Integer Multiplication and the Complexity of Binary Decision Diagrams

Beate Bollig*

Abstract

Integer multiplication as one of the basic arithmetic functions has been in the focus of several complexity theoretical investigations and ordered binary decision diagrams (OBDDs) are one of the most common dynamic data structures for Boolean functions. The BDD complexity of two output bits of integer multiplication, the so-called middle bit and the most significant bit, has been investigated intensively. In this column we briefly survey results on the complexity of restricted binary decision diagrams for integer multiplication and concentrate on two recent results on the complexity of OBDDs for the most significant bit. Our aim is not to be comprehensive but to deepen the knowledge on the structure of integer multiplication.

1 Introduction

Integer multiplication is certainly one of the most important functions in computer science and a lot of effort has been spent in designing good algorithms and small circuits and in determining its complexity. For some computation models integer multiplication is a quite simple function. It is contained in NC^1 and even in $TC^{0,3}$ (polynomial-size threshold circuits of depth 3) but neither in AC^0 (polynomial-size $\{\vee, \wedge, \neg\}$-circuits of unbounded fan-in and constant depth) nor in $TC^{0,2}$ [21]. For more than 35 years the algorithm of Schönhage-Strassen [30] has been the fastest method for integer multiplication running in time $O(n \log n \log\log n)$. Only recently, Fürer has presented an algorithm running in time $n \log n \cdot 2^{O(\log^* n)}$, where the running time holds for multitape Turing machines [16]. An algorithm with the same running time based on modular arithmetic has been obtained by De, Kurur, Saha,

*TU Dortmund, LS 2 Informatik, beate.bollig@uni-dortmund.de

and Sapthariski [13]. Until now it is open whether integer multiplication is possible in time $O(n \log n)$.

Definition 1.1. Let $B_{n,m}$ denote the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}^m$ and B_n the special case that $m = 1$. The Boolean function $\mathrm{MUL}_{i,n} \in B_{2n}$ maps two n-bit integers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$ to the ith bit of their product, i.e., $\mathrm{MUL}_{i,n}(x,y) = z_i$, where $x \cdot y = z_{2n-1} \dots z_0$ and x_0, y_0, z_0 denote the least significant bits. For $c \in \{0,1\}^n$ the Boolean function $\mathrm{MUL}^c_{i,n} \in B_n$ is defined by $\mathrm{MUL}^c_{i,n}(x) = \mathrm{MUL}_{i,n}(x,c)$.

The Boolean function MUL-Graph$_n \in B_{4n}$ maps two n-bit integers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$, and a $2n$-bit integer $z = z_{2n-1} \dots z_0$ to 1 iff the product of x and y equals z.

Besides Boolean circuits and formulas, circuits whose underlying graph is a tree after a suitable duplication of the inputs, branching programs (BPs) are one of the standard representations for Boolean functions. (For a history of results on branching programs see, e.g., the monograph of Wegener [32]).

Definition 1.2. A *branching program* (BP) on the variable set $X_n = \{x_1, \dots, x_n\}$ is a directed acyclic graph with one source and two sinks labeled by the constants 0 and 1. Each non-sink node (or decision node) is labeled by a Boolean variable and has two outgoing edges, one labeled by 0 and the other by 1. An input $b \in \{0,1\}^n$ activates all edges consistent with b, i.e., the edges labeled by b_i which leave nodes labeled by x_i. A *computation path* for an input b in a BP G is a path of edges activated by the input b which leads from the source to a sink. A computation path for an input b which leads to the 1-sink is called *accepting path* for b. The BP G represents a function $f \in B_n$ for which $f(b) = 1$ iff there exists an accepting path for the input b. The *size* of a branching program G is the number of its nodes. The *branching program size* of a Boolean function f is the size of the smallest BP representing f. The *length* of a branching program is the maximum length of a path.

Not only in complexity theory but also in applications people have used (restricted) branching programs, where they are most often called binary decision diagrams (BDDs). Representations of Boolean functions that allow efficient algorithms for many operations, in particular synthesis (combine two functions by a binary operation) and equality test (do two representations represent the same function?) are necessary. Bryant [11] introduced ordered binary decision diagrams (OBDDs) which have become one of the most popular data structures for Boolean functions. Among the many areas of application are verification, model checking, computer-aided design, and symbolic graph algorithms.

Lower and upper bounds for integer multiplication are motivated by the general interest in the complexity of important arithmetic functions. The complexity of two output bits of integer multiplication has been investigated intensively in the last years. The first one is the middle bit of integer multiplication, the bit with significance 2^{n-1}, which is the hardest bit to compute for space bounded models of computation in the sense that if it can be computed with size $s(n)$, then any other bit can be computed with size at most $s(2n)$. More precisely, any branching program for $\mathrm{MUL}_{2n-1,2n}$ can be converted into a branching program representing $\mathrm{MUL}_{i,n}$, $0 \leq i \leq 2n-1$, by relabeling the nodes and by replacing some inputs with the constant 0. As a consequence the first large lower bounds on the size of restricted branching programs have been shown for $\mathrm{MUL}_{n-1,n}$. The second one is the bit z_{2n-1} which is the most important bit of integer multiplication in the following sense. Since it has the highest value, for the approximation of the value of the product of two n-bit numbers x and y it is the most interesting one. On the other hand for space bounded models of computation z_{2n-1} is easy to compute in the sense that if it cannot be computed with size $s(n)$, then any other bit z_i, $2n-1 > i \geq 0$, cannot be computed with size $s(i/4)$.

In the following we give some motivation for the investigation of the OBDD size of integer multiplication.

The middle bit of integer multiplication

A lot of effort has been spent in trying to verify multiplier circuits using OBDDs. In 1998 an OBDD for the 16-bit multiplication circuit $c6288$, one of the most important ISCAS (International Symposium on Circuits and Systems) benchmark circuits, has been constructed [39]. To the best of our knowledge, until now it has been impossible to construct OBDDs for input length $n = 32$ and even the representation of all output bits of 16-bit multiplication by SBDDs, a more general OBDD model for the representation of multiple output Boolean functions, is a challenging task. Since the size of OBDDs and SBDDs can be quite sensitive to the chosen variable order, one of the reasons might be that different output bits of integer multiplication have different variable orders leading to reasonable size (for the definition of OBDDs and variable orders see Definition 2.2). Bryant [11] has already bounded the size of SBDDs for integer multiplication by proving that for each variable order there exists an output bit for which the OBDD size is at least $2^{n/8}$. For many applications it would be sufficient to represent each output bit by an OBDD of moderate size according to an suitably chosen variable order. Already Bryant has destroyed this hope in 1991 [12]. He has shown that OBDDs for the representation of the middle bit of integer multiplication

have at least size $2^{n/8}$ for any variable order. Nevertheless, Bryant's lower bound does not exclude that even 256-bit multiplication can be represented in reasonable size. Therefore, the OBDD size for integer multiplication has been further investigated.

The most significant bit of integer multiplication

In the last years a new research branch has emerged which is concerned with the theoretical design and analysis of so-called symbolic algorithms for classical graph problems on OBDD-represented graph instances (see, e.g., [17, 18], [28], and [38]). Symbolic algorithms have to solve problems on a given graph instance by efficient functional operations offered by the OBDD data structure. Therefore, at the beginning the OBDD-based algorithms have been justified by analyzing the number of executed OBDD operations (see, e.g., [17, 18]). Since the runtime of an operation on an OBDD G often depends on the size of G the analysis of the over-all runtime of symbolic methods including the analysis of all OBDD sizes occurring during such an algorithm is more significant (see, e.g., [38]). In order to investigate the limits of symbolic graph algorithms for the all pairs shortest paths problem Sawitzki [28] has investigated the graph of integer multiplication and has presented an exponential lower bound on its OBDD size. Afterwards he has defined inputs for the all pairs shortest paths problem such that during the computation representations for MUL-Graph$_n$ are necessary. Another investigated graph problem is the following. Computing the set of nodes that are reachable from some source $s \in V$ in a digraph $G = (V, E)$ is an important problem in computer-aided design, hardware verification, and model checking. Proving exponential lower bounds on the space complexity of a common class of OBDD-based algorithms for this reachability problem, Sawitzki [29] has presented the first exponential lower bound on the size of π-OBDDs representing the most significant bit for the variable order π where the variables are tested according to increasing significance, i.e. $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. For the lower bounds on the space complexity of the OBDD-based algorithms he has used the assumption that the output OBDDs use the same variable order as the input OBDDs. But in contrast, practical algorithms usually run variable reordering heuristics on intermediate OBDD results in order to minimize their size. Therefore, lower and upper bounds on the OBDD size of the most significant bit of multiplication with respect to an arbitrary variable order are interesting.

Organization

In this column we give a brief overview on results concerning the complexity of restricted branching programs or binary decision diagrams for the functions $\mathrm{MUL}_{n-1,n}$ and $\mathrm{MUL}_{2n-1,n}$. We do not aim to be comprehensive but focus on their OBDD complexity, in particular new results on the most significant bit. Since the article is meant to be self-contained, in Section 2 we start with the presentation of some restricted branching program or binary decision diagram models. Moreover, we repeat the relevant relation between one-way communication complexity and the size of OBDDs.

Section 3 contains results on the size of restricted branching programs or binary decision diagrams for the middle bit of integer multiplication. We present lower and upper bounds on the OBDD size and sketch results on the size of more general models.

The main results of this survey are presented in Section 4. Using only methods from one-way communication complexity Sawitzki's restricted lower bound on the size of OBDDs for $\mathrm{MUL}_{2n-1,n}$ [29] is improved. Afterwards a general lower bound and upper bounds are presented. Remarks on the size of more general models for the most significant bit complete our investigation.

Finally, in Section 5 we summarize our results by a comparison between the middle and the most significant bit.

2 Preliminaries

In this section we introduce some notation. Furthermore, we give an overview on some restricted branching program or binary decision diagram models and provide relevant technical background from communication complexity.

2.1 Notation

In the rest of the paper we use the following notation.

Let $[x]_r^l$, $n-1 \geq l \geq r \geq 0$, denote the bits $x_l \dots x_r$ of a binary number $x = (x_{n-1}, \dots, x_0)$. For the ease of description we use the notation $[x]_r^l = z$ if $(x_l, \dots, x_r)$ is the binary representation of the integer $z \in \{0, \dots, 2^{l-r+1}-1\}$. Sometimes, we identify $[x]_r^l$ with z if the meaning is clear from the context. We use the notation $(z)_r^l$ for an integer z to identify the bits at position $l, \dots, r$ in the binary representation of z.

Let $\ell \in \{0, \dots, 2^m - 1\}$, then $\overline{\ell}$ denotes the number $(2^m - 1) - \ell$. For a binary number $x = (x_{n-1}, \dots, x_0)$ we use the notation $\overline{x}$ for the binary number $(\overline{x}_{n-1}, \dots, \overline{x}_0)$.

Let a_S be an assignment to variables in a set S and $a_S(x_k) \in \{0,1\}$ be the assignment to $x_k \in S$, then we define $\|a_S\| := \sum_{x_k \in S} a_S(x_k) \cdot 2^k$.

In the following for the sake of simplicity we do not apply floor or ceiling functions to numbers even when they need to be integers whenever this is clear from the context and has no bearing on the essence of the presented proofs.

2.2 Restricted branching programs or binary decision diagrams

It is well known that the logarithm of the branching program size is essentially the same as the space complexity of the nonuniform variant of Turing machines (see, e.g., [32]). Hence, it is a fundamental open problem to prove superpolynomial lower bounds on the size of branching programs for explicitly defined Boolean functions, i.e., functions contained in NP. In order to develop and strengthen lower bound techniques one considers restricted computation models. There are several possibilities to restrict branching programs, among them restrictions on the multiplicity of variable tests or the order in which variables may be tested.

Definition 2.1. i) A branching program is called (syntactically) *read-k-times* (BPk) if each variable is tested on each path at most k times.

ii) A branching program is called *s-oblivious* for a sequence of variables $s = (s_1, \ldots, s_l)$, $s_i \in X_n$, or short *oblivious*, if the set of decision nodes can be partitioned into disjoint sets V_i, $1 \leq i \leq l$, such that all nodes from V_i are labeled by s_i and the edges which leave V_i-nodes reach a sink or a V_j-node where $j > i$. The *length* of an s-oblivious branching program is the length of the sequence s.

Nondeterministic branching programs and randomized branching programs are defined in the obvious way by introducing additional, unlabeled nodes at which nondeterministic or randomized decisions, resp., are taken. An approximating branching program for a Boolean function f with (two-sided) error ε is a deterministic branching program computing an ε-approximation of f, i.e., a function that differs from f on at most an ε-fraction of the inputs.

For nondeterministic read-once branching programs a further generalization of obliviousness can be obtained by restricting the order of variables in such a way that it equals for each input the order of variables for this input performed in a complete given deterministic read-once branching program, i.e., a BP1 where on each path from the source to the sinks all variables

are tested. This complete read-once branching program is called graph order and the resulting nondeterministic read-once branching program is called graph-driven. If the graph order is a tree of polynomial size, then it is called tree-driven.

Combining restrictions on the multiplicity of variable tests with the property of obliviousness we obtain oblivious read-once branching programs, better known as OBDDs.

Definition 2.2. An OBDD is a branching program with a *variable order* given by a permutation π on the variable set. On each path from the source to the sinks, the variables at the nodes have to appear in the order prescribed by π (where some variables may be left out). A π-OBDD is an OBDD ordered according to π. The π-OBDD size of f denoted by π-OBDD(f) is the size of the smallest π-OBDD representing f. The OBDD size of f, sometimes also called OBDD complexity of f, (denoted by OBDD(f)) is the minimum of all π-OBDD(f).

The size of the minimal π-OBDD representing a Boolean function f on n variables, i.e., $f \in B_n$, is described by the following structure theorem [31].

Theorem 2.3. *The number of $x_{\pi(i)}$-nodes of the minimal* π-OBDD *for f is the number s_i of different subfunctions $f_{|x_{\pi(1)}=a_1,\ldots,x_{\pi(i-1)}=a_{i-1}}$, $a_1,\ldots,a_{i-1} \in \{0,1\}$, essentially depending on $x_{\pi(i)}$ (a function g depends essentially on a variable z if $g_{|z=0} \neq g_{|z=1}$).*

It is well known that the size of an OBDD representing a function f depends on the chosen variable order. Since in applications the variable order is not given in advance we have the freedom (and the problem) to choose a good or even an optimal order for the representation of f. In general OBDDs do not have nice algorithmic properties. There are examples known such that g_n and h_n are two Boolean functions which have OBDDs of linear size (for different variable orders) but $f_n = g_n \vee h_n$ has even exponential BP1 size (for an example see, e.g., Proposition 2 in [6]). If a variable order π is fixed, all important operations can be performed efficiently.

SBDDs (shared binary decision diagrams) are an extension of OBDDs that can express multiple functions. An SBDD represents a Boolean function $f \in B_{n,m} : \{0,1\}^n \to \{0,1\}^m$ by representing simultaneously the output functions $f_1, f_2, \ldots, f_m$ of f, where the representations for the different coordinate functions $f_1, f_2, \ldots, f_m$ may share nodes.

2.3 One-way communication complexity and the size of OBDDs

In order to obtain lower bounds on the size of OBDDs one-way communication complexity has become a standard technique (see Hromkovič [22] and Kushilevitz and Nisan [23] for the theory of communication complexity and the results mentioned below).

The main subject is the analysis of the following (restricted) communication game. Consider a Boolean function $f \in B_n$ which is defined on the variables in $X_n = \{x_1, \ldots, x_n\}$, and let $\Pi = (X_A, X_B)$ be a partition of X_n. Assume that Alice has only access to the input variables in X_A and Bob has only access to the input variables in X_B. In a one-way communication protocol, upon a given input x, Alice is allowed to send a single message (depending on the input variables in X_A) to Bob who must then be able to compute the answer $f(x)$. The *one-way communication complexity* of the function f denoted by $C(f)$ is the worst case number of bits of communication which need to be transmitted by such a protocol that computes f. It is easy to see that an OBDD G with respect to a variable order where the variables in X_A are tested before the variables in X_B can be transformed into a communication protocol and $C(f) \leq \lceil \log |G| \rceil$. Therefore, linear lower bounds on the communication complexity of a function $f : \{0,1\}^{|X_A|} \times \{0,1\}^{|X_B|} \to \{0,1\}$ lead to exponential lower bounds on the size of π-OBDDs where the X_A-variables are before the X_B-variables in π.

One central notion of communication complexity are strong fooling sets which play an important role in the lower bound proofs later on.

Definition 2.4. Let $f : \{0,1\}^{|X_A|} \times \{0,1\}^{|X_B|} \to \{0,1\}$. A set $S \subseteq \{0,1\}^{|X_a|} \times \{0,1\}^{|X_B|}$ is called *strong fooling set* for f if $f(a,b) = c$ for all $(a,b) \in S$ and some $c \in \{0,1\}$ and if for different pairs $(a_1,b_1),(a_2,b_2) \in S$ at least one of $f(a_1,b_2)$ and $f(a_2,b_1)$ is unequal to c.

Theorem 2.5. *If $f : \{0,1\}^{|X_A|} \times \{0,1\}^{|X_B|} \to \{0,1\}$ has a strong fooling set of size t, the communication complexity of f is bounded below by $\lceil \log t \rceil$.*

Because of our considerations above, the size t of a strong fooling set for f is a lower bound on the size of OBDDs representing f with respect to a variable order where the variables X_A are tested before the variables X_B. Because of the symmetric definition of strong fooling sets, t is also a lower bound on the size of OBDDs representing f with respect to a variable order where the variables X_B are tested before the variables X_A. The crucial step to prove large lower bounds on the OBDD complexity of a function is to obtain for all partitions of the variables large lower bounds on the size of

fooling sets for subfunctions of the given function (best case communication complexity).

In the rest of this section our aim is to define a function f_n with large communication complexity which is a main ingredient in our lower bound proof on the OBDD size of the most significant bit of integer multiplication.

First, we take a look at known results about the communication complexity of some popular functions. Let $\mathrm{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{EQ}_n(a,b) = 1$ iff the vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ are equal. It is well-known and easy to prove that $C(\mathrm{EQ}_n) = n$. Obviously the same results can be achieved if Alice gets exactly one of the variables a_i and b_i, $1 \leq i \leq n$. Similar results can be obtained for the functions $\mathrm{GT}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and $\overline{\mathrm{GT}_n} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $\mathrm{GT}_n(a,b) = 1$ iff $[a]_1^n > [b]_1^n$ and $\overline{\mathrm{GT}_n}(a,b) = 1$ iff $[a]_1^n \leq [b]_1^n$.

Now, we are ready to define the function $f_n \in B_{3n}$ on the variables $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_n)$, and $c = (c_1, \ldots, c_n)$:

$$f_n(a,b,c) := (\mathrm{EQ}_n(a,\overline{c}) \wedge \overline{\mathrm{GT}_n}(a,b)) \vee \mathrm{GT}_n(a,\overline{c}).$$

Using case inspection on the distribution of the c-variables it is not difficult to prove that for a partition, where the a- and b-variables are separated, there exists a strong fooling set of size 2^n for f_n. In other words the communication complexity of f_n is not smaller than the communication complexity of $\overline{\mathrm{GT}_n}$ and the distribution of the c-variables does not simplify the task. The same result can be obtained if Alice gets exactly one of the variables a_i and b_i for all $i \in \{1, \ldots, n\}$. In this case it is not important whether the investigated c-variables belong to Alice or Bob but whether the considered a- and c-variables or b- and c-variables are tested together.

3 The middle bit of integer multiplication

In this section we present some results on the OBDD size of the middle bit of integer multiplication. Furthermore, we investigate more general BDD models.

3.1 On the OBDD size of the middle bit of integer multiplication

Bryant's lower bound of $2^{n/8}$ on the OBDD size of $\mathrm{MUL}_{n-1,n}$ is unsatisfactory since it does not rule out the possibility that 64-bit multipliers can be represented by OBDDs containing only 256 nodes. Since the aim is to use

OBDDs for realistic applications one is interested in small constructions or a better lower bound. Introducing a new technique based on universal hashing Woelfel [35] has improved the lower bound considerably to $2^{\lfloor n/2 \rfloor}/61-4$. This result implies that any OBDD for 64-bit multiplication needs more than 70 million nodes and the verification of 128-bit multipliers is infeasible because more than $3 \cdot 10^{17}$ OBDD-nodes are necessary.

The main proof idea in Bryant's and Woelfel's lower bound proofs is to show that for every variable order π there exists an integer $c \in \{1, \ldots, 2^n - 1\}$ such that the π-OBDD size of $\mathrm{MUL}^c_{n-1,n}$ is exponential. Bryant has chosen c in such a way that only two input bits of c are set to 1. Therefore, the product of x and y can be seen as the sum of two integers obtained by shifting x in an appropriate way. More precisely, if y is replaced by the binary representation of c and $c = 2^i + 2^{i+d}$ then $c \cdot x = x \cdot 2^i + x \cdot 2^{i+d}$. Woelfel has enlarged the possible choices for the integer c. As a result he has been able to prove that for every variable order π there exists an integer c such that $\mathrm{MUL}^c_{n-1,n}$ has a large number of subfunctions obtained by replacements of the first $n/2$ x-variables in π by constants. Summarizing Bryant's and Woelfel's lower bound proofs rely only on the existence of a constant factor c for each variable order π for which $\mathrm{MUL}^c_{n-1,n}$ leads to a large π-OBDD representation. If one would like to improve the lower bound there are two possibilities. The first one is to consider multiple values for c, the second one to improve the lower bound for the π-OBDD size of $\mathrm{MUL}^c_{n-1,n}$ for an suitably chosen constant c. Woelfel has shown that the latter approach cannot yield significant better lower bounds because the variable order $\pi = (x_0, x_1, \ldots, x_{n-1})$ leads to OBDDs of size at most $3 \cdot 2^{n/2}$ for each integer c. By combining this result with the observation that the k most significant bits of one input vector are not important any more if the k least significant bits of the other input vector are known, Woelfel has obtained the first non-trivial upper bound of $(7/3) \cdot 2^{(4/3)n}$ on the size of OBDDs for $\mathrm{MUL}_{n-1,n}$ with respect to the variable order $\pi = (y_0, \ldots, y_{n-1}, x_0, \ldots, x_{n-1})$. Amano and Maruoka [3] have improved this upper bound to $2.8 \cdot 2^{(6/5)n}$ for so-called quasi-reduced or complete OBDDs, i.e., OBDDs where on each path from the source to the sinks all variables have to be tested, and the pairwise ascending variable order $\pi = (x_0, y_0, \ldots, x_{n-1}, y_{n-1})$. (It is not difficult to see that the size of a quasi-reduced OBDD can be at most $n + 1$ times larger than the size of a reduced OBDD for a given function f with respect to the same variable order.) Despite the considerable amount of research dealing with the complexity of the middle bit of multiplication, the gap between lower and upper bounds on its OBDD size is still large. Furthermore, even Woelfel's improved lower bound does not really justify why OBDDs for multipliers of input length $n = 64$ cannot be constructed nowadays using current standard

PC hardware. Sauerhoff [26] has shown that the upper bound of Amano and Maruoka [3] is in fact asymptotically optimal for the order chosen by them which is believed to be one of the best ones. For $n = 64$ his bound is larger than $1.62 \cdot 10^{21}$. This surely explains why an OBDD with respect to this variable order cannot be generated. Nevertheless, there is the possibility that there are considerably better variable orders.

3.2 On the size of more general BDD models for the middle bit of integer multiplication

In learning theory and genetic programming OBDDs are used to represent approximations of Boolean functions. Gronemeier [20] has shown that for every variable order π the approximation of some output bits of integer multiplication with respect to the uniform distribution and constant error requires π-OBDDs of exponential size. Nevertheless, approximating the middle bit of integer multiplication with polynomially small error is easy even for read-once branching programs [27].

Although there has been considerable progress in the development of lower bound proofs by the investigation of weakly restricted BDD models, the lower bound methods often only work for a quite limited class of functions. Besides the interest in finding lower bounds as large as possible or proving superpolynomial lower bounds for more and more general BDD models, it is important to apply the existing methods to (practically) important functions. Lower bound proofs for such functions may help to develop new or refined proof techniques, or can lead to new insights into the properties of the considered functions. This is the motivation for the further investigation of the complexity of integer multiplication for more general BDD models.

Methods from communication complexity have been used to prove large lower bounds in several binary decision diagram models. Bryant [12] has used the *fooling set* method to obtain lower bounds on the communication complexity of the middle bit of multiplication which implies an exponential lower bound of size $2^{n/8}$ for OBDDs representing $\mathrm{MUL}_{n-1,n}$. Incorporating Ramsey theoretic arguments of Alon and Maass [2] and using the *rank method* of communication complexity Gergov [19] has extended Bryant's lower bound to arbitrary nondeterministic linear-length oblivious BPs. His lower bound is still non-polynomial for length $o(n \log n / \log \log n)$. Since Woelfel's larger lower bound on the OBDD size of $\mathrm{MUL}_{n-1,n}$ has not been proved using strong fooling sets his result cannot generalized in the same way as Bryant's to nondeterministic linear-length oblivious branching programs. In [1] Gergov's reduction has been applied to deduce that also randomized OBDDs require

exponential size and it has been shown that in contrast the graph of integer multiplication MUL-Graph$_n$ has randomized OBDDs of polynomial size. For the later result the fact has been used that it is easy to verify with small error probability whether the product of two integers equals some given output applying arithmetic modulo a random chosen prime. For non-oblivious models Ponzio [25] has presented the first (weakly) exponential lower bound. He has shown that the complexity of the middle bit of integer multiplication is $2^{\Omega(n^{1/2})}$ for read-once branching programs. In [6] the first exponential lower bound on the size of a nondeterministic non-oblivious read-once branching program model, namely for nondeterministic tree-driven BP1s, has been presented. An extension of the proof shows that all subfunctions of MUL$_{n-1,n}$ obtained by the replacement of up to $(n/\log n)^{1/2-\epsilon}$ variables, $\epsilon > 0$ any constant, have exponential size for nondeterministic OBDDs. Since the result also holds for the parity-acceptance mode, where the function value equals 1 for an input iff the number of its accepting paths is odd, this has been the first non-trivial lower bound for an important function on non-oblivious restricted BP1s with an unlimited number of parity nodes.

The fact that integer multiplication defines a universal hash class [14, 15, 36], called multiplicative hash class, has also been used by Bollig and Woelfel [9] to improve the exponential lower bound on the size of BP1s up to $2^{\lfloor \frac{n-9}{4} \rfloor}$ which is even larger than Bryant's lower bound on the OBDD size. Moreover, the analysis seems to be much easier than the counting technique used by Ponzio. At the beginning one reason for the difficulties in proving exponential lower bounds on the size of binary decision diagram models representing MUL$_{n-1,n}$ could have been arisen from the fact that integer multiplication can express many different shifting and adding combinations such that the effect of partial assignments and therefore the subfunctions are not easy to analyze. Using methods which rely on universal hashing it has been shown that even if almost a quarter of the variables of each factor has been replaced by constants, each result of the product bits between the positions $n-1$ and $(3/4)n$ (the results for MUL$_{n-1,n}$ to MUL$_{(3/4)n,n}$) is still possible. Using an algebraic approach in [8] a lower bound of $2^{\lfloor (n-46)/12 \rfloor} \cdot n^{-1}$ for a restricted nondeterministic BP1 model with parity acceptance mode, called parity graph-driven BP1s, has been shown. This result has been motivated by the fact that until now no superpolynomial lower bound on the size of unrestricted nondeterministic BP1s with parity acceptance mode for an explicitly defined Boolean function has been known. Since exponential lower bounds on the size of unrestricted nondeterministic read-once branching programs which represent MUL$_{n-1,n}$ had been unknown, one step towards proving such bounds was to investigate BP models "inbetween" deterministic

and nondeterministic BP1s and a model where some but not all variables may be tested multiple times [10, 37]. Finally, Sauerhoff and Woelfel [27] have achieved a major breakthrough presenting exponential lower bounds on the size of nondeterministic and randomized BPks for $\mathrm{MUL}_{n-1,n}$.

Wegener and Woelfel [34] have considered unrestricted branching programs and Boolean formulas over the basis B_2 of all binary operations. Since more than 40 years the best lower bounds for explicitly defined functions are for general branching programs of order $n^2/\log^2 n$ and for Boolean formulas of order $n^2/\log n$. These results have been proved with Nechiporuk's technique [24]. It is well known that this method cannot yield better lower bounds. In [34] the following results have been presented. Any branching program for $\mathrm{MUL}_{n-1,n}$ has at least $\Omega(n^{3/2}/\log n)$ nodes and any Boolean formula for $\mathrm{MUL}_{n-1,n}$ has size at least $\Omega(n^{3/2})$. Furthermore, it has been proved that using Nechiporuk's technique it is impossible to prove better lower bounds than $\Omega(n^{5/3}/\log n)$ and $\Omega(n^{5/3})$ for the branching program and Boolean formula size of $\mathrm{MUL}_{n-1,n}$. These are non-trivial limits of Nechiporuk's technique. Until now it is still an open question whether the lower bound method has been applied in the best possible way in [34].

4 The most significant bit of integer multiplication

Although many exponential lower bounds on the OBDD size of Boolean functions are known and the lower bound methods are simple, it is often a more difficult task to prove large lower bounds for some predefined and interesting functions. The most significant bit of integer multiplication is a good example. Despite the well-known lower bounds on the OBDD size of the so-called middle bit of multiplication ([12], [35]), only recently it has been shown that the OBDD complexity of the most significant bit is also exponential [5] answering an open question posed by Wegener [32]. Here, we start our investigation by an improved lower bound on the size of π-OBDDs for $\mathrm{MUL}_{2n-1,n}$, where π is a fixed variable order. Using communication complexity the proof is very simple and elementary. Afterwards we present the best general lower bound on the OBDD size for $\mathrm{MUL}_{2n-1,n}$ known so far and conclude our considerations by the best known upper bound.

4.1 Lower bounds on the OBDD size of the most significant bit of integer multiplication

In this section we start our investigation of lower bounds on the size of OBDDs for $\mathrm{MUL}_{2n-1,n}$ by presenting a lower bound for some fixed variable order. Afterwards we present a general lower bound which is much smaller but also exponential. The ideas of the general lower bound have been presented in [4].

Using techniques from analytical number theory Sawitzki [29] has presented a lower bound of $2^{n/6}$ on the size of π-OBDDs representing the most significant bit of integer multiplication for the variable order π where the variables are tested according to increasing significance, i.e., $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. Here, we prove a larger lower bound in an easier way and without analytical number theory.

Theorem 4.1. *Let $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. The π-OBDD size for the representation of $\mathrm{MUL}_{2n-1,n}$ is $\Omega(2^{n/4})$.*

Proof. We start with the following two useful observations. For a number $2^{n-1} + \ell 2^{n/2}$ the corresponding smallest number such that the product of the two numbers is at least 2^{2n-1} is $2^n - \ell 2^{n/2+1} + 4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$. Furthermore,

$$2^{n/2} > 4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor > 4(\ell-1)^2$$

for $0 < \ell \leq 2^{n/4-1}$. Using these two facts it is not difficult to construct a strong fooling set of size $2^{n/4-1}$:

Let $X_U := \{x_{n-1}, x_{n-2}, \ldots, x_{n/2}\}$, $Y_U := \{y_{n-1}, y_{n-2}, \ldots, y_{n/2}\}$, $X_L := \{x_{n/2-1}, x_{n/2-2}, \ldots, x_0\}$, and $Y_L := \{y_{n/2-1}, y_{n/2-2}, \ldots, y_0\}$. We define $Z_A := X_U \cup Y_U$ and $Z_B := X_L \cup Y_L$. The Set S contains all pairs (a, b) for $\ell \in \{1, 2, \ldots, 2^{n/4-1}\}$ with the following properties:

1. a is an assignment that consists of a partial assignment a_x to the variables in X_U and a partial assignment a_y to the Y_U-variables where $\|a_x\| = 2^{n-1} + \ell 2^{n/2}$ and $\|a_y\| = 2^n - \ell 2^{n/2+1}$ and

2. b is an assignment that consists of a partial assignment b_x to the variables in X_L and a partial assignment b_y to the Y_L-variables where $\|b_x\| = 0$ and $\|b_y\| = 4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$.

For all pairs in S the function value of $\mathrm{MUL}_{2n-1,n}$ is 1. Let (a_1, b_1) and (a_2, b_2) be two different pairs in S. If the value of the partial assignment of

the X_U-variables according to a_1 is $2^{n-1}+\ell_1 2^{n/2}$ and the value of the partial assignment of the X_U-variables according to a_2 is $2^{n-1}+\ell_2 2^{n/2}$, where w.l.o.g. $\ell_1 < \ell_2$, the function value of $\mathrm{MUL}_{2n-1,n}(a_2, b_1)$ is 0. Therefore, S is a fooling set of size $2^{n/4-1}$. □

Because of the symmetric definition of strong fooling sets we also obtain a lower bound of $2^{n/4-1}$ on the size of π'-OBDDs for the most significant bit, where $\pi' = (x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0)$.

Now, we prove the general lower bound.

Theorem 4.2. *The* OBDD *size for the representation of* $\mathrm{MUL}_{2n-1,n}$ *is* $\Omega(2^{n/60})$.

Proof. We start with a (simplified) presentation of the main proof ideas for a lower bound of $\Omega(2^{n/96})$ and present afterwards the idea how to improve this lower bound up to $\Omega(2^{n/60})$.

Our aim is to show for an arbitrary variable order π that a π-OBDD for $\mathrm{MUL}_{2n-1,n}$ contains a π-OBDD for the Boolean function $f_{n'}$ defined in Section 2.3:

$$f_{n'}(a,b,c) = (\mathrm{EQ}_{n'}(a,\overline{c}) \wedge \overline{\mathrm{GT}_{n'}}(a,b)) \vee \mathrm{GT}_{n'}(a,\overline{c}),$$

where for each position i the variables a_i and b_i are suitably separated in π and $n' = \Theta(n)$. Therefore, the size of the π-OBDD for $\mathrm{MUL}_{2n-1,n}$ has to be large. The vector a is a subvector of one of the inputs x and y for $\mathrm{MUL}_{2n-1,n}$, the vectors b and c of the other input.

We use the idea of the following reduction from multiplication to squaring presented by Wegener [33], where squaring computes the square of an m-bit input. For two m-bit numbers u and w the number $\ell := u \cdot 2^{2(m+1)} + w$ is defined. Then

$$\ell^2 = u^2 \cdot 2^{4(m+1)} + uw2^{2(m+1)+1} + w^2.$$

Since w^2 and uw are numbers of length $2m$, the binary representation of the product uw can be found in the binary representation of ℓ^2. (Figure 1 shows the bit composition of the number ℓ^2.)

A key observation is the following one. The number $\left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$ is smaller than ℓ if $\ell \leq 2^{n/4-3/2}$. As a consequence if b_ℓ is the binary representation of ℓ, b_{ℓ^2} is the binary representation of ℓ^2, L the length of b_ℓ, and if there exists j, where $j \geq L-2$, and $[b_{\ell^2}]_j = 1$, there is no difference in the upper half of the binary representations of the numbers $4\ell^2$ and $4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$.

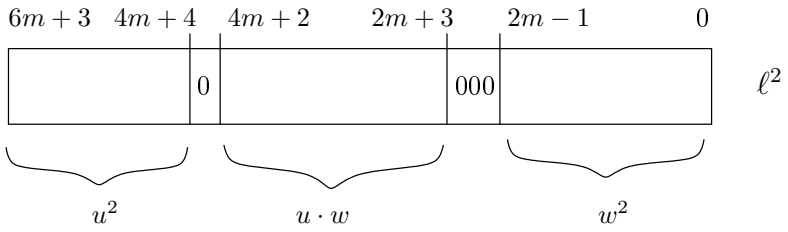

Figure 1: The bit composition of the number ℓ^2

More precisely, if b' is the binary representation of $4\ell^2$ and b'' is the binary representation of $4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$, then $[b']_{j+1}^{2L+1} = [b'']_{j+1}^{2L+1}$.

Next, we investigate requirements that have to be fulfilled for inputs x and y, where $\mathrm{MUL}_{2n-1,n}(x,y) = 1$. If x represents a number $2^{n-1} + \ell 2^{n/2}$, $1 \leq \ell \leq 2^{n/4-3/2}$, the upper half of y has to represent a number of at least $2^{n/2} - 2\ell$, i.e., $[y]_{n/2}^{n-1} \geq 2^{n/2} - 2\ell$. If the upper half of y represents a number greater than $2^{n/2} - 2\ell$, the function value $\mathrm{MUL}_{2n-1,n}(x,y)$ is 1. Let j be the minimum integer in the set $\{i \mid n/2 \leq i < (3/4)n - 3/2 \text{ and } x_i = 1\}$. If $[\overline{y}]_{j+2}^{n-1} > [x]_{j+1}^{n-2}$, the function value $\mathrm{MUL}_{2n-1,n}$ is 0. If $[\overline{y}]_{j+2}^{n-1} < [x]_{j+1}^{n-2}$, the function value $\mathrm{MUL}_{2n-1,n}$ is 1. If $y_{j+1} = 1$, $[\overline{y}]_{j+2}^{n-1} = [x]_{j+1}^{n-2}$, and $[y]_{n/2}^{j} = 0$, $[y]_0^{n/2-1}$ has to represent a number of at least $4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$.

In order to use Wegener's observation on squaring mentioned above we only consider integers ℓ where $\ell = u2^{2(m+1)} + w$, $u, w < 2^m$ and $m = n/12 - 5/6$. (Later on we show that m can be enlarged which leads to a larger lower bound.) For this reason we replace the variables $x_{n/2+m}, \ldots, x_{n/2+2m+1}$ by 0. (See Figure 2 for the composition of the number x.) Afterwards we replace some of the x-variables and the corresponding y-variables by constants, where y_{i+1} is the corresponding y-variable to x_i, such that a certain part of uw is equal to a certain part of $2^d \cdot w$ for d suitably chosen. Furthermore, we choose w in such a way that the assignments to the variables at position $3m + 5, \ldots, 6m + 5$ are the same in the binary representations of $4\ell^2$ and $4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$. Moreover, for different integers ℓ_1 and ℓ_2 (which means different assignments to the w-variables) the assignments to the variables at position $3m+5, \ldots, (7/2)m+4$ in the binary representations of $4\ell_1^2$ and $4\ell_2^2$ are different. (Figure 3 illustrates some of the replacements of the y-variables.)

Now we make our proof idea more precise. We rename $[x]_{n/2}^{n/2+(n/12)-11/6}$ by $[w]_0^{m-1}$ and $[x]_{n/2+n/6+1/3}^{n/2+n/4-3/2}$ by $[u]_0^{m-1}$. If $\ell = u \cdot 2^{2(m+1)} + w$ the product $u \cdot w$ can be found at position $2m+5, \ldots, 4m+4$ in the binary representation

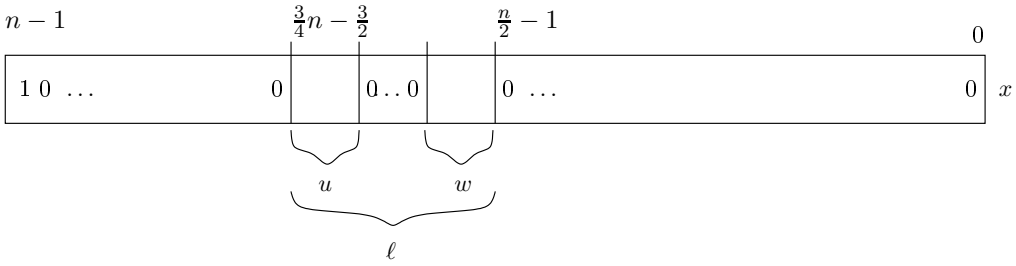

Figure 2: The composition of the input x

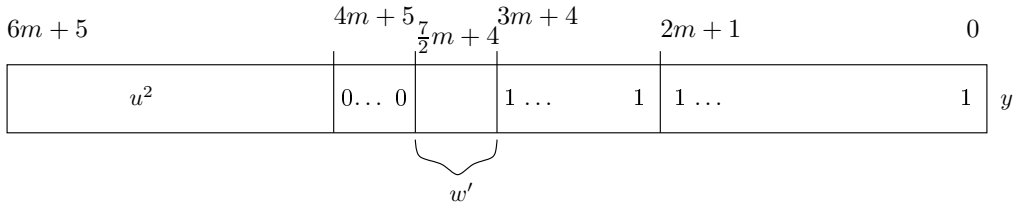

Figure 3: The effect of the replacements of some of the y-variables, where $u = [u]_0^{m-1}$ (w' has to be at least $(2^d \cdot w)_m^{(3/2)m-1}$)

of $4\ell^2$. The crucial step is to choose an appropriate subset of the input variables in order to show that there exists a large strong fooling set. Let $S := \{w_{m/2}, \ldots, w_{m-1}, y_{3m+5}, \ldots, y_{(7/2)m+4}\}$ and T be the set of the first $|T|$ variables according to π, where there are $m/2$ variables from S, and B be the set of the remaining variables. Let $W_{S,T}$ be the w-variables in $S \cap T$, $W_{S,B}$ the w-variables in $S \cap B$. Similar the sets $Y_{S,T}$ and $Y_{S,B}$ are defined. Using simple counting arguments we can prove that there exists a distance parameter d such that there are at least $m/8$ pairs $(w_i, y_{2m+5+i+d})$ in $W_{S,T} \times Y_{S,B} \cup W_{S,B} \times Y_{S,T}$ (for a similar proof see, e.g., [12]). Let I be the set of indices, where w_i belongs to such a pair. We replace the u-variables such that $[u]_0^{m-1} = 2^d$ and the variables $y_{4m+6}, \ldots, y_{6m+5}$ such that $[y]_{4m+6}^{6m+5} = 2^{2d}$.

The variables $x_{n/2+i}$, $i \in I$, are called free x-variables, the variables $y_{n/2+i+1}$ and $y_{2m+5+i+d}$, $i \in I$, free y-variables. The free x-variables will play the role of the a-variables, the free variables $y_{n/2+i+1}$, $i \in I$, the role of the c-, and the remaining free y-variables the role of the b-variables in the reduction from the function $f_{n'}$ mentioned above to $\mathrm{MUL}_{2n-1,n}$. Now we present the reduction. (Figure 4 shows some of the replacements to the inputs x and y of $\mathrm{MUL}_{2n-1,n}$.)

- The variables y_{n-1} and x_{n-1} are set to 1,

- $x_{n/2+m-d-1}$ (which corresponds to w_{m-d-1}) and $y_{n/2+m-d}$ are set to 1,

- $x_{n/2+2m+d}$ (which corresponds to u_d) is set to 1, the corresponding variable $y_{n/2+2(m+1)+d+1}$ is set to 0, $y_{4m+6+2d}$ to 1, the variables $y_{(7/2)m+5}, \ldots, y_{4m+5+2d}$ and $y_{4m+7+2d}, \ldots, y_{6m+5}$ to 0 (as a result $[y]_{4m+6}^{6m+5} = 2^{2d}$).

- The variables $y_{n/2}, \ldots, y_{n/2+m-d-1}$ are set to 0.

- Besides the free x-variables the remaining x-variables are replaced by 0.

- Besides the free y-variables the remaining y-variables are replaced by 1.

What is the effect of the replacements?

- The inputs x and y represent numbers that are at least 2^{n-1}, since otherwise the function value $\mathrm{MUL}_{2n-1,n}(x,y)$ is 0.

- Since $w_{m-d-1} = 1$ and $[u]_0^{m-1} = 2^d$, $4\ell^2$ and $4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2-1}+\ell} \right\rfloor$, where $\ell = u \cdot 2^{2(m+1)} + w$, do not differ in one of the bits at position $3m + 5, \ldots, 6m + 5$ of their binary representations.

- Since $x_{n/2+m-d-1} = 1$ and $y_{n/2+m-d} = 1$, $x_{n/2} = \ldots = x_{n/2+m-d-2} = 0$ and $y_{n/2} = \ldots = y_{n/2+m-d-1} = 0$, $[x]_{n/2+m}^{n-2} = [\overline{y}]_{n/2+m+1}^{n-1}$, $[x]_{n/2+m-d}^{n/2+m-1}$ has to be at least $[\overline{y}]_{n/2+m-d+1}^{n/2+m}$ for inputs x and y, where $\mathrm{MUL}_{2n-1,n}(x,y) = 1$. If $[x]_{n/2+m-d}^{n/2+m-1} > [\overline{y}]_{n/2+m-d+1}^{n/2+m}$, $\mathrm{MUL}_{2n-1,n}(x,y) = 1$.

- Since $[y]_{4m+6}^{6m+5} = 2^{2d} = u^2$ and because of the other replacements, $[y]_{3m+5}^{4m+4}$ has to be at least $(u \cdot w)$ div 2^m for inputs x and y, where $\mathrm{MUL}_{2n-1,n}(x,y) = 1$, if $[y]_{n/2}^{n-1} = 2^{n/2} - 2\ell$ and $[x]_{n/2}^{n-1} = 2^{n/2-1} + \ell$.

Therefore, the correctness of our reduction follows from our considerations above. Considering the fact that $m = n/12 - 5/6$, we get the result that the OBDD complexity of $\mathrm{MUL}_{2n-1,n}$ is at least $\Omega(2^{n/96})$.

Finally, we present the idea how to improve the lower bound on the OBDD complexity of $\mathrm{MUL}_{2n-1,n}$ up to $\Omega(2^{n/60})$. Up to now we have considered numbers ℓ, where $\ell = u \cdot 2^{2(m+1)} + w$ and $u, w < 2^m$ with $m = (n/12) - 5/6$. Using the fact that in our lower bound proof only the upper half of the bits in the binary representation of uw is important, uw div $2^{(3/2)m} = 0$, u^2 div $2^{(7/4)m} = 0$, and $u^2 \mod 2^{m/4} = 0$, we can choose $\ell = u \cdot 2^m + w$, $w < 2^m$ and $u < 2^{(7/8)m}$. As a result we can enlarge m up to $(2/15)n$.

□

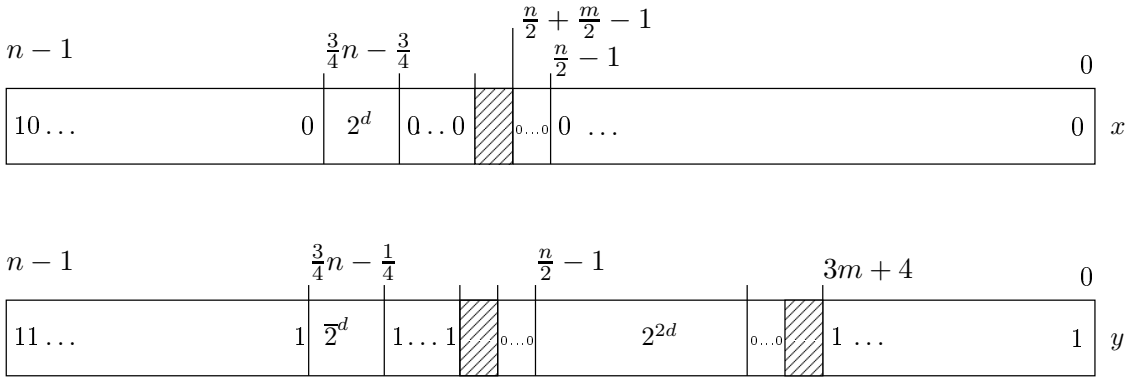

Figure 4: A (simplified) presentation of replacements to some of the x- and y-variables. The shaded areas contain the free variables (and possibly other variables)

4.2 Upper bounds on the OBDD size of the most significant bit of integer multiplication

In this section we prove an upper bound on the size of OBDDs according to an arbitrary variable order representing the most significant bit of integer multiplication. Afterwards we present the best known upper bound on the size of OBDDs representing $\mathrm{MUL}_{2n-1,n}$ according to the variable order $\pi = (x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0)$. The results of this section have been presented in [7].

We start our proof of the general upper bound with the investigation of a function f that is closely related to the most significant bit of integer multiplication. Let $n \in \mathbb{N}$ be arbitrary but fixed in the rest of the section.

Lemma 4.3. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be defined as $f(x) := \frac{2^{2n-1}}{x}$. For arbitrary $\Delta x, \Delta y > 0$ there exists exactly one value $x \in \mathbb{R}^+$ with $f(x) - f(x+\Delta x) = \Delta y$.*

In other words each distance pair $(\Delta x, \Delta y)$ defines uniquely two elements in the definition set.

Next, we consider some modifications of the function f.

Definition 4.4. For $c, d \in \mathbb{R}$ and $n \in \mathbb{N}$ we define the function $f_{c,d} : \mathbb{R} \to \mathbb{R}$ in the following way.

$$f_{c,d}(x) := \frac{2^{2n-1}}{c+x} - d.$$

The function $f_{c,d}$ contains the tuple (x, y) iff $f_{c,d}(x) = y$.

Our proof idea of the upper bound on the size of OBDDs representing $\mathrm{MUL}_{2n-1,n}$ is to use the functions $f_{c,d}$ in order to analyze the number of

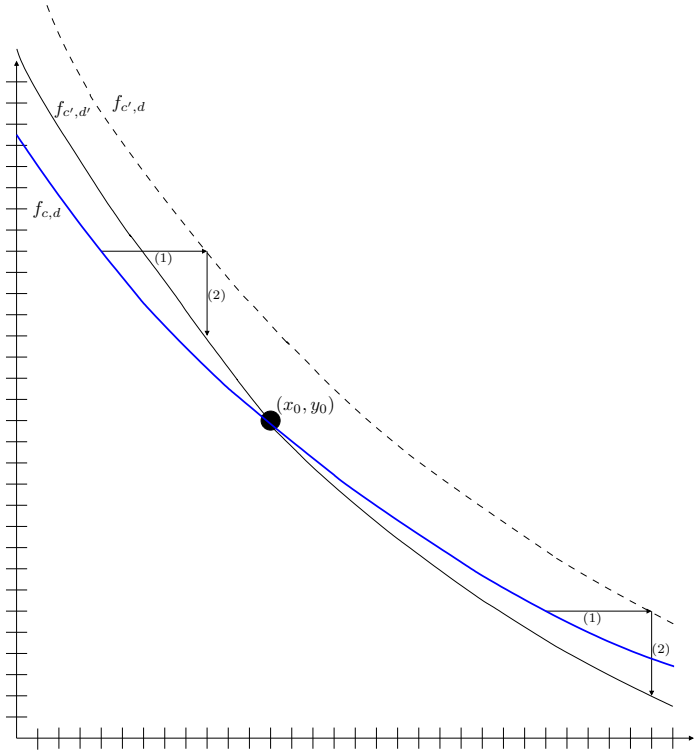

Figure 5: Rotation of the graph of the function $f_{c,d}$

different subfunctions of $\mathrm{MUL}_{2n-1,n}$ obtained by replacements of some x- and y-variables by constants. For this reason we have to relate the functions $f_{c,d}$ to subfunctions of $\mathrm{MUL}_{2n-1,n}$.

Definition 4.5. For a given function $f_{c,d}$ and two arbitrary finite sets $A, B \subseteq \mathbb{N}$ and $A, B \neq \emptyset$ the corresponding step function $f_{c,d}^{A,B} : A \to B$ is defined as

$$f_{c,d}^{A,B}(x) := \min\{y \in B | y \geq f_{c,d}(x)\}.$$

$\mathrm{MUL}_{2n-1,n}$ can be uniquely described by $f_{0,0}^{A,B}$, where $A, B = \{0, \ldots, 2^n - 1\}$. Obviously there can be several functions $f_{c,d}$ that lead to the same step function. It is easy to see that each function $f_{c,d}$ can be characterized by two tuples (x_1, y_1) and (x_2, y_2), where $f_{c,d}(x_i) = y_i$ and $x_i, y_i \in \mathbb{R}$ for $i \in \{1, 2\}$. Unfortunately, the length of the numbers could be large. In order to find a small representation for $f_{c,d}$ we modify $f_{c,d}$ without changing essentially the corresponding step function.

We start to analyze the effect of moderate modifications of the parameters c and d.

Lemma 4.6. *Let $c, d \in \mathbb{R}^+$ and A, B be two arbitrary finite, nonempty subsets of $\mathbb{N}$. Let y_x be the largest element in B that is smaller than $f_{c,d}(x)$ and $\epsilon_x := f_{c,d}^{-1}(y_x) - x$, if y_x is defined, otherwise $\epsilon_x := \infty$. We define $\epsilon_{min} := \min\{\epsilon_x | x \in A\}$. Then $f_{c,d}^{A,B} = f_{c+\epsilon_{min}/2,d}^{A,B}$.*

Lemma 4.7. *Let $c, d \in \mathbb{R}^+$ and A, B be two arbitrary finite, nonempty subsets of $\mathbb{N}$. For $x \in A$ let $\epsilon_x := f_{c,d}^{A,B}(x) - f_{c,d}(x)$ if $f_{c,d}^{A,B}(x)$ is defined and ∞ otherwise. We define $\epsilon_{min} := \min\{\epsilon_x | x \in A\}$. Then $f_{c,d-\epsilon_{min}}^{A,B} = f_{c,d}^{A,B}$.*

Lemma 4.7 tells us that it is allowed to move the graph of the function $f_{c,d}$ upwards, right until it hits its corresponding step function for the first time, without changing the step function.

Lemma 4.8. *Let $c, d \in \mathbb{R}^+$ and A, B be two arbitrary finite, nonempty subsets of $\mathbb{N}$, such that there exists at least one element $x_0 \in A$ where $f_{c,d}(x_0) = f_{c,d}^{A,B}(x_0)$, and there are at least two elements $x_1, x_2 \in A$ where $\min\{y | y \in B\} < f_{c,d}(x_i) \leq \max\{y | y \in B\}$, $i \in \{1, 2\}$. We define the following rotation operation for $f_{c,d}$ with respect to (x_0, y_0): decrease c continuously to c' and adjust d to d' at the same time such that $f_{c,d}(x_0) = f_{c',d'}(x_0)$ is always fulfilled until there exists another element $x' \in A$ with $f_{c',d'}(x') \in B$.*

1. *The rotation operation is finite.*
2. *The function $f_{c,d}^{A,B}$ can be reconstructed from $f_{c',d'}$ in the following way:*

$$f_{c,d}^{A,B}(x) = \begin{cases} \min\{y \in B | y \geq f_{c',d'}(x)\}, & \text{if } x \leq x_0, \\ \min\{y \in B | y > f_{c',d'}(x)\}, & \text{if } x > x_0. \end{cases}$$

If we replace (c, d) to (c', d'), the curve of the function $f_{c,d}$ seems visually to rotate to the graph of the function $f_{c',d'}$, because point (x_0, y_0) stays on the graph, whereas all points left of x_0 are shifted upwards and the other ones downwards. Nevertheless, the graph's shape does not change since the rotation can be decomposed to a vertical and a horizontal movement (see Figure 5). Therefore, it is still possible to use Lemma 4.3 for the identification of $f_{c',d'}$.

Now we are able to prove our general upper bound on the π-OBDD size for $\mathrm{MUL}_{2n-1,n}$.

Theorem 4.9. *Let π be an arbitrary variable order. The π-OBDD size for the representation of $\mathrm{MUL}_{2n-1,n}$ is $O(2^{(4/3)n})$.*

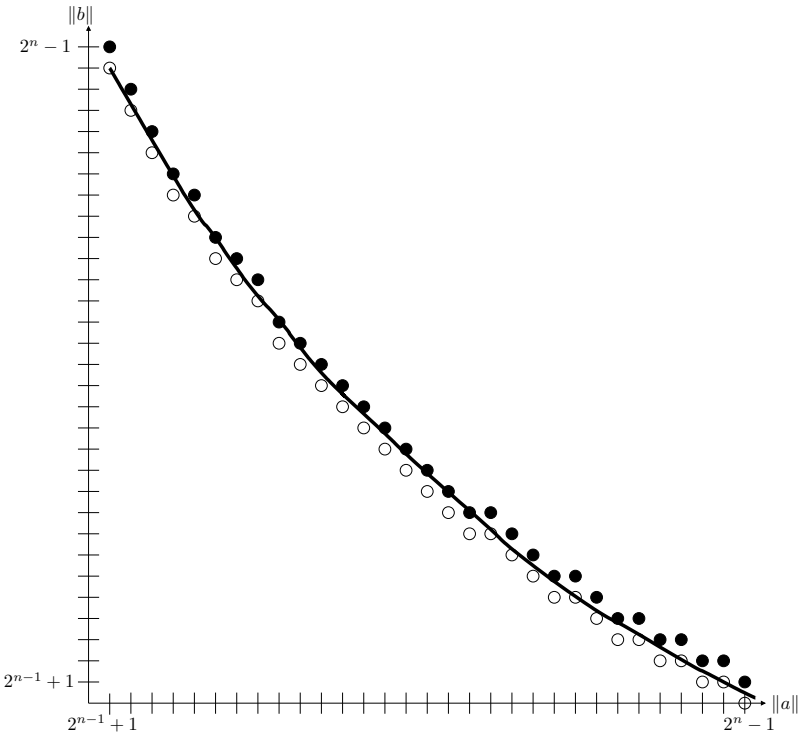

Figure 6: Significant points for the evaluation of $\mathrm{MUL}_{2n-1,n}$

Proof. Our aim is to prove an upper bound of $2^{2(n-i)+2(n-j)+2}$ on the number of subfunctions of $\mathrm{MUL}_{2n-1,n}$ obtained by replacements of i x- and j y-variables by constants.

We assume in the following that $i \neq 0$ and $j \neq 0$, since otherwise we are done. If $i = n$ an upper bound of 2^{n-j} is easy to prove and we are done, similarly an upper bound of 2^{n-i} can be shown for $j = n$. Therefore, we also assume in the following that $i \neq n$ and $j \neq n$. Let X_S be the set of i arbitrary x-variables and Y_S be the set of j arbitrary y-variables, $X_T := \{x_0, \ldots x_{n-1}\} \backslash X_S$, and $Y_T := \{y_0, \ldots, y_{n-1}\} \backslash Y_S$.

$\mathrm{MUL}_{2n-1,n}$ answers the question, whether for a given assignment (a, b) of the variables, the product $\|a\| \cdot \|b\|$ is at least 2^{2n-1}. Therefore, the function $\mathrm{MUL}_{2n-1,n}$ can be described by specifying for every possible assignment a of the x-variables, the assignment b of the y-variables with $\|b\| = \left\lceil \frac{2^{2n-1}}{\|a\|} \right\rceil$. Figure 6 shows $\mathrm{MUL}_{2n-1,n}$, where for a value $\|a\|$ the smallest corresponding value $\|b\|$ that fulfills $\mathrm{MUL}_{2n-1,n}$ is dotted. Such pairs of assignments are called significant points. (For sake of simplicity the possible values are at least 2^{n-1} because for smaller numbers the product cannot be at least 2^{2n-1}.)

Let $c := \|a_{X_S}\|$ and $d := \|b_{Y_S}\|$. We define A_T as the set of possible

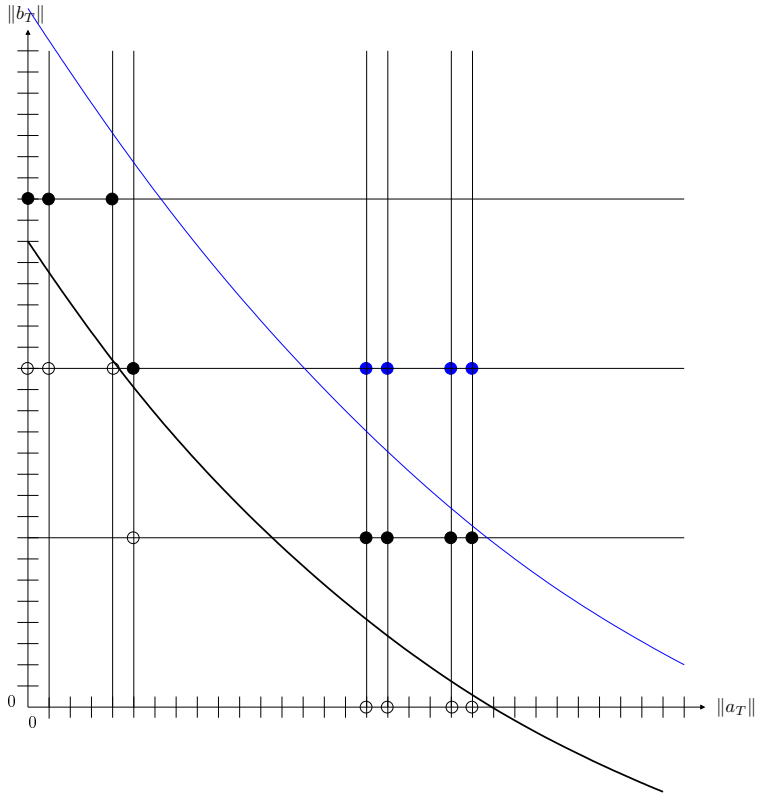

Figure 7: Two different step functions

values $\|a_{X_T}\|$ that can be expressed by the variables from X_T. Let B_T be defined in the same way. A_T and B_T are independent of the choice of c and d, i.e., a grid can be defined for the $\|a_{X_T}\|$- and $\|b_{X_T}\|$-values, which has the same appearance for all possible assignments c and d. A subfunction of $\mathrm{MUL}_{2n-1,n}$ obtained by replacing the variables in X_S to a_S and Y_S to b_S can be described by the pairs of A_T- and B_T-values $(\|a_{X_T}\|, \|b_{Y_T}\|)$, so that $\|b_{Y_T}\|$ is the minimal value that fulfills $\|b_{Y_T}\| \geq \frac{2^{2n-1}}{c+\|a_{X_T}\|} - d$. Therefore, the subfunction of $\mathrm{MUL}_{2n-1,n}$ can be characterized by the step function $f_{c,d}^{A_T,B_T}$ (see Definition 4.5) for the underlying function $f_{c,d}$.

Figure 7 shows an example for two different step functions that result from two different assignments to the variables in $X_S \cup Y_S$.

Since the subfunctions obtained by replacing the variables in X_S and Y_S by constants can uniquely be described by their step functions, our aim is to prove the existence of a small representation such that the corresponding step function and therefore the corresponding subfunction of $\mathrm{MUL}_{2n-1,n}$ can be reconstructed later on. As each representation implicates at most one

possible step function, the number of different representations is an upper bound on the number of different subfunctions.

The idea is to transform the function $f_{c,d}$ in a moderate way into a function $f_{c',d'}$, such that $f_{c',d'}$ contains at least two points from $A_T \times B_T$ and the step function $f_{c,d}^{A_T,B_T}$ can easily be obtained from $f_{c',d'}$. In the following we assume that for at least two A_T-values, the function value $f_{c,d}$ is greater than 0 and smaller or equal to the greatest value in B_T. The other cases will be considered later on. If c equals 0, we have to make some extra considerations. Since the function $f_{c,d}$ is not defined for the value $\|a_{X_T}\| = 0$, we use Lemma 4.6 to move the graph a tiny distance to the left. As a result we obtain the function $f_{c',d}$ and $f_{c',d}^{A_T,B_T} = f_{c,d}^{A_T,B_T}$.

According to Lemma 4.7 the graph is moved upwards by decreasing the parameter d, right until the graph cuts the graph of its step function. Let $f_{c',d'}$ be the resulting function and $f_{c',d'}^{A_T,B_T}$ its step function. Obviously $f_{c',d'}^{A_T,B_T} = f_{c',d}^{A_T,B_T}$. We now have at least one element $p_1 \in A_T$, so that $f_{c',d'}(p_1) = f_{c',d'}^{A_T,B_T}(p_1) = q_1$.

If $f_{c',d'}$ contains another point $(p_2, q_2) \in A_T \times B_T$, we can be sure that q_2 is not equal to q_1 because the function is strictly monotonic. In this case we stop the transformation and encode the step function $f_{c',d'}^{A,B}$ by the triple $((p_1, q_1), (p_2, q_2), 1)$ where the last bit indicates that we stopped at this point.

Otherwise we modify the function $f_{c',d'}$ again to hit a second point of $A_T \times B_T$. Using Lemma 4.8 the graph is rotated clockwise by decreasing c' and adjusting d', so that the point (p_1, q_1) stays on the graph. We get a new function $f_{c'',d''}$ and another point $(p_2, q_2) \in A_T \times B_T$ with $f_{c'',d''}(p_2) = q_2$.

Now we have achieved that the function $f_{c'',d''}$ contains two tuples (p_1, q_1) and (p_2, q_2) that can be addressed by the variables in $X_T \cup Y_T$. The distance between these points is independent of the assignment to the variables in $X_S \cup Y_S$. In order to apply Lemma 4.3 we have to be sure, that (p_1, q_1) and (p_2, q_2) can be used to identify a shifted cutting of the initial graph $\frac{2^{2n-1}}{x}$, i.e., $\frac{2^{2n-1}}{x} \rightarrow \frac{2^{2n-1}}{c''+x} - d''$, with positive numbers in the denominator. The modification of d is not critical, because it does not have any influence on the denominator. For the values c we assure at the beginning that c is greater that 0 (either because $c = \|a_{X_S}\|$ is greater than 0 or by using $\epsilon_{min}/2$). Just the rotation operation decreases c. But as we continuously check, whether the value of $f_{c'',d''}$ hits a point in $A_T \times B_T$, it is impossible that the function's pole will be translated across any point of the grid. Therefore, Lemma 4.3 can be used to identify the underlying function $f_{c'',d''}$ with (p_1, q_1) and (p_2, q_2).

Our last step is now the reconstruction of the original step function $f_{c,d}^{A_T,B_T}$. If we have just moved the graph upwards without rotating it, then for every $x \in A_T$ the corresponding value of the step function $f_{c,d}^{A_t,B_T}$ is the

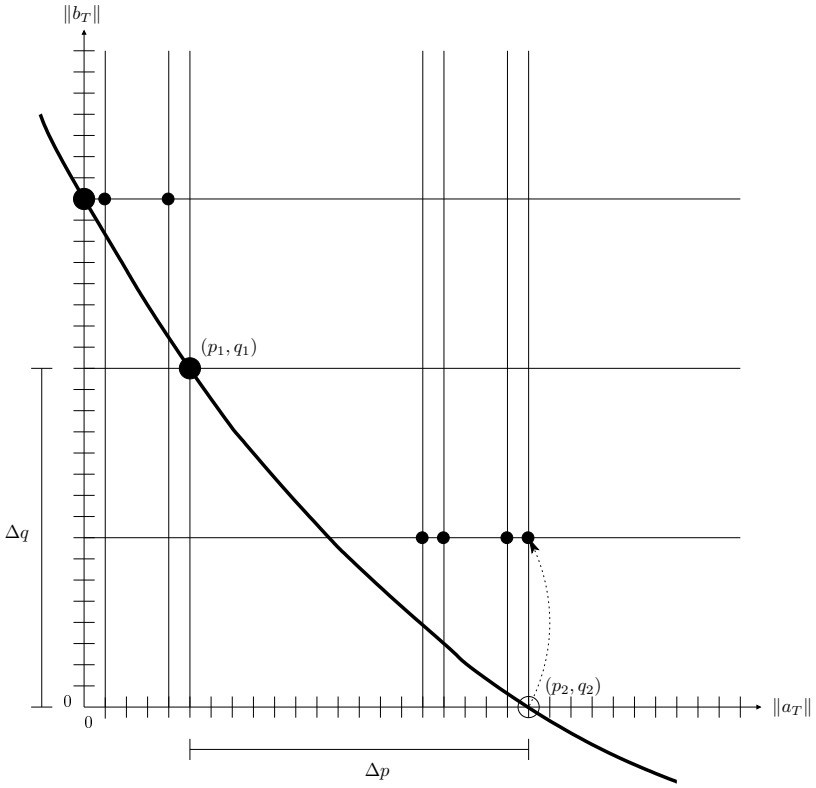

Figure 8: Reconstruction of the step function

smallest value of B_T that is at least $f_{c',d'}(x)$. In the other case we can use the second statement of Lemma 4.8 to reconstruct the original step function. Figure 8 illustrates the reconstruction of the step function $f_{c,d}^{A_T,B_T}$.

As we have seen a triple that consists of two points and an additional bit can encode any possible step function that itself represents a subfunction of $\mathrm{MUL}_{2n-1,n}$ obtained by replacing i x- and j y-variables by constants. As this subfunction can by uniquely reconstructed by this representative, there cannot be two different subfunctions with the same representation. The maximal number of these representation is

$$\underbrace{2^{n-i} \cdot 2^{n-j}}_{(p_1,q_1)} \cdot \underbrace{2^{n-i} \cdot 2^{n-j}}_{(p_2,q_2)} \cdot \underbrace{2}_{\text{bit } z} = 2^{2(n-i)+2(n-j)+1}.$$

Up to now we have assumed that for at least two A_T-values the function $f_{c,d}$ is greater than 0 and smaller or equal to the greatest value in B_T. A subfunction that is not of this type can be characterized by only one point (p, q) of the step function $f_{c,d}^{A_T,B_T}$. Summarizing there are less than $2^{2(n-i)+2(n-j)+2}$ different subfunctions.

Obviously there are at most 2^{i+j} different subfunctions obtained by the replacement of $i+j$ variables by constants. Using the minimum of the two upper bounds for each layer we obtain the result that the π-OBDD size for $\mathrm{MUL}_{2n-1,n}$ is $O(2^{(4/3)n})$ for any variable order π. □

Combining Theorem 4.9 with an upper bound of 2^{i+6} on the number of subfunctions obtained by the replacement of the variables $x_{n-1},\ldots,x_{n-i}$ and $y_{n-1},\ldots,y_{n-i}$ by constants presented in [3], we get the following result.

Corollary 4.10. *Let* $\pi = (x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0)$. *The* π-OBDD *size for the representation of* $\mathrm{MUL}_{2n-1,n}$ *is* $O(2^{(4/5)n})$.

4.3 More general models and the most significant bit of integer multiplication

Similar to the results presented in [19] for the middle bit of integer multiplication the lower bound on the OBDD size of the most significant bit can be extended to arbitrary oblivious binary decision diagrams of linear length. The complexity of $\mathrm{MUL}_{2n-1,n}$ for more general non-oblivious models than OBDDs is open.

Intuitively the most significant bit of integer multiplication seems to be much easier than the middle bit. Using the same proof method as described by Wegener and Woelfel [34] it can be shown that it is impossible to prove a better lower bound than $\Omega(n^{3/2}/\log n)$ and $\Omega(n^{3/2})$ for the branching program and Boolean formula size of the most significant bit using Nechiporuk's technique. Until now non-trivial lower bounds for the branching program and Boolean formula size are unknown.

5 A comparison between the middle and the most significant bit of integer multiplication

In this section we finish our considerations with a brief comparison between the functions $\mathrm{MUL}_{n-1,n}$ and $\mathrm{MUL}_{2n-1,n}$. For the most significant bit replacing a constant number of variables by constants may lead to a constant subfunction but for the middle bit we can replace almost an arbitrary quarter of the variables for each factor by constants without obtaining a constant subfunction. The best known variable order for the most significant bit is $\pi = (x_{n-1}, y_{n-1}, x_{n-2}, \ldots, x_0, y_0)$ and the π-OBDD size of $\mathrm{MUL}_{2n-1,n}$ is $O(2^{(4/5)n})$. For the middle bit of integer multiplication the best known variable order is $\pi' = (x_0, y_0, x_1, \ldots, x_{n-1}, y_{n-1})$ and the π'-OBDD size of

$\mathrm{MUL}_{n-1,n}$ is $\Theta(2^{(6/5)n})$. For each variable order there exists an assignment c to the variables of one factor such that the corresponding OBDD size of $\mathrm{MUL}^c_{n-1,n}$ is $\Omega(2^{n/2})$. In contrast it is not difficult to prove that for each variable order the corresponding OBDD size of $\mathrm{MUL}^c_{2n-1,n}$ is $O(n^2)$ for each assignment c. For the middle bit also large lower bounds for more general BDD models are known whereas exponential lower bounds for non-oblivious models are unknown for the most significant bit.

Conclusion

We have already learned in primary school how to multiply integers, nevertheless, the complexity of integer multiplication is a fascinating subject. Here, we have tried to deepen the knowledge on the set of subfunctions of the most significant bit of integer multiplication in order to obtain the best lower and upper bounds on its OBDD size.

Acknowledgments

I would like to thank Stefan Droste for proofreading a preliminary version of this survey.

References

[1] F.M. Ablayev and M. Karpinski. A lower bound for integer multiplication on randomized ordered read-once branching programs. *Information and Computation* 186(1), 78–89, 2003.

[2] N. Alon and W. Maass (1988). Meanders and their applications in lower bound arguments. *Journal of Computer and System Sciences* 37, 118–129, 1988.

[3] K. Amano and A. Maruoka. Better upper bounds on the QOBDD size of integer multiplication. *Discrete Applied Mathematics* 155, 1224–1232, 2007.

[4] B. Bollig. Larger lower bounds on the OBDD complexity of integer multiplication. *In Proc. of LATA*, LNCS 5457, 212–223, 2009.

[5] B. Bollig. On the OBDD complexity of the most significant bit of integer multiplication. *In Proc. of TAMC*, LNCS 4978, 306-317, 2008.

[6] B. Bollig. Restricted nondeterministic read-once branching programs and an exponential lower bound for integer multiplication. *RAIRO Theoretical Informatics and Applications* 35, 149–162, 2001.

[7] B. Bollig and J. Klump. New results on the most significant bit of integer multiplication. *In Proc. of ISAAC*, LNCS 5369, 129–140, 2008.

[8] B. Bollig, St. Waack, and P. Woelfel. Parity graph-driven read-once branching programs and an exponential lower bound for integer multiplication. *Theoretical Computer Science* 362, 86–99, 2006.

[9] B. Bollig and P. Woelfel. A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing. *In Proc. of 33rd STOC*, 419–424, 2001.

[10] B. Bollig and P. Woelfel. A lower bound technique for nondeterministic graph-driven read-once branching programs and its applications. *Theory of Computing Systems* 38, 671–685, 2005.

[11] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Computers* 35, 677–691, 1986.

[12] R.E. Bryant. On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. *IEEE Trans. on Computers* 40, 205–213, 1991.

[13] A. De, P. Kurur, C. Saha, and R. Sapthariski. Fast integer multiplication using modular arithmetic. *In Proc. of 40th STOC*, 499–506, 2008.

[14] M. Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. *In Proc. 13th STACS*, LNCS 1046, 569–580, 1996.

[15] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms* 25, 19–51, 1997.

[16] M. Fürer. Faster integer multiplication. *In Proc. of 39th STOC*, 57–66, 2007.

[17] R. Gentilini, C. Piazza, and A. Policriti. Computing strongly connected components in a linear number of symbolic steps. *In Proc. of SODA*, ACM Press, 573–582, 2003.

[18] R. Gentilini, C. Piazza, and A. Policriti. Symbolic graphs: linear solutions to connectivity related problems. *Algorithmica* 50, 120–158, 2008.

[19] J. Gergov. Time-space trade-offs for integer multiplication on various types of input oblivious sequential machines. *Information Processing Letters* 51, 265–269, 1994.

[20] A. Gronemeier. Approximating Boolean functions by OBDDs. *Discrete Applied Mathematics*, 155(2), 194–209, 2007.

[21] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. *In Proc. 28th FOCS*, 99–110, 1987.

[22] J. Hromkovič. *Communication Complexity and Parallel Computing.* Springer Verlag, 1997.

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

[24] É.I. Nechiporuk. A Boolean function *Soviet Mathematics Doklady* 7 (4), 999–1000, 1966.

[25] S. Ponzio. A lower bound for integer multiplication with read-once branching programs. *SIAM Journal on Computing* 28, 798–815, 1998.

[26] M. Sauerhoff. An asymptotically optimal lower bound on the OBDD size of the middle bit of multiplication for the pairwise ascending variable order. Submitted, 2008.

[27] M. Sauerhoff and P. Woelfel. Time-space trade-off lower bounds for integer multiplication and graphs of arithmetic functions. In *Proc. of 33rd STOC*, 186–195, 2003.

[28] D. Sawitzki. Lower bounds on the OBDD size of graphs of some popular functions. *In Proc. of SOFSEM*, LNCS 3381, 298-309, 2005.

[29] D. Sawitzki. Exponential lower bounds on the space complexity of OBDD-based graph algorithms. In *Proc. of LATIN*, LNCS 3831, 471-482, 2006.

[30] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing* 7, 281–292, 1971.

[31] D. Sieling and I. Wegener NC-algorithms for operations on binary decision diagrams. *Parallel Processing Letters* 48, 139–144, 1993.

[32] I. Wegener. *Branching Programs and Binary Decision Diagrams - Theory and Applications.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[33] I. Wegener. Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. *Information Processing Letters* 46/2, 85–87, 1993.

[34] I. Wegener and P. Woelfel. New results on the complexity of the middle bit of multiplication. *Computational Complexity*, 16(3), 298–323, 2007.

[35] P. Woelfel. Bounds on the OBDD-size of integer multiplication via universal hashing. *Journal of Computing and System Science* 71(4), 520–534, 2005.

[36] P. Woelfel. Efficient strongly universal and optimally universal hashing. *In Proc. 24th MFCS*, LNCS 1672, 262–272, 1999.

[37] P. Woelfel. On the complexity of integer multiplication in branching programs with multiple tests and in read-once branching programs with limited nondeterminism. *In Proc. of 17th Conference on Computational Complexity*, 80–89, 2002.

[38] P. Woelfel. Symbolic topological sorting with OBDDs. *Journal of Discrete Algorithms* 4(1), 51–71, 2006.

[39] B. Yang, Y.-A. Chen, R.E. Bryant, and D.R. O'Hallaron. Space- and time-efficient BDD construction via working set control. *In Proc. of the Asia South-Pacific Design Automation Conference (ASPDAC)*, 423–432, 1998.