On the Limitations of Ordered Representations of Functions

Jayram S. Thathachar*

Abstract. We demonstrate the limitations of various ordered representations that have been considered in the literature for symbolic model checking including BDDs [3], *-BMDs [6], HDDs [15], MTBDDs [13] and EVBDDs [25]. We introduce a lower bound technique that applies to a broad spectrum of such functional representations. Using an abstraction that encompasses all these representations, we apply this technique to show exponential size bounds for a wide range of integer and boolean functions that arise in symbolic model checking in the definition and implicit exploration of the state spaces. We give the first examples of integer functions including integer division, remainder, high/low-order words of multiplication, square root and reciprocal that require exponential size in all these representations. Finally, we show that there is a simple regular language that requires exponential size to be represented by any *-BMD, even though BDDs can represent any regular language in linear size.

1 Introduction

Model checking, proposed in [14], is a verification technique for determining whether a given property expressed as a temporal logic formula is satisfied by a system specification ([17] is an excellent source of references.) One of the major bottlenecks of model checking is the state explosion problem, i.e. the exponential growth in the number of states relative to the size of the system being verified.

Symbolic methods [9, 29, 8] have successfully combated this problem in many instances. Central to these methods is an underlying representation for various boolean and integer functions, and predicates combining such functions in arbitrary ways, in order to encode and implicitly explore state spaces. Ideally, these representations must satisfy certain important properties. First, they must be able to concisely represent the functions that occur in the definition of the components of the system being verified and arise in the implicit exploration of the state spaces. It is also necessary to combine these representations efficiently in order for composing boolean functions and integer functions using boolean and arithmetic operators, respectively. Finally, there should be efficient algorithms for testing various properties such as equivalence of representations and detecting satisfying assignments for boolean functions (more generally, finding roots of equations and inequalities involving boolean and integer functions).

BDDs (Ordered Binary Decision Diagrams) [3] are generalizations of decision trees to directed acyclic graphs, where the queries are made in some fixed order. Because they

* This work was supported by the National Science Foundation under Grant CCR-9303017.
Mailing address: Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, Washington 98195 E-mail: jayram@cs.washington.edu

are canonical, easy to manipulate, and compactly represent many boolean functions that are natural components of circuit designs, they have been useful in many instances for equivalence testing of circuit designs against their specification. After their importance to symbolic model checking was realized, various optimizations and heuristics [8] have resulted in enormously successful BDD-based verification packages.

The main drawback of BDDs is in concisely representing some important functions, particularly integer functions such as multiplication which requires exponential size [4]. Therefore other extensions and alternatives, e.g [13, 29, 6, 15], have been proposed to overcome some of the limitations of BDDs (see [5] for references to the "alphabet soup" of various representation schemes). *-BMDs [6] and HDDs [15] are two notable examples that are able to efficiently represent multiplication and other integer functions. They have been used to verify and identify errors in SRT division circuits [7, 16] similar to the one used in the Intel Pentium chip. *-BMDs obtain some of their power by treating the outputs of integer functions as a whole rather than splitting them into bits and have been used for verifying many arithmetic circuit designs that were previously intractable [6]. HDDs combine many of the advantages of BDDs and *-BMDs and thus have been successfully incorporated into verification packages, e.g. [11].

However, none of these representations are satisfactory for verifying general systems. A common feature of all the verification approaches is that the system is evaluated in a bottom-up manner to represent its transition relation. Therefore, the complexity of the various components that arise in this bottom-up evaluation limits the success of a representation scheme. Such components typically include many other integer functions such as division, reciprocal etc. and predicates such as linear equalities, e.g. $xy = c$. Another issue, which arises in verifying arithmetic circuits, is that the outputs are truncated, e.g. certain multiplication circuits require that both the high-order and low-order words of the product be represented efficiently.[1] This research is aimed at understanding the effectiveness of these representations in dealing with these functions and predicates.

We show that none of the representations referred to above, including recently defined representations [12], can represent a variety of specific integer and boolean functions concisely. Our specific results include

- Exponential bounds in the *-BMD and HDD representation for natural integer functions such as division (*Div*), remainder (*Mod*), high-order word (*HiMult*) and low-order word (*LoMult*) of multiplication, integer square root (*Sqrt*), and reciprocal (*Inv*). These are the first theoretical results that show the limitations of *-BMDs and HDDs in representing integer functions. Some of the functions listed above are natural components of microprocessor instruction sets that need to be verified.
- Exponential bounds for many boolean predicates including factor verification, string matching, selection/equality, shifted equality, and undirected graph predicates such as connectivity, s-t connectivity and bipartiteness that hold in all the representations considered above.
- A simple regular language that requires *-BMDs of exponential size. In contrast, BDDs can represent any regular language in linear size.

Existing lower bounds for BDDs [4] or even read-once branching programs [32] do not extend to *-BMDs and HDDs. We derive our lower bounds by defining an abstrac-

[1] R. E. Bryant. Private Communication.

tion, called the *Binary Linear Diagram* (BLD), that encompasses all the representations referred to above. We then show that for any function f, the rank of a certain matrix associated with f is a lower bound on the size of any BLD for f. This matrix is (essentially) the one usually used for VLSI AT^2 bounds, and has been studied extensively in communication complexity by theoreticians. (An excellent source for results and references for this area is [24].) Our lower bound technique is analogous to previously known results in multiplicity automata theory which relate the size of a multiplicity automaton to the rank of the Hankel matrix computed by the automaton [20, 10].

Our technique provides insight into the contrast between boolean and integer representations. For example, consider multiplication. For the boolean function which computes the middle bit of the product, one of our results shows that the associated matrix has exponential rank, but it can be easily verified that the matrix of the integer function has constant rank. This gives us better intuition as to why the integer function has linear-sized *-BMDs but the middle-bit version requires exponential size in all the ordered representations.

For the boolean predicates listed above, the exponential bounds on the rank are a corollary of two of the approaches used for bounding the best-partition communication complexity of boolean functions. In the approach taken in [28, 31, 4], one constructs exponentially large fooling sets. By a theorem of [18], these results imply exponential bounds for the rank. The second approach involves directly bounding the rank, as in [22] for the graph predicates stated above, although there are fewer results that use this approach. On the other hand, the exponential bounds that we prove for the integer functions mentioned previously do not follow from standard communication complexity results but from directly analyzing the associated matrices and bounding their rank.

As mentioned earlier, *-BMDs represent many arithmetic functions that require exponential size BDDs. Enders [19] obtained the first separation result in the other direction: the graph predicate that checks whether a graph is a triangle has polynomial-sized BDDs but requires exponential size *-BMDs. A variety of separation results have been shown in [1], contrasting the representational power of bit-level and word-level ordered representations. Our result for regular languages is the first (as far as we know) that shows such a separation for some natural language class. It also validates the belief in [6] that the strengths and weaknesses of *-BMDs and BDDs are orthogonal.

The paper is organized as follows. In Section 2, we define the BLD representation and illustrate how it generalizes all the ordered representations. Section 3 describes the basic lower bound technique of relating the BLD size of a function to the rank of certain matrices associated with that function. Applying this technique, we prove in Section 4 that the integer functions *Div*, *Mod*, *HiMult*, *LoMult*, *Sqrt* and *Inv* require exponential-sized BLDs. In Section 5, we give exponential lower bounds for many boolean functions by either using fooling sets or directly bounding the rank. Finally, in Section 6, we demonstrate for a simple regular language that the *-BMD complexity is exponential.

2 Binary Linear Diagrams

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of boolean variables. We consider functions that map boolean inputs (which assign 0-1 values to the variables) to elements of some fixed

ground field $\mathcal{K}$. We also consider *subfunctions* of a function f obtained by setting some of the input variables to 0-1 values. If σ is a partial assignment of 0-1 values to $Y \subseteq X$, we denote the resulting subfunction by $f\lceil_\sigma$ (which is defined on $X \backslash Y$).

Definition 1. *An* (Ordered) Binary Linear Diagram (BLD) *is a labeled, directed acyclic graph with a designated node called the* source. *The nodes that have out-degree* 0, *called the* sinks, *are labeled with elements from $\mathcal{K}$. Every other node v has out-degree two and the two edges directed from v are distinguished as the* 0-*edge and* 1-*edge, respectively. The node that the* 0-*edge (respectively,* 1-*edge) is incident to is called the* 0-*child (respectively,* 1-*child). The node v is labeled with a variable from X and a* 2×2 *matrix with entries in* $\mathcal{K}$.[2] *For some order* $O = x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ *on the variables, the BLD satisfies the constraint that the sequence of variables appearing in order along any path is a subsequence of O. The* size *of a BLD is defined as the number of nodes that it contains.*

We define the semantics of computation in a BLD by associating a node *function* g_v *with each node v: if v is a sink,* g_v *is a constant function as given by its label; if v is a non-sink, labeled with a* 2×2 *matrix* T_v *and a variable* x_{p_k} *for some k,* $1 \leq k \leq n$, g_v *is defined on the variable set* $\{x_{p_k}, x_{p_{k+1}}, \ldots, x_{p_n}\}$ *in terms of its* 0-*child u, and* 1-*child w, by* $\begin{bmatrix} (g_v)\lceil_{\overline{x_{p_k}}} \\ (g_v)\lceil_{x_{p_k}} \end{bmatrix} = T_v \begin{bmatrix} g_u \\ g_w \end{bmatrix}$. *The function* computed *by the BLD is the node function associated with the source.*

Note that unlike many of the ordered representations that have canonical representations of functions, it is possible to have different BLDs computing the same function. They are purely an abstraction of a large class of representations, used for proving lower bounds. For each representation, the corresponding BLD has the same underlying acyclic graph, variable and sink labels. The 2×2 matrix that labels any (non-sink) node is uniquely determined by the representation that the BLD corresponds to: for a BDD, the label is an identity matrix, for a *-BMD having no edge weights, it is $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, and for an HDD, it is the matrix that is assigned by the HDD to the variable label of that node. The following example illustrates how weights can be handled.

Example 1. Consider the integer multiplication function $f(x,y)$ for a pair of two-bit numbers $x = x_1x_0$ and $y = y_1y_0$. Figure 1 shows both the *-BMD representation and the corresponding BLD representation of f. Using our definition, the node function at node d is y_0 and at node c is $(1-y_1)\cdot(1\cdot y_0 + 0\cdot 2) + y_1\cdot(1\cdot y_0 + 1\cdot 2) = y_0 + 2\cdot y_1$.

3 The Rank Bound for BLDs

We now describe our main result for getting lower bounds on the BLD complexity of a function, that is, lower bounds on the BLD size that hold independent of the order of the variables. A slightly weaker result can be inferred from standard results on *multiplicity*

[2] Alternatively, we could have defined BLDs using edge variables and weights for abstracting non-deterministic ordered representations such as Parity-OBDDs. Our bounds apply to this alternate definition as well.

automata which have been previously considered in stochastic automata [10], theory of formal series [23], and learning theory [2]. Informally, a multiplicity automaton is similar to a non-deterministic automaton with weights (in some field $\mathcal{K}$) on transitions and states. It computes a function $f : \{0,1\}^* \to \mathcal{K}$ such that for each input w, $f(w)$ equals the sum over all paths conforming to w of the product of weights of the transitions and the last state along each such path. Define the *Hankel matrix* F associated with f as an infinite matrix whose rows and columns are indexed by strings in $\{0,1\}^*$. The $(x,y)^{\text{th}}$ entry of F for strings x and y is $f(x \circ y)$. It is known that the size of a minimal automaton computing f equals $\text{rank}(F)$ [10, 20].

Given any BLD P computing a function f that uses the order $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$, we transform it to a multiplicity automaton N via the following procedure. First, by adding dummy nodes, we transform P to a BLD P' of size at most $n \cdot \text{size}(P)$ in which no variable is missed along any source-sink path. Next, we view P' as a multiplicity automaton N: nodes of P' become states of N, and the source of P' becomes the start state of N. For a node v in P' with the associated matrix $\begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \end{bmatrix}$, whose 0-child and 1-child are u_0 and u_1 respectively, we define the weight of the edge (v, u_b) in N corresponding to the symbol b' to be $v_{b'b}$, where $b, b' \in \{0,1\}$. The weight of a sink is equal to its label and equals 0 for non-sink nodes. If we identify any string $b = b_1 b_2 \ldots b_n \in \{0,1\}^n$ with the input that assigns b_i to x_{p_i} for all i, then it is not too difficult to show that N also computes f. Therefore, $\text{size}(P) \geq \text{rank}(F)/n$.

For our purposes, we consider certain special submatrices of F. Fix a k, $0 \leq k \leq n$. Let $L = \{x_{p_1}, x_{p_1}, \ldots, x_{p_k}\}$ and R be the remaining variables. Consider the submatrix $M_f = M_{f,k}^{p_1,p_2,\cdots,p_n}$ of F whose rows and columns correspond to all the 0-1 assignments to L and R, respectively. Using a proof similar to the one that relates the size of a multiplicity automaton to the rank of the associated Hankel matrix, we can show that the rank of M_f is also a lower bound on the BLD size. A brief sketch of this proof is as follows:[3] For each input $\sigma : L \to \{0,1\}$, we can associate a unique node in the BLD that can be reached from the source by tracing the path of 0-edges and 1-edges according to σ and stopping as soon as either a sink or a node labeled with a variable of R is reached. Let V_k denote the set of nodes associated, in the manner described above, with all the 0-1 input assignments to L. A proof by induction on k shows that the subfunction $f\lceil_\sigma$, for any input $\sigma : L \to \{0,1\}$, is linearly related to the node functions associated with the nodes in V_k. Therefore, the matrix M_f can be expressed as a product $T \cdot H$, where H is a matrix of $|V_k|$ rows corresponding to all the node functions. From elementary linear algebra, $\text{rank}(M_f) \leq \text{rank}(H) \leq |V_k|$. automata results.

Theorem 1. *For any k, $0 \leq k \leq n$, and for any order of the variables $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$, let $M_{f,k}^{p_1,p_2,\cdots,p_n}$ denote the matrix where the $(\sigma,\pi)^{\text{th}}$ entry is $f(\sigma \cdot \pi)$, for each σ and π that assign 0-1 values to $\{x_{p_1}, x_{p_2}, \ldots, x_{p_k}\}$ and $\{x_{p_{k+1}}, x_{p_{k+2}}, \ldots, x_{p_n}\}$, respectively. Then, the size of any BLD that computes f, using an arbitrary order on the variables, is at least $\min_{p_1,p_2,\ldots,p_n} \max_k \text{rank}(M_{f,k}^{p_1,p_2,\cdots,p_n})$.*

[3] We can extend this proof to the case where the BLD is defined using edge weights and labelings. Here, the rank bound does not follow from multiplicity automata results.

Corollary 1. *The statement in Theorem 1 holds when we substitute any of the ordered representations such as BDDs, *-BMDs, and HDDs in place of BLDs.*

We will use a form of Theorem 1 that is easier to apply for proving exponential bounds on the rank. Notice that the rank of the matrix $M_f = M_{f,k}^{p_1,p_2,\ldots,p_n}$ depends only on L and R and not on the order of the variables in L or R. Therefore, denote this matrix by $M_f^{L,R}$. Let $\mathcal{P} \subseteq \{(L,R) | X = L \cup R\}$ be a family of partitions of X such that for every order $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ of the variables, there is at least one k such that $(\{x_{p_1}, x_{p_2}, \ldots, x_{p_k}\}, \{x_{p_{k+1}}, x_{p_{k+2}}, \ldots, x_{p_n}\}) \in \mathcal{P}$. It follows that the *best-partition rank*, defined as the minimum rank of $M_f^{L,R}$ over all partitions in $\mathcal{P}$, is a lower bound on the BLD size.

Example 2. Consider the multiplication function $f(x,y)$ of Example 1. Let $g(x,y)$ denote the middle (second least significant) bit of xy. Setting $L = \{x_0, x_1\}$ and $R = \{y_0, y_1\}$,

$$M_f = \begin{array}{c} \\ \overline{x_1}\overline{x_0} \\ \overline{x_1}x_0 \\ x_1\overline{x_0} \\ x_1x_0 \end{array} \begin{array}{c} \begin{array}{cccc} \overline{y_1}\overline{y_0} & \overline{y_1}y_0 & y_1\overline{y_0} & y_1y_0 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 9 \end{bmatrix} \end{array} \qquad M_g = \begin{array}{c} \\ \overline{x_1}\overline{x_0} \\ \overline{x_1}x_0 \\ x_1\overline{x_0} \\ x_1x_0 \end{array} \begin{array}{c} \begin{array}{cccc} \overline{y_1}\overline{y_0} & \overline{y_1}y_0 & y_1\overline{y_0} & y_1y_0 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

M_f has rank 1 (over reals or rationals), and remains constant at 1 for larger input sizes. In contrast, M_g has rank 2 over $GF[2]$. We will see shortly that this rank increases exponentially for larger input sizes (for any order of the variables).

4 Integer Functions with Exponential BLD Size

We consider integer functions of the form $f(x,y,\ldots)$ where $x,y,\ldots$ are n-bit integers encoded in binary. Define the division function $Div(x,y)$ as $\lfloor x/y \rfloor$ and the mod function $Mod(x,y)$ as x mod y. The functions $HiMult(x,y)$ and $LoMult(x,y)$ represent the high-order word and low-order word, respectively of the product xy, that is $HiMult(x,y) = \left\lfloor \frac{xy}{2^n} \right\rfloor$ and $LoMult(x,y) = xy \bmod (2^n)$. Finally, let the square root function $Sqrt(x)$ denote $\lfloor \sqrt{x} \rfloor$ and the reciprocal function $Inv(x)$ denote $\left\lfloor \frac{2^{2n}}{x} \right\rfloor$. The following theorem shows that the BLD complexity of each of these integer functions is exponential.

Theorem 2. *Let* BLD(f) *denote the minimum size of a BLD representing f over any field that includes the integers. Then,*

$$\text{BLD}(Div) \geq 2^{n/16} - 1 \qquad \text{BLD}(Mod) \geq 2^{n/16} - 3 \qquad \text{BLD}(HiMult) \geq 2^{n/16}$$

$$\text{BLD}(LoMult) \geq 2^{n/16} - 3 \qquad \text{BLD}(Sqrt) \geq 2^{\Omega(n)} \qquad \text{BLD}(Inv) \geq 2^{\Omega(n)}$$

Proof. Let the variable sets $X = x_{n-1}x_{n-2}\ldots x_0$, $Y = y_{n-1}y_{n-2}\ldots y_0$, etc., each represent n-bit integer inputs of a function f. For all the functions that we consider, we will choose a set $Z = U \cup V \subseteq X$ of $2m$ consecutive variables, where $U = \{x_{\ell+i} \mid 2m > i \geq m\}$, and $V = \{x_{\ell+i} \mid m > i \geq 0\}$, for some ℓ, m. Choose any (L,R) in the family of partitions $\{(L,R) \mid R = X \backslash L, |L \cap Z| = |Z|/2\}$. We refer to assignments to the variables of L (respectively, R) as *row* (respectively, *column*) assignments.

Proposition 1 ([4, Lemma 3]). *There exists an index set $I \subseteq \{1,2,\ldots,m\}$, with $|I| \geq m/8$, and integers p, $\ell+m \geq p \geq \ell+\max(I)$, and q, $\ell+2m \geq q \geq \ell+m+\max(I)$ such that the two sets $A = \{x_{q-k} \mid k \in I\} \subseteq U$ and $B = \{x_{p-k} \mid k \in I\} \subseteq V$, satisfy the property that either $A \subseteq L$ and $B \subseteq R$ or $A \subseteq R$ and $B \subseteq L$.*

Thus, the words U and V can be aligned in a such a way that the variables in A and B can be "matched" (see Figure 2). Without loss of generality, assume from the proposition above that $A \subseteq L$ and $B \subseteq R$. We will restrict all the variables that are *not* in $A \cup B$ (and only those variables) to certain fixed values that depend on f. We then show that the $2^{|I|} \times 2^{|I|}$ submatrix $N_f = N_f^{A,B}$ of M_f obtained by varying the row and column assignments in all possible ways but still conforming to the above restrictions has (almost) full rank. If $m = \Omega(n)$, then N_f and consequently M_f has exponential rank.

To simplify the presentation below, let $\widehat{I}$ denote $\max(I)$. Each set of boolean values b_k, $k \in I$, can be thought of as assigning values to the variables of A or B and thus can be associated with a unique row or column of N_f. We identify the set b_k, $k \in I$, with the (unique) number $\sum_{k \in I} b_k 2^{-k}$. Let $0 \leq s_1 < s_2 < \ldots < s_{2^{|I|}} < 1$ be all the numbers arising in this way. (Although these are rational, $2^{\widehat{I}} s_i$ is always an integer.) Permute the rows and columns of N_f so that the i^{th} row and and i^{th} column are associated with s_i, for $1 \leq i \leq 2^{|I|}$. Let t be the integer which corresponds to the fixed assignment of values to the variables of $X \backslash (A \cup B)$.[4] Note that the input X which corresponds to the i^{th} row and j^{th} column is $X_{ij} = 2^q s_i + 2^p s_j + t$. Our goal is to chose t in such a way that when computing $f(X_{ij},\ldots)$, the integers $2^q s_i$ and $2^p s_j$ will be multiplied by suitable factors so as to obtain a term which "aligns" s_i and s_j. Since these numbers affect the same bit positions, we will use the alignment to affect the value of f for the various X_{ij}'s in a way that N_f has almost full rank.

Summarizing the paradigm, for each function f we choose U and V, and apply Proposition 1 to obtain I, p, q, A, and B. We then fix the values of all the variables not in $A \cup B$ and show that the resulting submatrix N_f has almost full rank. We omit the proof for *Inv*, which appears in the full version.

(a) Div: Let $n = 2m$. Choose $U = \{x_{2m-1},\ldots,x_m\}$, and $V = \{x_{m-1},\ldots,x_0\}$ and apply Proposition 1 to obtain I, p, q, A, and B. Set each of the variables in $X \backslash (A \cup B)$ to 0 so that $X_{ij} = 2^q s_i + 2^p s_j$. Fix Y to be the integer $2^{q-\widehat{I}} + 2^{p-\widehat{I}}$ by setting both $y_{p-\widehat{I}}$ and $y_{q-\widehat{I}}$ to 1 and each remaining variable in Y to 0.

Observe that for all i, j, $X_{ij} = (2^{\widehat{I}} s_i)Y + 2^p(s_j - s_i)$. Since $|2^p(s_j - s_i)| < 2^p < Y$, it follows that (1) $Div(X_{ii}, Y) = 2^{\widehat{I}} s_i$ and (2) for all $j < i$, $Div(X_{ij}, Y) = 2^{\widehat{I}} s_i - 1$. Thus, from elementary linear algebra, $\text{rank}(N_{Div}) \geq 2^{|I|} - 1 \geq 2^{n/16} - 1$.

(b) Mod: For the same parameters considered in part (a), note that $N_{Mod} = M - Y \cdot N_{Div}$, where the $(i,j)^{\text{th}}$ entry of M is $2^q s_i + 2^p s_j (= X_{ij})$. It can be verified that M has rank 2, implying that $\text{rank}(N_{Mod}) \geq \text{rank}(N_{Div}) - \text{rank}(M) \geq 2^{n/16} - 3$.

(c) HiMult: Let $n = 2m$. Choose $U = \{x_{2m-1},\ldots,x_m\}$, $V = \{x_{m-1},\ldots,x_0\}$ and apply Proposition 1 to obtain I, p, q, A, and B. For each $k \in I' = \{1,2,\ldots,\widehat{I}\} \backslash I$, we set the

[4] In other words, if each $x_j \in X \backslash (A \cup B)$ is set to $c_j \in \{0,1\}$, then $\sum_{x_j \in X \backslash (A \cup B)} c_j 2^j = t$.

variable $x_{q-k} \in X\backslash(A \cup B)$ to 1, and all the remaining variables in $X\backslash(A \cup B)$ to 0; these variables form the integer $2^q r$, where $r = \sum_{k \in I'} 2^{-k}$. Therefore, $X_{ij} = 2^q(s_i + r) + 2^p s_j$. We also set both y_{2m-q} and y_{2m-p} to 1 and the remaining variables in Y to 0 so that the input Y corresponds to the integer $2^{2m-q} + 2^{2m-p}$. Now,

$$HiMult(X_{ij}, Y) = \left\lfloor \frac{X_{ij} \cdot Y}{2^{2m}} \right\rfloor = \left\lfloor \frac{(2^q(s_i + r) + 2^p s_j 2^{2m-p+q}) \cdot (2^{2m-q} + 2^{2m-p})}{2^{2m}} \right\rfloor$$
$$= 2^{q-p}(s_i + r) + \lfloor (s_i + s_j + r) + 2^{p-q} s_j \rfloor \qquad (1)$$

Because $s_i + s_j + r = a_{ij} 2^{-\widehat{I}}$, for some *integer* a_{ij}, and $2^{p-q} s_j < 2^{-\widehat{I}}$, the expression in Line 1 simplifies to $2^{q-p}(s_i + r) + \lfloor s_i + s_j + r \rfloor$.

Suppose $s_i = \sum_{k \in I} b_k 2^{-k}$, for some b_k, $k \in I$. If $i^* = 2^{|I|} - i + 1$, then $s_{i^*} = \sum_{k \in I} \overline{b_k} 2^{-k}$, that is, s_{i^*} is the one's complement of s_i with respect to the bit positions in I. Therefore, $s_i + s_{i^*} = \sum_{k \in I} 2^{-k}$, implying that $s_i + s_{i^*} + r = \sum_{i=1}^{\widehat{I}} 2^{-i} = 1 - 2^{-\widehat{I}}$.

We have the following two cases. When $j \leq i^*$, $\lfloor s_i + s_j + r \rfloor \leq \lfloor s_i + s_{i^*} + r \rfloor = 0$. Thus, $HiMult(X_{ij}, Y) = 2^{q-p}(s_i + r)$. On the other hand, $s_i + s_{i^*+1} + r \geq s_i + s_{i^*} + 2^{-\widehat{I}} + r = 1$, so $\lfloor s_i + s_{i^*+1} + r \rfloor \geq 1$. Therefore, $HiMult(X_{i^*+1,j}, Y) \geq 2^{q-p}(s_i + r) + 1$. It follows that $\text{rank}(N_{HiMult}) \geq 2^{|I|} - 1 \geq 2^{n/16} - 1$.

(d) LoMult: With the same parameters as in part (c), $N_{LoMult} = M - 2^m N_{HiMult}$, where the $(i,j)^{\text{th}}$ entry of M is $X_{ij} \cdot Y = (2^{2m-q} + 2^{2m-p}) X_{ij}$. Therefore,

$$\text{rank}(N_{LoMult}) \geq \text{rank}(N_{HiMult}) - \text{rank}(M) \geq \text{rank}(N_{HiMult}) - 2 \geq 2^{n/16} - 3.$$

(e) Sqrt: Let $n = 10m$, $U = \{x_{5m-1}, x_{5m-2}, \ldots, x_{4m}\}$, $V = \{x_{4m-1}, x_{4m-2}, \ldots, x_{3m}\}$, for large enough m, and apply Proposition 1 to obtain I, p, q, A, and B. Fix each of the variables $x_{2q-2\widehat{I}-2}, x_{2p-2\widehat{I}-2}, x_{p+q-2\widehat{I}-1}$ to 1 (which are in $X\backslash(A \cup B)$ because $2p - 2\widehat{I} - 2 > 5m$) and all the remaining variables in $X\backslash(A \cup B)$ to 0. Therefore, $X_{ij} = r^2 + 2^q s_i + 2^p s_j$, where $r = 2^{q-\widehat{I}-1} + 2^{p-\widehat{I}-1}$.

We claim that for $i \geq 2$, (a) $X_{ii} < (r + 2^{\widehat{I}} s_i)^2 < X_{i,i+1}$ and (b) for all $j \leq i$, $X_{ij} \geq (r + 2^{\widehat{I}} s_i - 1)^2$, which would imply the desired inequality, $\text{rank}(N_{Sqrt}) \geq 2^{|I|} - 2 = 2^{\Omega(n)}$. Observe that $(r + 2^{\widehat{I}} s_i)^2 = r^2 + 2^q s_i + 2^p s_i + (2^{\widehat{I}} s_i)^2$. Since $3\widehat{I} \leq 3m \leq p$, it follows that $(2^{\widehat{I}} s_i)^2 < 2^{2\widehat{I}} \leq 2^{p-\widehat{I}} \leq 2^p(s_{i+1} - s_i)$, proving part(a) of the claim.

Since $p \geq 3\widehat{I}$ and $q - p \geq \widehat{I}$, part(b) of the claim follows by verifying for each j that

$$(r + 2^{\widehat{I}} s_i - 1)^2 = r^2 + 2^q s_i - (2^{q-\widehat{I}} - 2^p s_i) - (2^{p-\widehat{I}} - (2^{\widehat{I}} s_i - 1)^2) \leq r^2 + 2^q s_i \leq X_{ij}.$$

5 Boolean Functions with Exponential BLD Size

For a fixed partition of X into L and R, the matrix $M_f^{L,R}$ has been used to study communication complexity of f ([33]). Among the approaches that give lower bounds on this measure are (i) constructing large *boolean fooling sets* and (ii) computing the rank of $M_f^{L,R}$ [30]. A fooling set consists of pairs of input assignments to L and R such that for

any two distinct pairs (σ,π) and (σ',π'), $f(\sigma\cdot\pi)=f(\sigma'\cdot\pi')$, but $f(\sigma\cdot\pi')\neq f(\sigma'\cdot\pi)$. The following proposition [18], which extends to unequal-sized partitions, shows that exponentially large fooling sets imply exponential rank.

Proposition 2 ([18]). *For any boolean function f, and any equipartition of its variable set into L and R, let $M_f^{L,R}$ be the associated matrix of f with respect to this partition. Let $r=\mathrm{rank}(M_f^{L,R})$ over any field. If s is the size of a fooling set, then $r\geq\sqrt{s}-1$.*

For the best-partition rank, the more relevant measure is the best-partition communication complexity [31] in which one computes the communication complexity for the best choice of a partition into L and R in some appropriate family of partitions. By Proposition 2 and the discussion following the statement of Theorem 1, any function for which lower bounds on the best-partition communication complexity have been proved either by constructing exponential size fooling sets or by proving exponential rank bounds for all partitions imply exponential bounds on the BLD complexity. Examples of such functions are in [28], [31], [4], [27] and [22], of which we list some below.

Corollary 2. *The following predicates require BLDs of exponential size:*

PATTERN MATCHING: *Verify if the binary pattern string of αn bits occurs in the binary text string of $(1-\alpha)n$ bits, where $0<\alpha<1$.*

FACTOR VERIFICATION: *Verify if two n-bit numbers multiply to a $2n$-bit number.*

MIDDLE BIT OF PRODUCT: *Does the middle bit of the product of two n bit numbers equal 1?*

SELECTION/EQUALITY TESTING: *For two n bit numbers, x and y such that x has $n/2$ bits set to 1, check if the $n/2$-bit number obtained by selecting the bits in y at positions corresponding to 1s in x equals the remaining $n/2$-bit number in y.*

SHIFTED EQUALITY: *Given two input strings and a number i, does the first string equal the second shifted circularly to the right i times? (Lam and Ruzzo [26] generalized this to show that any function f that has a large fooling set under some* fixed *partition has a shifting version that has large fooling sets under* all *partitions. However, these shifted versions may not be natural.)*

GRAPH PROPERTIES: *Verifying any of the following predicates on undirected graphs: Connectivity, Bipartiteness, and s-t-Connectivity.*

6 *-BMDs and Regular Languages

We saw earlier that the rank approach is useful for proving bounds that apply uniformly to all the ordered representations. A related and important problem is to contrast specific representations in order to understand what representations are best suited for a class of functions or languages. For *regular* languages, we know that BDDs can represent any regular language in *linear* size by keeping track of the state in the automaton that represents it. The following theorem shows that there is a simple regular language that has exponential complexity in the *-BMD representation. The proof is omitted and appears in the full paper.

Theorem 3. *For $i=0,2,3,4$, let $A_i=\{w\in\{0,1\}^7 : w \text{ has } i \text{ 1s}\}$. Any *-BMD representing the regular language $S=A_0^*A_3(A_0\cup A_2)^*\cup A_0^*A_4A_0^*$ requires size $2^{\Omega(n)}$.*

7 Conclusions

We have shown that a variety of integer functions such as integer division, remainder, high/low-order words of multiplication, square root, and reciprocal require exponential-sized BLDs. We then showed similar results for a variety of boolean functions by relating its complexity to two measures, the fooling set size and the rank. The generality in the BLD definition implies that minor variations in the known ordered representations will not be sufficient and we may have to consider non-linear definitions to be able to handle the hard functions. Another approach is to consider read-once representations that relax the notion of an implicit order on the variables, e.g. Free Binary Decision Diagrams [21] and their generalizations similar to BLDs.

References

1. B. Becker, R. Drechsler, and R. Enders. On the computational power of bit-level and word-level decision diagrams. In *4. GI/ITG/GME Workshop zur Methoden des Entwurfs und der Verifikation Digitaler Systeme*, Berichte aus der Informatik, pages 71–80, Kreischa, March 1996. Shaker Verlag, Aachen.
2. Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. On the applications of multiplicity automata in learning. In *37th Annual Symposium on Foundations of Computer Science*, Burlington, Vermont, 14–16 October 1996. IEEE.
3. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
4. R. E. Bryant. On the complexity of VLSI implementations and graph representations of boolean functions with application to integer multiplication. *IEEE Transactions on Computers*, 40(2):205–213, February 1991.
5. R. E. Bryant. Binary decision diagrams and beyond: Enabling technologies for formal verification. In *International Conference on Computer Aided Design*, pages 236–245, Los Alamitos, Ca., USA, November 1995. IEEE Computer Society Press.
6. R.E. Bryant and Y.-A. Chen. Verification of arithmetic circuits with binary moment diagrams. In *32nd ACM/IEEE Design Automation Conference*, Pittsburgh, June 1995.
7. R.E. Bryant and Y.-A. Chen. Bit-level analysis of an SRT divider circuit. In *33rd ACM/IEEE Design Automation Conference*, 1996.
8. J.R. Burch, E.M. Clarke, D.E. Long, K.L. MacMillan, and D.L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, April 1994.
9. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., June 1990. IEEE CS Press.
10. J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40, February 1971.
11. Y.-A. Chen, E. Clarke, P. H. Ho, Y. Hoskote, T. Kam, M. Khaira, J. O'Leary, and X. Zhao. Verification of all circuits in a foating-pont unit using word-level model checking. In *First International Conference on Formal Methods in Computer-Aided Design*, volume 1166 of *Lecture Notes Comp. Sci.*, pages 19–33, Palo Alto, CA, November 1996. Springer Verlag.
12. Ying-An Chen and R.E. Bryant. *PHDD: an efficient graph representation for floating point circuit verification. In *International Conference on Computer Aided Design*, pages 2–7, Los Alamitos, Ca., USA, November 1997. IEEE Computer Society Press.

13. E. Clarke, K.L. McMillian, X. Zhao, M. Fujita, and J.C.-Y. Yang. Spectral transforms for large boolean functions with application to technologie mapping. In *30th ACM/IEEE Design Automation Conference*, pages 54–60, Dallas, TX, June 1993.
14. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons from branching time temporal logic. *Lecture Notes Comp. Sci.*, 131:52–71, 1982.
15. E. M. Clarke, M. Fujita, and X. Zhao. Hybrid decision diagrams — overcoming limitations of MTBDDs and BMDs. In *International Conference on Computer Aided Design*, pages 159–163, Los Alamitos, CA, November 1995. IEEE Computer Society Press.
16. E. M. Clarke, S. M. German, and X. Zhao. Verifying the SRT division algorithm using theorem proving techniques. *Lecture Notes in Computer Science*, 1102, 1996.
17. Edmund M. Clarke and Jeanette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, December 1996.
18. M. Dietzfelbinger, J. Hromkovic, and G. Schnitger. A comparison of two lower bound methods for communication complexity. In *Symposium on Mathematical Foundations of Computer Science*, pages 326–335, 1994.
19. R. Enders. Note on the complexity of binary moment diagram representations. In *IFIP WG 10.5 Workshop on Applications of Reed-Muller Expansion in Circuit Design*, pages 191–197, 1995.
20. M. Fliess. Matrices de Hankel. *J. Math. Pures et Appl.*, 53:197–224, 1974.
21. J. Gergov and Ch. Meinel. Efficient boolean manipulation with OBDD's can be extended to read-once only branching programs. *IEEE Transactions on Computers*, 43(10):1197–1209, October 1994.
22. András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 186–191, Chicago, Illinois, 2–4 May 1988.
23. Harju and Karhumaki. The equivalence problem of multitape finite automata. *Theoretical Computer Science*, 78, 1991.
24. Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, Cambridge [England] ; New York, 1997.
25. Y.-T. Lai and S. Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *29th ACM/IEEE Design Automation Conference*, pages 608–613, 1992.
26. Tak Wah Lam and Larry Ruzzo. Results on communication complexity classes. *Journal of Computer and System Sciences*, 44, 1992.
27. Thomas Lengauer. VLSI theory. In *Handbook of Theoretical Computer Science*, volume 1. The MIT Press/Elsevier, 1990.
28. Richard J. Lipton and Robert Sedgewick. Lower bounds for VLSI. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computation*, pages 300–307, Milwaukee, Wisconsin, 11–13 May 1981.
29. K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
30. Kurt Mehlhorn and Erik M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 330–337, San Francisco, California, May 1982.
31. C. Papadimitriou and M. Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28, 1984.
32. Stephen Ponzio. A lower bound for integer multiplication with read-once branching programs. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 130–139, Las Vegas, Nevada, 29 May–1 June 1995.
33. Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 209–213, Atlanta, Georgia, 30 April–2 May 1979.

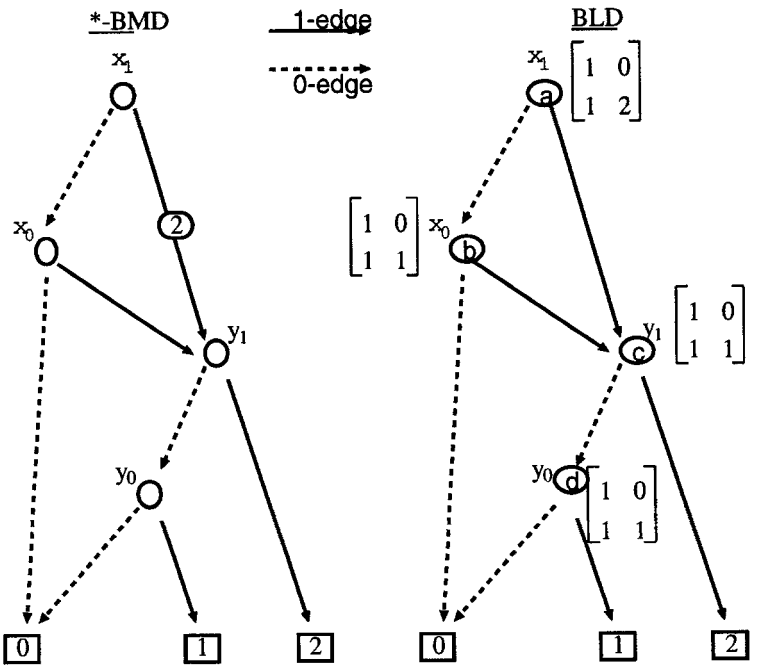

Fig. 1. *The *-BMD (left) and BLD(right) for multiplication using the order* x_1, x_0, y_1, y_0.

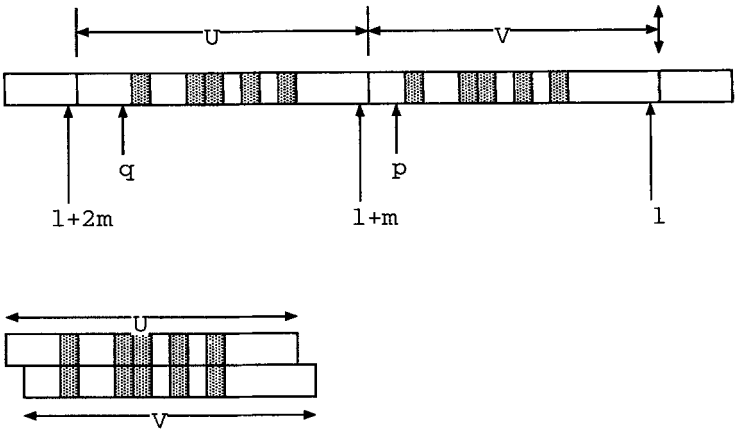

Fig. 2. *This figure illustrates Proposition 1. The sets* $A \subseteq U$ *and* $B \subseteq V$ *(shown shaded in the figure) can be matched by suitably aligning the words* U *and* V.