997.

[15] A. Yao. Some complexity questions related to distributive computing. In *Proc. of the ACM Symposium on Theory of Computing*, pages 209–213, 1979.