BDD VARIABLE ORDERING FOR INTERACTING FINITE STATE MACHINES

by

Adnan Aziz, Serdar Tasiran, and Robert K. Brayton

Memorandum No. UCB/ERL M93/71

18 October 1993

BDD VARIABLE ORDERING FOR INTERACTING FINITE STATE MACHINES

by

Adnan Aziz, Serdar Tasiran, and Robert K. Brayton

Memorandum No. UCB/ERL M93/71

18 October 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

BDD VARIABLE ORDERING FOR INTERACTING FINITE STATE MACHINES

by

Adnan Aziz, Serdar Tasiran, and Robert K. Brayton

Memorandum No. UCB/ERL M93/71

18 October 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

BDD Variable Ordering for Interacting Finite State Machines

Adnan Aziz Serdar Tasiran Robert K. Brayton [*]
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, USA

Abstract

We address the problem of obtaining good variable orderings for the BDD representation of a system of interacting finite state machines (FSMs). Orderings are derived from the communication structure of the system. Communication complexity arguments are used to prove upper bounds on the size of the BDD for the transition relation of the product machine in terms of the communication graph, and optimal orderings are exhibited for a variety of regular systems. Based on the bounds we formulate algorithms for variable ordering. We perform reached state analysis on a number of standard verification benchmarks to test the effectiveness of our ordering strategy; experimental results demonstrate the efficacy of our approach.

[*]This research was supported by SRC 93-DC-008

1 Introduction

Verification of a design is typically done by modelling it as a finite state machine. Properties to be verified can be specified in a Temporal Logic[MP81], or by a Task Automaton[Kur]. Verification algorithms proceed by performing some form of traversal of the state transition graph[Dil, Bur]. Large designs arising in practice are invariably the product of small interacting finite state machines, depicted in Figure 1. Industrial experience indicates that the largest component machines rarely have more than a hundred states[York]. However forming the product machine leads to the state explosion problem[Gru, Chi]; given n Finite State Machines (FSMs) $\{M_1, M_2, \ldots, M_n\}$, the number of states in the product machine is the product of the number of states in each individual machine. As a result algorithms that explicitly operate on the state space of the product machine may have exponential time and space complexity.

A Binary Decision Diagram (BDD) [Bry] is a graph based data structure used for represent logic functions. It can be used to represent the transition relation of a binary encoded sequential machine implicitly by forming the corresponding characteristic function [Tou]. This representation can capture the regularity in the transition structure of the machine, and its canonicality makes it very useful in fixed point calculations. BDDs are now routinely used in formal verification[HTBSV, Bur]. The success of such algorithms depends critically on the size of the resulting BDD's which is very sensitive to the variable ordering chosen. Given a logic function, the problem of finding the ordering which leads to a minimum sized BDD for the function is co-NP complete[Bry].

In this paper we address the variable ordering problem for interacting finite state machines. Using communication complexity[Ull], upper bounds on the size of the BDD for a specified ordering are derived. Similar results have been shown for combinational circuits in [Ber, McMil] and our work was inspired by these. Indeed the transition relation for the product machine can be viewed as a logic circuit which takes the conjunction of the transition relations of the component machines, and the techniques of [McMil] yield upper bounds on the BDD size. The bound this method yields is weaker than the one we derive, and our proof is significantly different. We stress that we obtain orderings that minimize the representation of the transition relation of the product machine, and as such may not be good for representations of the reached state sets, or equivalent state sets. Indeed there are examples of systems where orderings exist such that the BDD for the transition relation is linear sized, whereas the BDD for the reached state set is exponential sized under any ordering. However our experimental results indicate the orderings we obtain work well in reached *state* computations. Furthermore, dynamic variable reordering can be used if the BDDs for the reached state sets become unwieldy.

Previous work in this area deals with ordering strategies for combinational [Mal] and sequential logic circuits [Tou, Jeo]. Touati [Tou] suggests deriving an ordering on the next state variables first using a heuristic based on minimizing the cumulative variable support of the latches. The inputs and present state variables are interleaved with the next state variables; their ordering is derived by standard DFS ordering on the next state logic [Mal]. Jeong [Jeo] gives efficient algorithms for finding BDD orderings based on the algebraic structure of the circuit. Another approach is based on dynamic ordering[Felt]. In this the BDD package automatically invokes a reordering routine which seeks to minimize the total number of BDD nodes by permuting small sets of adjacent variables. All these approaches are largely heuristic and do not yield a priori bounds.

In section 2 the basic notions of product machines and process communication graphs are defined. We prove upper bounds on the BDD size in terms of communication graph parameters. We characterize a large variety of interconnect structures for which asymptotically optimum orderings are derived. We also discuss interleaved orderings, and compare our approach with that of [Tou]. In section 3, we propose various algorithms based on these bounds for the variable ordering problem. In section 4 we present some results based on these algorithms. We conclude by discussing various extensions.

2 Theory

2.1 Definitions

In this section we define finite state machines and the semantics of their interaction. The definitions are motivated by the desire to model hardware designs. At the early stages of VLSI design, components may be incompletely specified, and the wires and states may not be encoded. Our definition allows this flexibility. Non-determinism is commonly used to abstract the environment or parts of the designs, and is reflected in the use of transition relations rather than functions to represent the state dynamics.

Definition 1 *A* **finite state machine** *with state space Q, inputs I, and output alphabet O is characterized by its transition relation $T \subset Q \times I \times Q$ and output relation $\Theta \subset Q \times O$, as illustrated in Figure 1.*

We allow machines to be non-deterministic and incompletely specified.

Systems are described as a collections of hardware units, communicating through a set of wires, and driven in lockstep by a single clock; this is the basis for the definition of product machine. Consider a system of n interacting machines $M_1, M_2 \ldots, M_n$. We assume there are no external inputs. Any external inputs I_{ext} can be modeled by adding a one state FSM such that this FSM non-deterministically outputs any of the inputs from I_{ext}. Each component machine M_i has present state s^{M_i}, next state variable t^{M_i}, and takes as input $\vec{o}_{ext,i}$ (some subset of outputs of the other machines).

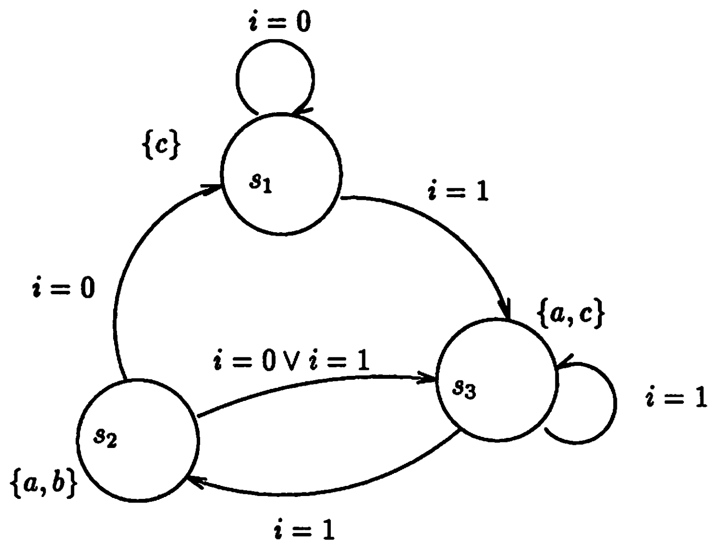


Figure 1: An FSM on states s_1, s_2, s_3, with inputs $0, 1$, and outputs a, b, c.

Basically, we are dealing with non-deterministic Moore machines with transition predicates on the edges.

Definition 2 *Given n machines $M_1, M_2, \ldots, M_n$, $M_i = (Q_i, T_i, \Theta_i)$, the* **product machine** $M = M_1 \otimes M_2 \otimes \ldots \otimes M_n$ *is the machine on state space $Q = Q_1 \times Q_2 \times \ldots \times Q_n$ and output space $O = O_1 \times O_2 \times \ldots \times O_n$, characterized by*

- *Output relation*

$$\Theta(x, o) = \prod_{i=1}^{n} \Theta_i(x_i, o_i)$$

- *Transition relation*

$$T(x, y) = (\exists \vec{i})[\prod_{k=1}^{n} T_k(x_k, i, y_k) \cdot \Theta_k(x_k, o_k) \cdot (i = [o_1 \ldots o_n])]$$

where the present state variable is $x = [x_1 x_2 \ldots x_n]$, the next state variable is $y = [y_1 y_2 \ldots y_n]$, and the output variable $o = [o_1 o_2 \ldots o_n]$.. We assume the sets O_i, O_j are disjoint for all i, j.

Binary Decision Diagrams may be used to represent the characteristic function of the transition relation, the output relation, and set of states encountered in verification.

The problem now addressed is: *Given machine M as previously defined, find a good variable ordering for T, using only the communication structure, i.e. without making use of the internal* *details of the component machines.* We start by restricting attention to the class of variable orderings C^M where variables corresponding to different machines are not interleaved, i.e. only orderings which are permutations of $\{<\vec{x}_1, \vec{y}_1>, \ldots, <\vec{x}_n, \vec{y}_n>\}$, allowing arbitrary permutations within the variables of a given machine. In section 2.3 we discuss reasons for this restriction and examine interleaved orderings.

As a first step towards solving this problem we use communication complexity to prove upper bounds on the size of the BDD corresponding to a permutation of the above form. The following definition is used to make the notion of bit communication complexity precise. To illustrate our approach we use a simplified notion of finite state machine, where the machine output is simply the state. Our results and algorithms are easily extended to machines with outputs that are functions of the state.

Definition 3 *Given machine M as above, a directed edge labeled graph G_M is defined as follows: There is a vertex v_j for each component machine M_i. If the transition relation for M_j depends on the state of M_i, we add edge $e_{i,j}$ labelled with $\vec{x}_i$. We refer to this graph as the* **process communication graph (PCG).**

2.2 Upper Bounds

We first develop some intuition behind the bounds derived in this section As an example let M, σ be as given in figure 2. $T(\vec{x}, \vec{y})$ is the product $\prod_{i=1}^{6} T_i$. The number of BDD nodes at level 4 is bounded as follows. Split T into the product of $(T_2T_6T_3T_1)$ and (T_5T_4). The number of cofactors of T with respect to the variables in $V = \{x_2, y_2, x_6, y_6, x_3, y_3, x_1, y_1\}$ is no more than the number of cofactors of $(T_2T_6T_3T_1)$ with respect to V times the number of cofactors of (T_5T_4) with respect to V.

(T_5T_4) has only a limited number of variables ($w_f^\sigma(4)$,defined as below) which are being cofactored in its support. There are at most $2^{w_f^\sigma(4)}$ possible for cofactors of (T_5T_4).

$(T_2T_6T_3T_1)$ has only a limited number of variables ($w_r^\sigma(4)$, defined as below) remaining after cofactoring. There are at most 2^{2^n} functions on n boolean variables, so there are at most $2^{2^{w_r^\sigma(4)}}$ possible cofactors of $(T_2T_6T_3T_1)$.

Hence the BDD for $T(\vec{x}, \vec{y})$ under the non-interleaved ordering derived from σ has no more than $2^{w_f^\sigma(4)} \cdot 2^{2^{w_r^\sigma(4)}}$ nodes at level 4. This approach can be extended to derive bounds at each level of the BDD and hence on the total size of the BDD.

Theorem 2.1 *Let M be a system of n interacting machines $M_1, M_2, \ldots, M_n$, and let σ be a permutation on $\{1, 2, \ldots, n\}$. Then the number of distinct cofactors of $T(\vec{x}, \vec{y})$ with respect to the variables in $\{\vec{x}_{\sigma(1)}, \vec{y}_{\sigma(1)}, \vec{x}_{\sigma(2)}, \vec{y}_{\sigma(2)}, \ldots, \vec{x}_{\sigma(k)}, \vec{y}_{\sigma(k)}\}$ is bounded by $2^{w_f^\sigma(k)} \cdot 2^{2^{w_r^\sigma(k)}}$, where*

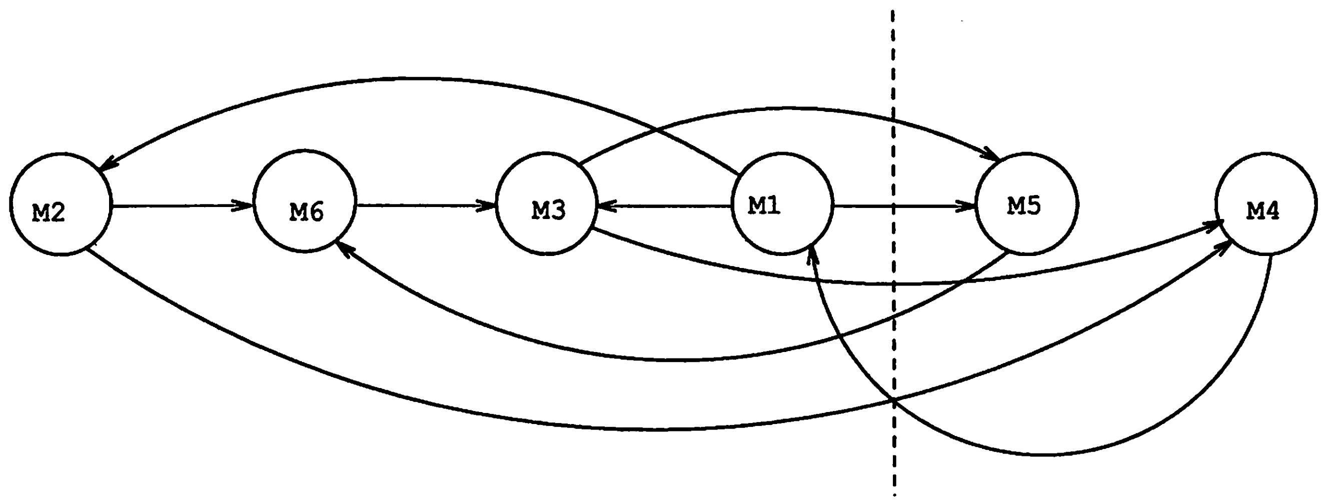


Figure 2: Machine M as in figure 1. The machines follow the permutation $\sigma = (413652)$. We are considering $k = 4$. $w_f^\sigma(4) = |x_1| + |x_3| + |x_2|$, $w_r^\sigma(4) = |x_5| + |x_4|$.

$$\begin{aligned} w_f^\sigma(k) &= \textit{number of distinct bits communicated from} \\ &\quad \{M_{\sigma(1)}, \ldots, M_{\sigma(k)}\} \textit{ to } \{M_{\sigma(k+1)}, \ldots, M_{\sigma(n)}\} \\ w_r^\sigma(k) &= \textit{number of distinct bits communicated from} \\ &\quad \{M_{\sigma(k+1)}, \ldots, M_{\sigma(n)}\} \textit{ to } \{M_{\sigma(1)}, \ldots, M_{\sigma(k)}\} \end{aligned}$$

Proof: Refer to Appendix A for a detailed proof. ∎

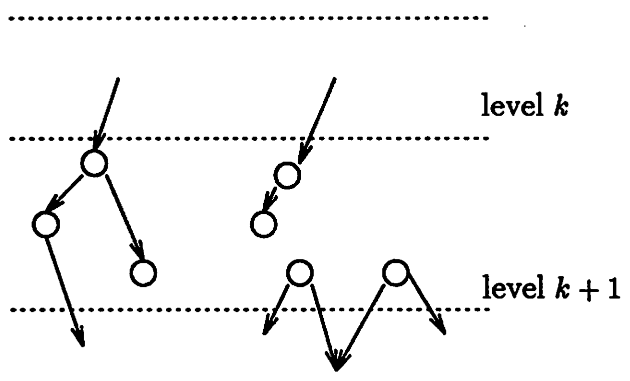
As a corollary to the above theorem we get an upper bound on the number of nodes for the BDD for $T(\vec{x}, \vec{y})$:

Corollary 2.1 *The number of nodes in the BDD for $T(\vec{x}, \vec{y})$ for the ordering $\vec{x}_{\sigma(1)} \prec \vec{y}_{\sigma(1)} \prec \ldots \prec \vec{x}_{\sigma(n)} \prec \vec{y}_{\sigma(n)}$, is bounded by :*

$$S^\sigma = \sum_{i=1}^{n} (2^{2(|x_{\sigma(i)}|)} \cdot 2^{w_f^\sigma(i+1)} \cdot 2^{2^{w_r^\sigma(i+1)}}) \tag{1}$$

A looser bound is

$$M^\sigma = n \cdot c \cdot 2^{w_f^\sigma} \cdot 2^{2^{w_r^\sigma}} \tag{2}$$



- PS variables at level $k+1$: $x_{\sigma(k)}$
- NS variables at level $k+1$: $y_{\sigma(k)}$
- At most $2^{w_f^\sigma(k)} \cdot 2^{2^{w_r^\sigma(k)}}$ roots at level $k+1$
- For each root at most $2^{|x_{\sigma(k)}|}$ extra nodes

Figure 3: Bounding the total number of nodes in the BDD for $T(\vec{x}, \vec{y})$

where

- *w_f^σ is the maximum number of forward crossing bits across any partition induced by σ*
- *w_r^σ is the maximum number of reverse crossing bits across any partition induced by σ*
- *c depends only on the maximum number of state bits in some machine*

Proof:

From Theorem 2.1, it follows that the number of nodes in the BDD for $T(\vec{x}, \vec{y})$ at level $k+1$, where the variables from the FSMs $M_{\sigma(1)}, \ldots, M_{\sigma(k)}$ have been cofactored out, is bounded by $2^{w_f^\sigma(k)} \cdot 2^{2^{w_r^\sigma(k)}}$. The total number of nodes at all the levels between level k and level $k+1$ is no more than $2^{2(|x_{\sigma(i)}|)}$ times the number at level k. Summation over k yields the result. This is sketched out in figure 3

■

Remark: Given k, there are always choices of the individual machines such that there are at least $2^{w_f^\sigma(k)}$ distinct cofactors of $T(\vec{x}, \vec{y})$ with respect to $\{\vec{x}_{\sigma(1)}, \vec{y}_{\sigma(1)}, \vec{x}_{\sigma(2)}, \vec{y}_{\sigma(2)}, \ldots, \vec{x}_{\sigma(k)}, \vec{y}_{\sigma(k)}\}$, ie the bound of Theorem 2.1 is tight in the first term. Details are available in [TR].

2.3 Interleaved Orderings

Typically, communication within a machine is dense, ie each bit of the next state depends on all the present state bits, since otherwise the machine would have a trivial factorization. Reasoning as in Corollary 2.1, it would seem that interleaving the variables leads to increased communication complexity and higher bounds. This suggests that variables corresponding to a single machine be ordered contiguously.

However, there are situations where interleaving state variables from different machines may be superior to a non-interleaved ordering. Consider a product machine in which a component machine has a transition relation $T = [y\bar{\oplus}(x_1^A x_1^B + x_2^A x_2^B + \ldots + x_n^A x_n^B)]$, where x_i^A, x_i^B are the present state bits from machine M^A and M^B. In this case, a non-interleaved ordering will result in an exponential sized BDD for T, where as the ordering $< x_1^A, x_1^B, \ldots, x_n^A x_n^B >$, is optimum and yields a linear sized BDD for T.

Interleaved orderings are also superior in the context of equivalent state computations. If states are equivalent only to themselves, the equivalence predicate is equality. For the equality relation a noninterleaved ordering is exponential, and an interleaved ordering is linear.

2.3.1 Comparison with Touati's heuristic

Touati et. al.[Tou] consider a BDD based approach to the problem of the equivalence of sequential hardware consisting of latches and logic gates. Their heuristic for variable ordering proceeds by first finding a permutation σ^* on the latches which minimizes the support, i.e. minimizes the following cost function:

$$\text{cost}(\sigma) = \sum_{1 \leq j \leq n} \mid \bigcup_{1 \leq i \leq j} \text{supp}(f_{\sigma_j}) \mid$$

where $\mid A \mid$ denotes the cardinality of A, and $\text{supp}(f_i)$ is the set of variables in the support of f_i. Malik's heuristic [Mal] is used to order the supports of the f_i, $\text{supp}(f_i)$ individually. Finally input and output variables are interleaved as follows: $\text{supp}(f_{\sigma(1)}), y_{\sigma(1)}, \ldots, \text{supp}(f_{\sigma(n)} - \bigcup_{1 \leq i \leq n-1} \text{supp}(f_{\sigma(i)})), y_{\sigma(n)}$.

Naturally, this ordering procedure can be used to derive orderings for systems of interacting FSMs. The component machines correspond to latches, and their fanins correspond to the support. $\text{cost}(\sigma)$ can be calculated directly from the process communication graph. However $\text{cost}(\sigma)$ is not directly correlated to the communication complexity of σ. Consider the system of FSMs interacting through a binary tree as shown in Figure 3. The ordering $<< M_1, M_2, \ldots, M_{2^n-1} >>$ is optimum for $\text{cost}(\sigma)$ but has high communication complexity as is seen in the partition $U = \{1, 2, \ldots, 2^{n-1} - 1\}$, $V = \{2^{n-1}, \ldots, 2^n - 1\}$, for which $w_f = 2^{n-1}$. In fact, there exists an ordering with low communication complexity, namely the ordering returned by lexicographically first DFS, for which $w_f = 2$.

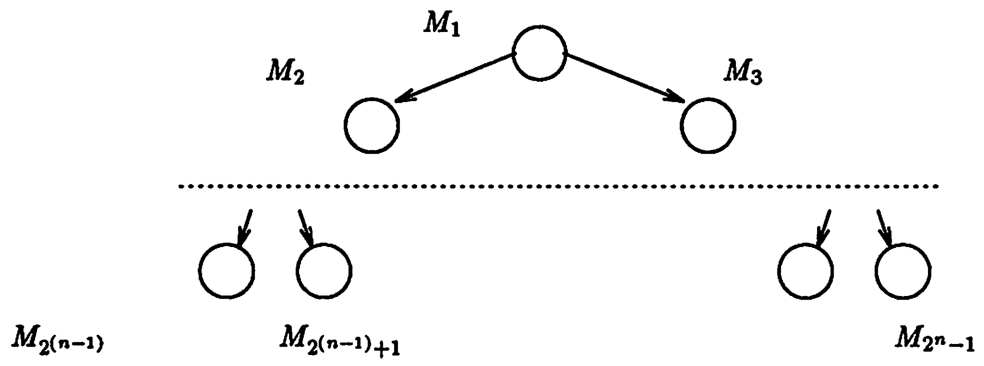


Figure 4: An example on which Touati's Heuristic fails

2.4 Optimum Orderings

We can use the bounds derived above to get optimum variable orderings for a variety of sparse interconnect structures. The following lemma shows that for certain recursive structures whose communication complexity is independent of the number of nodes in the graph, the transition relation for the product machines grows linearly in n.

Lemma 2.1 *Let $M(n)$ be the composition of n component machines $\{M_1, \ldots, M_n\}$. Suppose there exist constants w_f^*, w_r^*, and b such that for for all n,*

1. *there exists an ordering σ_n^* of the vertices of the PCG such that for each k*

$$\begin{aligned} w_f^{\sigma_n^*}(k) &\leq w_f^* \\ w_r^{\sigma_n^*}(k) &\leq w_r^* \end{aligned}$$

2. *the state of each M_k^n can be encoded in not more than b bits*

Then for all n there is exists a variable ordering such that the BDD for $T(n) = T_1 \times T_2 \ldots \times T_n$ has at most $c \cdot n$ nodes, where c is independent of n.

Proof: We use the bound of corollary 2.1, (equation 2)

$$M^\sigma = n \cdot 2^{2\max_i(|x(i)|)} \cdot \max_k(2^{w_f^\sigma(k)} \cdot 2^{2^{w_r^\sigma(k)}})$$

For each machine M_n using an ordering defined by σ_n^* leads to a BDD for T_n such that

$$| T_n | \leq n \cdot 2^{2\cdot\max_k|x_k|} \cdot \max_k(2_f^{w^{\sigma_n^*}(k)} \cdot 2^{2^{w_r^{\sigma_n^*}(k)}})$$

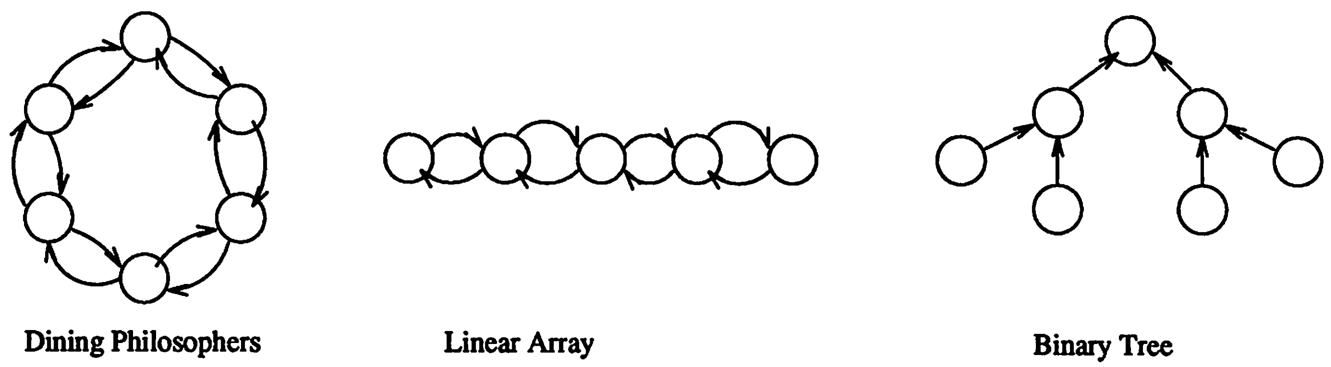


Figure 5: A variety of sparse interconnect structures commonly encountered in verification

By the hypothesis of the lemma, $\max_k \mid x_k \mid \leq b$, $\max_k w_f^{\sigma_n^*}(k) \leq w_f^*$, and $\max_k w_r^{\sigma_n^*}(k) \leq w_r^*$. Therefore

$$\mid T_n \mid \leq \; n \cdot 2^{2 \cdot b} \cdot 2^{w_f^*} \cdot 2^{2^{w_r^*}} \leq c \cdot n$$

where $c = 2^{2 \cdot b} \cdot 2^{w_f^*} \cdot 2^{2^{w_r^*}}$

■

Armed with this lemma, we can find variable orderings for a variety of interconnect structures (parametrized by n) that yield linear sized BDDs, and so are optimum within constants. Figure 4 illustrates some communication graphs which satisfy the hypothesis of the lemma.

3 Algorithms

The upper bounds S^σ and M^σ of section 2 can be used to obtain a good variable ordering for the transition relation of a given product machine M composed of $\{M_1, \ldots M_n\}$. We first extract the process communication graph (PCG). The parameters in the bounds can be calculated directly from the PCG. We then find a permutation σ^* on the vertices of the graph that minimizes the bound. A non-interleaved variable ordering corresponding to σ is simply one in which all variables corresponding to the machine $M_{\sigma^*(1)}$ appear first (in any internal order amongst themselves), followed by all variables from $M_{\sigma^*(2)}$, etc.

Finding an optimum permutation on the vertices of the communication graph is a hard problem, akin to the Travelling Salesman Problem. We conjecture that it is NP-complete. Exhaustive search has factorial complexity, and dynamic programming yields an algorithm with complexity $O(n^3 2^n)$. We now discuss an exact branch and bound procedure and some heuristics for finding good permutations.

3.1 Branch and Bound

Consider the problem of obtaining a permutation σ^* which minimizes $S^\sigma = \sum_{i=1}^{n} g_i(\sigma)$. This problem lends itself to a branch and bound algorithm solution. This can be better understood by a closer look at the sum for S^σ. We have to minimize S^σ over all permutations. Let $S^{\hat{\sigma}}$ be the smallest sum obtained from the permutations considered so far. For the next permutation σ' being examined, if for some k $\sum_{i=1}^{k} g_i(\sigma') \geq S^{\hat{\sigma}}$, then the summation can be terminated. Moreover, any permutation that has $< \sigma'(1) \ldots \sigma'(k) >$ as a prefix can be eliminated. This provides a way to prune the search tree. In fact any permutation that has the FSMs above in that order can be pruned. But it is expensive to store and check for this information. The same formulation yields a branch and bound algorithm to find a permutation minimizing M^σ.

To increase the probability of pruning, we first calculate a good initial guess. This is explained in more detail in the next subsection. Then, we go over all permutations lexicographically, pruning wherever possible. The exhaustive search tree is never constructed, since this would require excessive memory and time. Instead, when a "bad" (in the sense described above) prefix is discovered, the algorithm branches to the next permutation in lexicographic order which does not have the bad prefix.

The input to the branch and bound algorithm is a "process communication graph". Observe that all parameters in the bound can be calculated directly from the PCG. For each permutation examined, for each k, the number of bits crossing the cut in the forward and reverse directions are counted separately. Then the cost for the cuts so far for that permutation is calculated. Either pruning takes place before all cuts are examined, or the permutation considered is the best seen so far. The algorithm proceeds in this manner until all permutations are exhausted.

The memory used by this algorithm is linear in the size of the process communication graph. The algorithm can take factorial time in the worst case. It was necessary to experiment with the algorithm to see how effective the pruning is at reducing the search.

3.2 Heuristics

Several schemes, with varying degrees of sophistication, exist for picking the initial guess for the optimum permutation. The simplest is to choose vertices one at a time in the greedy sense, ie, choose the next vertex to minimize the partial cost. This greedy approach can be extended to an algorithm that has bounded look-ahead k. The algorithm proceeds recursively by computing all possible choices for the first k vertices in the permutation. It chooses the best among these and recursively completes the ordering. While improving the quality of the initial guess, look-aheads of k increase the time complexity of the algorithm by adding a factor of n^k. This is acceptable compared with the complexity of the branch and bound algorithm, since a good starting point can prune the search space drastically. When n is large, the initial guesses themselves can be used as approximate solutions. We implemented a simple greedy algorithm to minimize S^σ, and algorithms with look-aheads of 2 to minimize S^σ and M^σ.

4 Results

4.1 Computing an Optimum Permutation

The branch and bound algorithms that find permutations that minimize S^σ and M^σ were implemented using a greedily generated initial guess. Still for more than 10 to 15 vertices in the communication graph, the exact branch and bound algorithm was too slow. Our experiments show that a permutation which minimizes S^σ generated using a look-ahead of 2 yields a solution that is sufficiently close to being optimum and has negligible running time. When the communication graph has a regular structure (e.g. a mesh, tree, ring, etc), this permutation is often optimum. Thus, this is the method of choice.

4.2 Correlation between bound and actual size of BDD for Transition Relation

For assessing the usefulness of our bound as a measure of the actual BDD size, and checking the validity of assumptions, we needed a set of representative examples to test our algorithms. We artificially constructed several product machines according to various interconnection schemes. In each case the component machines had a small state space and a randomly chosen (possible nondeterministic) transition relation.

Descriptions of the process communication graphs of the test examples are given in Table 1. The benchmarks were written in the S/R language [Har]. We extracted the process communication graphs, and found variable orderings as follows:

ran_vars All variables are ordered randomly

ran_comps Non-interleaved ordering where component machines are ordered randomly

touati_heur Interleaved ordering as described in section 2.3.1

min_comm Non-interleaved order where components are ordered to minimize communication complexity as described in section 2.1

As seen in Table 2, in all cases, our approach significantly reduces the BDD size as compared with random orderings. All algorithms took only a few seconds to compute the ordering. The running time was negligible when compared to the time taken to read in the example and build the

EXAMPLE	Communication Structure
Acyclic	Acyclic graph on 25 FSMs
Cyclic	Graph on 25 FSMs with more than 10 cycles, no specific direction to the data flow
Few_Ran_C	Random graph with few (< 10) cycles on 25 FSMs
RT	Tree on 25 FSMs with communication in both directions between parent and child
Mesh20	Rectangular grid on 25 FSMs as in Figure 1
Tree31	Complete binary tree on 31 FSMs
Ran_Mesh	16 randomly connected FSMs, with outdegrees from 2 to 5
AcyclicII	Similar to Acyclic, Larger component machines
CyclicII	Similar to Cyclic, Larger component machines

Table 1: Communication structure of our benchmarks

EXAMPLE	*ran_vars*	*ran_comps*	*touati_heur*	*min_comm*
Acyclic	∞	inf	5,336	1,188
Cyclic	∞	inf	21,174	3,185
Few_Ran_C	∞	inf	35,444	1,085
RT	∞	inf	2,455	421
Mesh20	∞	inf	21,573	9,203
Tree31	∞	inf	75,034	1,134
Ran_Mesh	∞	43,727	17,435	6,511
AcyclicII	∞	inf	112,379	10,756
CyclicII	∞	inf	583,603	79,199
∞: Intermediate BDD grew larger than 1,000,000 nodes				

Table 2: Number of BDD nodes for the TR of the product machine; in all cases time to find the ordering was negligible

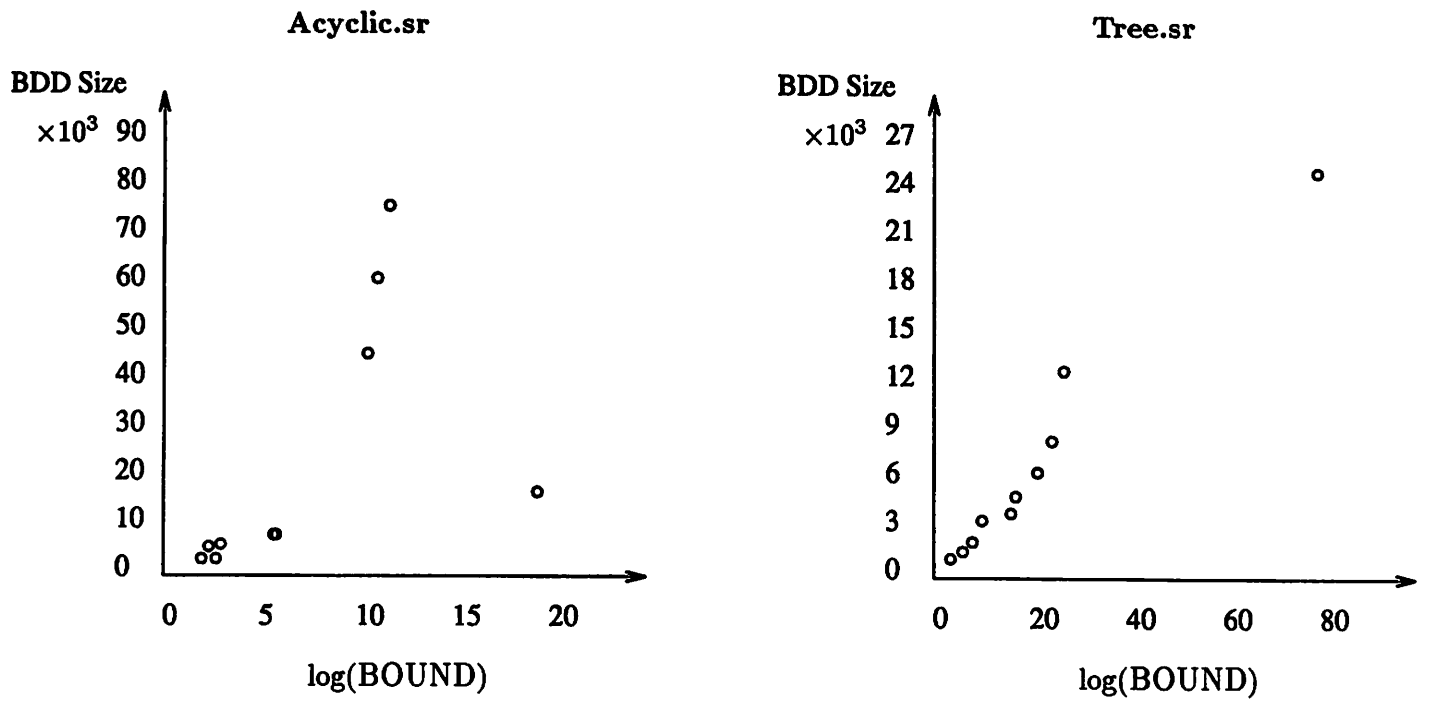


Figure 6: Correlation between bound and actual BDD Size

BDD. Note that *Mesh20*, which has high communication complexity (as shown in [TR]) had a large BDD compared to *Tree31* which has low communication complexity (as discussed in section 2.4). Also observe that Touati's heuristic does particularly badly on *Tree31* and *RT*, corroborating the discussion of section 2.3.1. We also experimented with some interleaving, similar to that given by Touati's heuristic. In many cases this allowed further reduction in BDD size. We concluded that ordering based on ***minimizing the communication complexity, while allowing some heuristic interleaving works best in practice.***

The bound S^σ derived in Corollary 2.1 is an upper bound. To examine the relationship S^σ between the actual number of nodes in the BDD for the transition relation under the non-interleaved ordering derived from σ, we took several different permutations of the components and computed the bound. We then built the BDD and plotted the actual BDD size against the bound. The plots are seen in figure 5; they demonstrate a strong correlation between the bound and the actual BDD size.

4.3 Performance on Reachability Analysis

As mentioned in the introduction, an ordering which minimizes the size of the BDD representing the transition relation does not necessarily lead to an ordering which is good for general verification

EXAMPLE	Description
DinPhil32	Dining Philosopher protocol, 32 processes arranged in a ring
MilSched16	Milner Job Shop Scheduler, 6 jobbers and hammers in an augmented ring
Fis5	Fischer Mutual Exclusion protocol, 5 processes, dense complex interaction
DME15	Distributed mutual exclusion circuit, 15 elements, tree structured
Dynachek3	Railway controller, 3 train tracks, Random sparse interconnect
2MDLC	Part of a Data Link Controller Random interconnect

Table 3: Verification benchmarks used for reached state computation

calculations. Reached state computation is a core routine in verification. A truer measure of the effectiveness of our ordering is given by our performance on reached state analysis for realistic verification benchmarks.

Table 3 lists a series of common verification benchmarks with a short description of their communication structure. In Table 4 we describe results for reached state analysis using ordering derived from *min_comm* on these examples. We computed the reached state set in the most direct fashion, namely by directly computing the image of the reached state set under the global transition relation. There are more sophisticated ways of computing the image that involve using dont cares to minimize the BDDs, and partitioned transition relations to avoid building the full transition relation. However this simple experiment is enough to give us a good idea of the performance of our ordering.

Specifically we report the following three statistics—

TR size : Number of nodes in BDD for transition relation

Max-Reached : Number of nodes in largest BDD representing a set of states encountered during reached state computation

Time-Reached : Time taken to perform reached state computation

The ordering given by *touati_heur* was identical to that given by *min_comm* on all examples except for *DME15* where it was orders of magnitude worse. This is explained by the fact that *DinPhil32, MilSched16, Fis4* have ring like structure, and *touati_heur* will find the asymptotically best ordering for rings. However, as pointed out in section 2.3.1, *touati_heur* is especially bad for

EXAMPLE	min_comm		
	TR size	Max-reached	Time-reached (sec)
Dynachek3	332	37	2.0
DME16	4,982	45	1.2
DinPhil32	4,863	304	26.0
MilSched16	5,003	938	33.6
Fis5	22,227	27,901	1145.4
2MDLC	24,487	1,335	635

Table 4: Results on reached state analysis

trees, and this is brought across in *DME15*. Also, the best ordering that could be obtained by hand for *Fis4* was much worse than that generated by *min_comm*.

5 Conclusion

We addressed the problem of deriving good variable orderings for the BDD representation of a system of interacting finite state machines for formal verification applications. Towards this end we introduced the notion of the process communication graph and proved results connecting BDD size to the communication graph. We justified the decision to derive orderings based only on knowledge of the communication graph. We use the bounds to formulate fast heuristic algorithms for variable ordering. The experimental results show good correlation between the BDD size predicted by the bound and the actual BDD size, and comparing our results on the transition relation size with orderings derived from Touati's heuristic validates our decision to use orderings which minimize the communication complexity through the communication graph. Our performance on reached state analysis for verification benchmarks further demonstrates the effectiveness of our approach.

We have implemented our algorithms in a hierarchical synthesis and verification tool currently under development. We are applying our techniques to the *quantification ordering* problem which comes up in reached state computation. We have also developed methods to partition logic designs using a cost function which minimizes bit communication complexity. For these applications, even the greedy approach to finding a good ordering is too slow, and we are working on faster algorithms.

References

[Bry] R. E. Bryant. "Graph-based algorithms for Boolean Function Manipulation". *IEEE Transactions on Computers*, August 1986.

[Tou] H.J.Touati, H.Savoj,B.Lin,R.K.Brayton, and A.Sangiovanni-Vincentelli. "Implicit state enumeration of finite state machines using BDD's", in *Proceedings of the International Conference on Computer-Aided Design*, pp. 130-133 Nov. 1990.

[Chi] M.Chiodo, T. R. Shiple, A. Sangiovanni-Vincentelli, R. K. Brayton,"Automatic Compositional Minimization in CTL Model Checking", in *Proceedings of the International Conference on Computer-Aided Design*, pp. 172-178 Nov. 1992.

[McMil] K. L. McMillan, "Symbolic Model Checking", Ph.D. thesis, Carnegie Mellon University 1992.

[Felt] E. Felt, G, York, R. Brayton, and A. Sangiovanni-Vincentelli, "Dynamic Variable Reordering for BDD Minimization", submitted to *EURO-DAC*

[Ull] J. D. Ullman, "Computational Aspects of VLSI", Computer Science Press 1984.

[Har] Z. Har El and R. P. Kurshan, "Software for Analytic Development of Communication Protocols", *AT&T Technical Journal*, pp. 45-49, Jan 1990.

[Mal] S.Malik, A.Wang, R.K.Brayton, and A.Sangiovanni-Vincen telli. "Logic Verification using Binary Decision Diagrams", In *ICCAD 1988*

[York] G. York, Personal Communication 1993

[MP81] Z. Manna, A, Pneuli, "Verification of concurrent programs: The temporal framework", The correctness problem in Computer Science, R. Boyer, J. Moore, Eds, Academic Press, London 1981, 215-273

[TR] *Technical Report*

[Ber] C.L. Berman, "Ordered Binary Decision Diagrams and Circuit Structure", In *ICCD 1989*

[HTBSV] R. Hojati, H. Touati, R. Brayton, and A. Sangiovanni-Vincentelli, "Efficient Omega-Regular Language Containment", In *CAV 1992*

[Jeo] S.W. Jeong, B. Plessier, G.D. Hachtel, and F. Somenzi, "Variable Ordering for FSM Traversal", in *ICCAD 1991*

[Kur] R. Kurshan, "Anlysis of Discrete Event Coordination", Lecture Notes in Computer Science 1990

[Bur] J.R. Burch, E.M. CLarke, K.L. McMillan, and D.L. Dill, "Symbolica Model Checking: $10^2$0 state and beyond", In *LICS 1990*

[Gru] O. Grumberg and D.E. Long, "Model Checking and Modular Verifcation", In *CONCUR 1991*

[Dil] D.L. Dill, A.J. Hu, and H, Wong-Toi, " Checking for Language Inclusion Using Simulation Preorder", In *CAV 1991*

A Proof of Upper Bound on BDD size

Theorem A.1 *Let M be a system of n interacting machines $M_1, M_2, \ldots, M_n$, and let σ be a permutation on $\{1, 2, \ldots, n\}$. Then the number of distinct cofactors of $T(\vec{x}, \vec{y})$ with respect to $<< \vec{x}_{\sigma(1)}, \vec{y}_{\sigma(1)}, \vec{x}_{\sigma(2)}, \vec{y}_{\sigma(2)}, \ldots, \vec{x}_{\sigma(k)}, \vec{y}_{\sigma(k)} >>$ is bounded by $2^{w_f^\sigma(k)} \cdot 2^{2^{w_r^\sigma(k)}}$, where*

$$\begin{aligned} w_f^\sigma(k) &= \textit{number of distinct bits communicated from} \\ &\quad \{M_{\sigma(1)}, \ldots, M_{\sigma(k)}\} \textit{ to } \{M_{\sigma(k+1)}, \ldots, M_{\sigma(n)}\} \\ w_r^\sigma(k) &= \textit{number of distinct bits communicated from} \\ &\quad \{M_{\sigma(k+1)}, \ldots, M_{\sigma(n)}\} \textit{ to } \{M_{\sigma(1)}, \ldots, M_{\sigma(k)}\} \end{aligned}$$

Proof:

Observe

$$\begin{aligned} T(\vec{x}, \vec{y}) &= \prod_{i=1}^{n} T_i(\vec{x}_i, \vec{y}_i) && (3) \\ &= \left(\prod_{i=1}^{k} T_i(\vec{x}_{\sigma(i)}, \vec{y}_{\sigma(i)})\right) \cdot \left(\prod_{i=k+1}^{n} T_i(\vec{x}_{\sigma(i)}, \vec{y}_{\sigma(i)})\right) && (4) \end{aligned}$$

Let

$$\begin{aligned} A_\sigma^k &= \prod_{i=1}^{k} T_i(\vec{x}_{\sigma(i)}, \vec{y}_{\sigma(i)}) \\ B_\sigma^k &= \prod_{i=k+1}^{n} T_i(\vec{x}_{\sigma(i)}, \vec{y}_{\sigma(i)}) \end{aligned}$$

Observe A_σ^k involves only the state variables internal to $M_{\sigma(1)}, \ldots, M_{\sigma(k)}$ plus $w_r^\sigma(k)$ variables internal to the remaining $N - k$ FSMs. Similarly B_σ^k involves only the state variables internal to $M_{\sigma(k+1)}, \ldots, M_{\sigma(n)}$ plus $w_f^\sigma(k)$ variables internal to the remaining k FSMs.

Since cofactoring distributes with respect to conjunction we have the following identity:

$$\begin{aligned} & T(\vec{x}, \vec{y})_{[(\vec{x})_{\sigma(1)}=\vec{b}_1, (\vec{y})_{\sigma(1)}=\vec{b}'_1, \ldots, (\vec{x})_{\sigma(k)}=\vec{b}_k, (\vec{y})_{\sigma(k)}=\vec{b}'_k]} = \\ & \quad (A_\sigma^k)_{[(\vec{x})_{\sigma(1)}=\vec{b}_1, (\vec{y})_{\sigma(1)}=\vec{b}'_1, \ldots, (\vec{x})_{\sigma(k)}=\vec{b}_k, (\vec{y})_{\sigma(k)}=\vec{b}'_k]} \cdot (B_\sigma^k)_{[(\vec{x})_{\sigma(1)}=\vec{b}_1, (\vec{y})_{\sigma(1)}=\vec{b}'_1, \ldots, (\vec{x})_{\sigma(k)}=\vec{b}_k, (\vec{y})_{\sigma(k)}=\vec{b}'_k]} \qquad (5) \end{aligned}$$

Notation: $F_{t_1, \ldots, t_l}$ denotes the set of distinct cofactors of F with respect to $t_1, \ldots, t_l$.

Then by equation 1,

$$|\,T(\vec{x},\vec{y})_{\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)},\vec{y}_{\sigma(k)}\}}\,| \le |\,(A^k_\sigma)_{\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)}\}}\,| \cdot |\,(B^k_\sigma)_{\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)}\}}\,| \tag{6}$$

Since B^k_σ involves only $w^\sigma_f(k)$ variables from $\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)},\vec{y}_{\sigma(k)}\}$ there are at most $2^{w^\sigma_f(k)}$ possible cofactors of B^k_σ with respect to $\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)}\}$. Hence

$$|\,(B^k_\sigma)_{\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{(x)}_{\sigma(k)}\}}\,| \le 2^{w^\sigma_f(k)} \tag{7}$$

Since A^k_σ involves only $w^\sigma_r(k)$ variables not from $\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)},\vec{y}_{\sigma(k)}\}$, and there are no more than $2^{2^{w^\sigma_r(k)}}$ Boolean functions on $w^\sigma_r(k)$ variables,

$$|\,(A^k_\sigma)_{\{\vec{x}_{\sigma(1)},\vec{y}_{\sigma(1)},\ldots,\vec{x}_{\sigma(k)},\vec{y}_{\sigma(k)}\}}\,| \le 2^{2^{w^\sigma_r(k)}} \tag{8}$$

The theorem follows from equations 6, 7 and 8. ■

B Lower Bounds

As another application of communication complexity arguments, we prove a lower bound on the BDD size for a specific function.

Consider the function $f_n : B^{n \times n} \to B$,where n is even, given by

$$f_n(x_{11},\ldots,x_{nn}) = \prod_{i=1}^{n}\prod_{j=1}^{n}(x_{ij} + x_{(i+1)\text{mod}nj} + x_{i(j+1)\text{mod}n} + x_{(i-1)\text{mod}nj} + x_{i(j-1)\text{mod}n}) \tag{9}$$

Each expression $(x_{ij} + x_{(i+1)\text{mod}nj} + x_{i(j+1)\text{mod}n} + x_{(i-1)\text{mod}nj} + x_{i(j-1)\text{mod}n})$ will be referred to as the product term p_{ij}

Graphically f_n is derived from a symmetric mesh M^* on $n \times n$ vertices as shown in Figure 1. We will refer to the vertices by the associated variables x_{ij}.

We know from graph theory that the bisection width of an $n \times n$ mesh M is n, ie at least n edges must be removed from the mesh to disconnect any two disjoint sets of $n^2/2$ vertices. Since M^* contains M, the bisection width of M^* is also at least n.

Consider any ordering σ of the variables. $\sigma : \{1,\ldots,n\} \times \{1,\ldots,n\} \to \{1,\ldots,n^2\}$ is a bijection. Consider the bipartition of the vertex set X into

$$X_1 = \{x_{ij} \mid \sigma(i,j) \le n^2/2\}$$
$$X_2 = \{x_{ij} \mid \sigma(i,j) > n^2/2\}$$

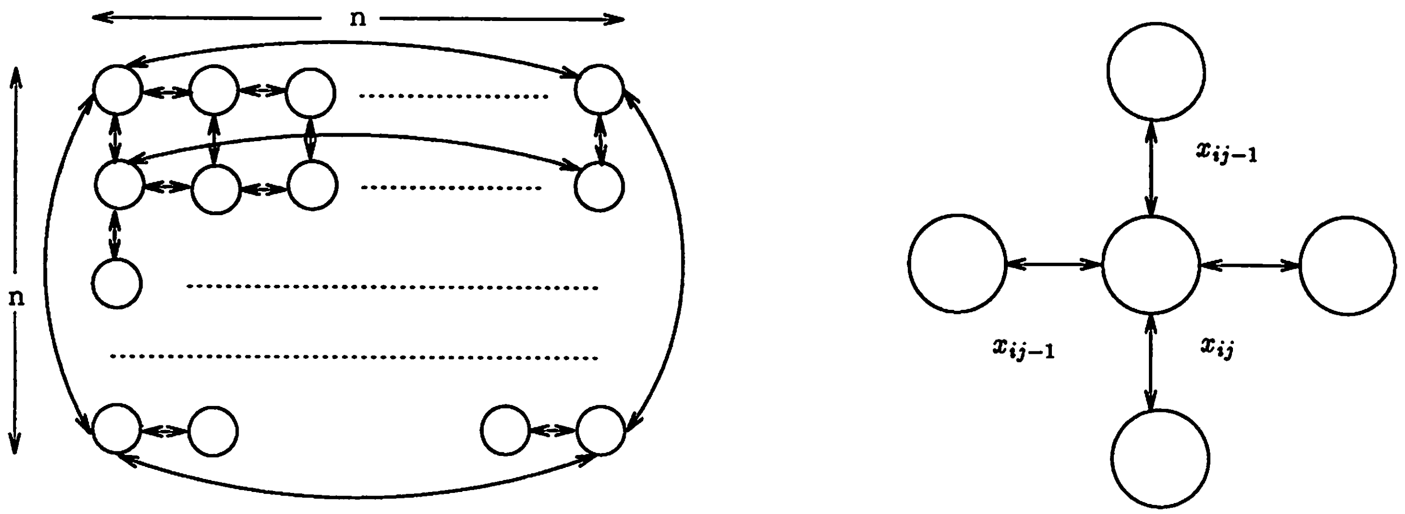


Figure 7: Graphic representation of the function f_n

Theorem B.1 *There are at least $2^{n/125}$ distinct cofactors of f_n with respect to the first $n^2/2$ variables.*

Proof: Observe each $x_{i,j}$ appears in 5 product terms. Since M^* has a bisection width of n, there are at least $n/5$ product terms $p_{i,j}$ where at least one variable in $p_{i,j}$ is from X_1 and at least variable is from X_2. Let P be the set of all such product terms, clearly $\mid P \mid \geq n$. There are at least $\mid P \mid /5$ product terms in P which have distinct variables from X_1. Let P' be a set of such product terms. Clearly $\mid P' \mid \geq (n/25)$. Again there are at least $(\mid P' \mid /5)$ product terms in P' which have a distinct variable from X_2. Let P'' be such a set. Clearly $\mid P'' \mid \geq (n/125)$.

Lemma B.1 *$\Phi = \prod_{p_{i,j} \in P''} p_{i,j}$ yields at least $2^{n/125}$ cofactors with respect to the variables in X_1.*

Proof: Each $p_{i,j} \in P''$ has a distinct variable from X_1 and from X_2. Let $X_1'' \subset X_1$ and $X_2'' \subset X_2$ refer to these sets variables. Consider only cofactors of Φ with respect to variables from X_1 where variables from $X_1 - X_1''$ are assigned zeros. Observe: *Distinct assignments to variables from X_1'' yield distinct cofactors of Φ.* For, let the assignments differ in the value given to some $x_{l_1,l_2} \in X_1''$ and let p_{t_1,t_2} be the corresponding term in Φ. Let x_{r_1,r_2} be the variable in p_{t_1,t_2} from X_2''. x_{r_1,r_2} is a controlling value for the cofactor in which x_{l_1,l_2} is assigned zero, and dominated in the cofactor in which x_{l_1,l_2} is assigned one. The lemma follows from the observation.

■

Since $f_n = \Phi \cdot \prod_{p_{i,j} \notin P''} p_{i,j}$, and the second term is non trivial, f_n has as many cofactors as Φ. Thus the theorem follows immediately from Lemma A.1. ■