# A Lower Bound for Integer Multiplication on Randomized Read-Once Branching Programs

FARID ABLAYEV * MAREK KARPINSKI †

**Abstract**

We prove an exponential lower bound $2^{\Omega(n/\log n)}$ on the size of any *randomized* ordered read-once branching program computing *integer multiplication*. Our proof depends on proving a new lower bound on Yao's randomized one-way communication complexity of certain boolean functions. It generalizes to some other common models of randomized branching programs. In contrast, we prove that *testing integer multiplication*, contrary even to nondeterministic situation, can be computed by *randomized* ordered read-once branching program in polynomial size. It is also known that computing the latter problem with deterministic read-once branching programs is as hard as factoring integers.

## 1 Preliminaries

*Oblivious* (or *ordered*) read-once branching programs become an important tool in the field of digital design and verification (see, for example, [8] and [22]). In

*Dept. of Computer Science University of Bonn. Visiting from University of Kazan. Research partially supported by the Volkswagen-Stiftung and Russia Fund for Basic Research 96-01-01962. Email: `ablayev@cs.uni-bonn.de`

†Dept. of Computer Science University of Bonn, and International Computer Science Institute, Berkeley, California. Research partially supported by DFG Grant KA 673/4-1, by the ESPRIT BR Grants 7097, and EC-US 030, and by the Max–Planck Research Prize. Email: `marek@cs.uni-bonn.de`

these fields they are also known as "OBDDs" (ordered binary decision diagrams). There are some important practical functions which are *hard* for OBDDs. One of such functions is integer multiplication [7]. The other function is testing multiplication for which there is an exponential lower bound ($2^{\Omega(n^{1/4})}$) known for nondeterministic OBDDs [12]. An interesting open problem remained whether randomization can help in computation of these functions by OBDDs. In this paper we show, firstly, that the method of [4] yields polynomial size ($O(n^6 \log^4 n)$) bound for the latter function for randomized OBDDs. Interestingly, it is known that computing this function with deterministic read-once branching programs is as hard as integer factoring [22, 15]. Further we prove an exponential lower bound $2^{\Omega(n/\log n)}$ on the size of any randomized OBBD computing *integer multiplication*.

During last decade there were several attempts to find generalizations of OBDDs model for hardware verification, strong enough to compute efficiently integer multiplication. But again the results showed that multiplication remained hard for these models ([11, 15]).

In [4], a randomized model of branching programs was introduced. The importance of this model was highlighted by the fact that there is a function which is hard for deterministic OBDDs but is easy for randomized OBDDs [4]. During the last couple of years new examples of such function were presented by different authors. For example, *clique-only function* is hard for nondeterministic syntactic read-$k$-times branching programs [5] but is simple for randomized OBDDs [18, 20]. See [21] for another example.

It was proved that randomized and nondeterministic models of OBDD are incomparable [2]. So there was still hope (note that multiplication is hard for nondeterministic OBDD [11]) that randomized OBDDs can compute integer multiplication in polynomial size. Our results show that randomized OBDDs can test integer multiplication in polynomial size but integer multiplication itself requires exponential size.

Up to now it was not clear what is harder to multiply or to test the multiplication (see [16] for more information). It is known that $DMULT$ (testing multiplication) is hard for *syntactic nondeterministic read-k-times* branching programs [12]. Note that $DMULT$ function is $AC^0$ equivalent to $MULT$ [9]. Our result answers also to the open problem raised in [22] about succinct representations for functions $DMULT$ and $MULT$.

We recall now basic definitions ([17]).

A *deterministic* branching program $P$ for computing a boolean function $g : \{0,1\}^n \to \{0,1\}$ is a directed acyclic multi-graph with a distinguished source node $s$ and a distinguished sink node $t$. The out degree of of each non-sink node is exactly 2 and the two outgoing edges are labeled by $x_i = 0$ and $x_i = 1$ for variable $x_i$ associated with the node. Call such node an $x_i$-node. The label "$x_i = \delta$" indicates that only inputs satisfying $x_i = \delta$ may follow this edge in the computation. The branching program $P$ computes a function $g$ in the obvious way: for each $\sigma \in \{0,1\}^n$ we let $f(\sigma) = 1$ iff there is a directed $s - t$ path starting in the source $s$ and leading to to the (accepting) node $t$ such that all labels $x_i = \sigma_i$ along this path are consistent with $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$.

We define a *randomized* branching program [4] as a program having in addition specially designated *random* ("coin-toss") inputs. When values of these random inputs are chosen from the uniform distribution, the output of the branching program is a random variable.

We say that a randomized branching program $(a, b)$-computes a boolean function $f$ if it outputs 1 with probability at most $a$ for input $\sigma$ such that $f(\sigma) = 0$ and outputs 1 with probability at least $b$ for inputs $\sigma$ such that $f(\sigma) = 1$. For $1 \geq p > 1/2$ we write shortly "$p$-computes" instead of "$(1 - p, p)$-computes". A randomized braching program computes a function $g$ with on-sided error if it $(\epsilon, 1)$-computes $g$.

We define the size of $(P)$, $size(P)$, (complexity of the branching program $P$) as the number of its *internal* nodes.

Read-once branching program is a branching program in which every variable is tested at most once in every path. A $\tau$-ordered read-once branching program is a read-once branching program which respects an ordering $\tau$ of the variables, i.e. if an edge leads from an $x_i$-node to an $x_j$-node, the condition $\tau(i) < \tau(j)$ has to be fulfilled. An OBDD (alternatively ordered read-once branching program) is a $\tau$-ordered read-once branching program respecting some ordering $\tau$ of variables.

# 2 Results

We start with defining a boolean decision function: *the testing integer multiplication function* (or alternatively, *decision problem of recognizing the graph* *of multiplication*) $DMULT$ as follows. $DMULT : \{0,1\}^{3n} \to \{0,1\}$ and $DMULT(X,Y,Z) = 1$ iff $XY = Z$. Here $X, Y$, and $Z$ are binary representations of integer numbers, $|X| = |Y| = n$, $|Z| = 2n$.

**Theorem 1** *Function $DMULT$ can be computed by a randomized OBDD with one-sided $\varepsilon(n)$-error of size*

$$O\left(\frac{n^6}{\varepsilon^5(n)} \log^4 \frac{n}{\varepsilon(n)}\right).$$

**Proof.** Uniformly at random select a prime number $p$ from the set $Q_{d(n)} = \{p_1, \ldots, p_{d(n)}\}$, $d(n) = O(n)$, of first $d(n)$ primes. Then deterministically count $a = X \bmod p$, $b = Y \bmod p$, multiply $ab$, then count $c = Z \bmod p$, and verify whether $ab = c$. If $ab = c$ then *accept* an input else *reject*. Chinese reminder theorem provides the correctness of such computation and fingerprinting arguments of [4] provide a correct result of testing $XY = Z \bmod p$ by randomized OBDDs with high probability. All these manipulations can be done by a polynomial size randomized OBDD $P$ constructed below.

*Phase 1. (randomized).* Choose $d(n)$ to be some function in $O(n)$, s.t. $d(n) > 4n$. $P$ randomly selects a prime number $p$ from the set $Q_{d(n)} = \{p_1, p_2, \ldots, p_{d(n)}\}$ of first $d(n)$ prime numbers.

$P$ uses $t = \lceil \log d(n) \rceil$ random bits for selecting a prime number $p$. $P$ reads random bits in the order $\xi_1, \ldots, \xi_t$. $\xi = \xi_1 \ldots \xi_t$ is interpreted as binary notation of a number $N(\xi)$. $P$ selects $i$-th prime number $p_i \in Q_{d(n)}$ iff $N(\xi) = i \bmod d(n)$.

*Phase 2. (deterministic).* During a computation path $P$ counts $a = X \bmod p$, by reading consequently bits from $X$. $P$ stores $a$ by internal node (state). Then, $P$ counts $b = Y \bmod p$ and stores the product $ab$. At last $P$ counts $c = Z \bmod p$ and verify whether $ab = c$. If $ab = c$ then it *accepts* else it *rejects*.

So, if $XY = Z$, then $P$ with probability 1 outputs the correct answer. If $XY \neq Z$, then it can happen that $XY = Z \pmod{p}$ for some $p \in Q_{d(n)}$. In these cases $P$ makes an error.

For $XY \neq Z$ we have $|XY - Z| \leq 2^{2n} < p_1 \cdots p_{2n}$ where $p_1, \ldots, p_{2n}$ are the first $2n$ prime numbers. This means that in the case when $XY \neq Z$, the probability $\varepsilon(n)$ of the error of $P$ on the input $X, Y, Z$ is less than equal to $4n/d(n)$ (less than equal to $2n/d(n)$ if $t$ is a power of 2).

For $p \in Q_{d(n)}$ denote by $S_p$ a deterministic subprogram of $P$ that carries out the deterministic part of computations of the *phase 2* with the prime $p$.

The size of $P$ is bounded by

$$2^{t+1} - 1 + \sum_{p \in Q_{d(n)}} size(S_p).$$

$S_p$ has the length $3n$. For the realization of the procedure described in the *phase 2* it is sufficient to store in the internal nodes four numbers: $X \bmod p, Y \bmod p, XY \bmod p$ and $Z \bmod p$. The $i$-th prime is of order $O(i \log i)$. Therefore we have

$$size(S_p) = O(np^4) = O(n(d(n) \log d(n))^4).$$

¿From the above upper bounds for the $size(S_p)$, $size(P)$ and from the upper bound for $\varepsilon(n)$ ($\varepsilon(n) < 4n/d(n)$), the upper bound of the theorem follows. ▮

We define now *integer multiplication function MULT* as follows. The function $MULT_k : \{0,1\}^{2n} \to \{0,1\}$ computes the $k$-th bit, $0 \leq k \leq 2n-1$ in the product of two $n$-bit integers. That is $MULT_k(X,Y) = z_k$ where $X = x_{n-1} \dots x_0$, $Y = y_{n-1} \dots y_0$, and $Z = z_{2n-1} \dots z_0$. Now denote by $MULT$ function $MULT_{n-1}$ which computes the *middle* bit in the product $xy$. It is known that the middle bit is the "hardest" bit (see, for example [15]).

For $p \in (1/2, 1)$, $k \in \{0, \dots, 2n-1\}$, and a permutation $\tau$ of $\{1, \dots, 2n\}$ let $P_p(k,\tau)$ be a randomized OBDD with the ordering $\tau$ that $p$-computes $MULT_k$.

**Theorem 2** *Given $p \in (1/2,1)$. For every $\tau$ there exists a $k$ such that*

$$size(P_p(k,\tau)) \geq 2^{n(1-H(p))/8},$$

*where $H(p) = -p \log p - (1-p)\log(1-p)$ is Shannon entropy.*

**Theorem 3** *Let for $p \in (1/2,1)$ the function $MULT(X,Y)$ is $p$-computed by a randomized OBDD $P$. Then*

$$size(P) \geq 2^{\Omega(n/\log n)}.$$

These two theorems state that multiplication is hard for randomized OBDD. The first one is "theoretically weaker" than the second. But the proof of the first one is shorter and more direct. It is based on proving lower bound for the polynomial projection function of $MULT_k$ ([6]). The proof of the theorem 3 itself is based on proving lower bound for another polynomial projection of $MULT$ [7, 11] using randomized binary search communication game. See [14, 13] for more information. Proofs of the theorems are presented in the next section.

# 3 Proofs

## 3.1 Proof of the theorem 2

Our proof proceeds as follows:
i) we construct a polynomial projection $f^{k,\tau}$ of $MULT_k$ and then
ii) we prove that $f^{k,\tau}$ is hard for a randomized $\tau$-ordered OBDD.

For an arbitrary ordering $\tau$ in a randomized OBDD, there are two subsets $L$ and $W$ of equal sizes $l \geq n/2$ such that:

1) $P$ reads all variables from $L$ before starting reading variables from $W$ and

2) $L \subset X$ and $W \subset Y$ or $L \subset Y$ and $W \subset X$.

W.l.g. assume in the rest of the proof that $L \subset X$ and $W \subset Y$. So, $L = \{x_{i_1}, \ldots, x_{i_l}\}$ and $W = \{y_{j_1}, \ldots, y_{j_l}\}$.

¿From now on we are interested only in inputs $\sigma \in \{0,1\}^{2n}$ such that:
for variables $Y$ all bits of $\sigma$ except for a one bit of $W$ are 0. Call such $W$ *control* set. Variables from $L$ can take arbitrary values from $\{0,1\}$. For convenience fix the remaining variables from $X \backslash L$ to be 0. Call such $L$ *data* set.

Denote by $[k]$ a set of pair of bits of data and control sets that are transmitted to the $k$-th bit of the product $XY$. Formally

$$[k] = \{(x_i, y_j) \in L \times W : i + j = k\}.$$

As $|L \times W| = l^2 \geq n^2/4$, there exists a $k$ such that

$$|[k]| = t \geq l^2/(2n) = n/8. \tag{1}$$

Now fix this set $[k]$. Denote by $L_k \subset L$ ($W_k \subset W$) a subset of $L$ ($W$) that consists of all variables $x_i$ ($y_j$) that "take part" in the set $[k]$.

Consider a projection $f^{k,\tau} : L_k \times W_k \rightarrow \{0,1\}$ of $MULT_k$, for which all variables from $(Y \cup X) \backslash (L_k \cup W_k))$ are fixed and equal 0. The communication matrix $CM$ of $f^{k,\tau}$ for a partition $(L_k, W_k)$ of inputs has the following property:
1) it is $2^t \times t$ boolean matrix and
2) all rows of $CM$ are different.

We use now Yao's standard randomized one-way communication computation [23, 24] (see also [13]) for boolean functions.

The following lemma is proved in [2]. It states the connection between the size of OBDDs and the one-way communication complexity. Consider a boolean function $h : \{0,1\}^m \rightarrow \{0,1\}$. Let $U = V \times R$ be partition of a set of variables of $h$ into two parts. For $p > 1/2$ denote by $PC_p^U(h)$ a randomized one-way communication $p$-computation for $h$ (a computation which outputs the correct result with the probability greater or equal to $p$) according to the partition $U$ of inputs.

**Lemma 1** *Let $\varepsilon \in [0, 1/2]$, $p = 1/2 + \varepsilon$. Let a randomized OBDD $P$ $p$-computes the function $h$. Let $U = V \times R$ be a partition of inputs between players with $V$ and $R$ defined according to ordering $\tau$ of inputs of $P$. That is $P$ can read variables from $R$ only after reading variables from $L$ and does not read variables from $L$ after starting reading variables from $R$. Then*

$$size(P) \geq 2^{PC_p^U(h)-1}.$$

Now use the theorem proved in [1] which states that the randomized one-way communication complexity cannot be too "small" for a function with a "large" *data set* and a "small" *control set.*

Choose a set $Z \subseteq R$ such that for an arbitrary two words $u, u' \in V$ there exists a word $y \in Z$ such that $h(u,y) \neq h(u',y)$. *The set $Z$ is called the control set for the matrix CM.*

Denote by $ts(CM)$ the minimum size of a control set for matrix $CM$ and $nrow(CM)$ the number of different rows of matrix $CM$.

For a number $p \in [1/2, 1]$, define (probabilistic communication characteristic (cf. [1])) $pcc_p^U(h) = \frac{ts(CM)}{\log nrow(CM)} H(p)$, where $H(p) = -p \log p - (1-p)\log(1-p)$ is the Shannon entropy [10].

**Theorem 4 ([1])** *Let $\varepsilon \in [0, 1/2]$ and $p = 1/2 + \varepsilon$. Let $U \subseteq \{0,1\}^n$ be such that $U = V \times R$, where $V$ and $R$ are defined in according to partition $\pi$ of inputs of function $h : \{0,1\}^n \to \{0,1\}$. Then*

$$PC_p^U(h) \geq DC^U(h)(1 - pcc_p^U(h)) - 1,$$

*where $DC^U(h)$ is the deterministic one-way communication complexity of $h$.*

In our case we have that 1) for $U = L_k \times W_k$ $pcc_p^U(f^{k,\tau}) = H(p)$ and 2) $DC^U(f^{k,\tau}) = \log t$ (because all rows of the communication matrix $CM$ are different). From the above we get that

$$size(P) \geq 2^{t(1-H(p))}.$$

Using (1) and the inequality above we get the lower bound of the theorem.

## 3.2 Proof of the theorem 3

The proof consists of 3 steps:

i) we construct a polynomial projection $f$ of $MULT$ (cf. [7, 11]),

ii) using randomized OBDD $P$ for $MULT$ (which is turned to a randomized OBDD for $f$ when values of proper variables are fixed) construct a randomized one-way communication protocol for computing the function $g$ defined in [19],

and

iii) finally we prove the lower bound of the theorem, using the fact

— that randomized one-way communication complexity gives the lower bound for randomized OBDD size [2] and

— that $g$ is hard for randomized one-way communication computation [2].

Let $\tau$ be an ordering of variables of randomized OBDD $P$. Then there are two subsets $L$ and $W$ of the set $X$ such that:

1) $|L| = |W| = l(n) = \Omega(n)$ and

2) $P$ reads all variables from $L$ before starting reading variables from $W$.

Now if the *remaining* variables (variables from $(Y \cup X) \backslash (L \cup W)$) are fixed in a proper way, then randomized OBDD $P$ $p$-computes the boolean function $f$ (polynomial projection of $MULT$) which has the following communication description. Communication matrix $CM(f)$ of size $2^{l(n)} \times 2^{l(n)}$ for $f$ with rows corresponding to variables from $L$ and columns to variables from $W$ is the lower triangle boolean matrix. That is all the elements above the second diagonal are 0 and all elements in the second diagonal and below it — are 1 [11]. Formally, function $f(L, W)$ can be described as follows. View $L$ and $W$ as a binary presentation of numbers. Numbers presented in the reverse order (first bits of $L$ and $W$ represent the lowest bits and last bits — the highest bit of a number). Then $f(L, W) = 1$ iff $L + W \geq 2^{l(n)+1}$ [11].

We assume in the remaining part of the proof the variables from $(Y \cup X) \backslash (L \cup W)$ been fixed as needed. So $P$ is turned to the randomized OBDD that $p$-computes $f$. Below, using $P$ we construct a randomized one-way communication protocol $\Phi$ for a "pointer" function.

The "pointer" function $g_n$ ([19]) is defined as follows. Let $n$ be an integer and let $p[n]$ be the smallest prime number greater than or equal to $n$. Then, for every integer $s$, let $\omega_n(s)$ be defined as follows. Let $j$ be the unique integer satisfying $j = s \bmod p[n]$ and $1 \leq j \leq p[n]$. Then, $\omega_n(s) = j$, if $1 \leq j \leq n$, and $\omega_n(s) = 1$ otherwise.

For every $n$, the boolean function $g_n : \{0,1\}^n \rightarrow \{0,1\}$ is defined as $g_n(\sigma) = \sigma_j$, where $j = \omega_n(\sum_{i=1}^{n} i\sigma_i)$.

For the purposes of the proof we use the following "communication" variant of the "pointer" function $g$ in the remaining part of the proof.

Let $L = \{x_{i_1}, \ldots, x_{i_{l(n)}}\}$. Let for $k(n) = \log l(n)$ (w.l.g. we consider that $l(n)$ is a power of 2) $R = \{z_1, z_2, \ldots, z_{k(n)}\}$ is a set of "new" variables, that is $R$ does not contains variables from $X \cup Y$. Then define a "communication" variant of the "pointer" function $g$ as $g : L \times R \rightarrow \{0,1\}$.

We use now Yao's standard randomized one-way communication computation for $g$ when the first player $I$ gets values of the variables from $L$ and the second player $II$ gets values from the remaining variables $R$. Player $I$ starts the computation on his part of inputs, then the player $II$, on receiving a message from $I$ and his part of the input, outputs the result.

Below, in Lemma 2 we construct a randomized one-way communication protocol $\Phi$ for $q$-computing ($q \in (1/2, 1)$) $g$ such that

$$C(\Phi) \leq a(\log bl(n))(\log size(P)), \tag{2}$$

where $a$, $b$ are positive constants. Then we prove (see Lemma 3 below) that for this partition of inputs between players, the following lower bound for randomized one-way communication $q$-computation is true

$$PC_q(g) \geq c(q)l(n), \tag{3}$$

where $c(q)$ is positive constant. As the inequality (3) is correct for all the randomized one-way communication protocols that $q$-computes $g$ then from (2) we get the lower bound of the theorem.

$$size(P) \geq 2^{cl(n)/\log l(n)}.$$

**Lemma 2** *For $q \in (1/2, 1)$ there is a randomized one-way communication protocol $\Phi$ for q-computing function $g$ such that*

$$C(\Phi) \leq a(\log bl(n))(\log size(P)),$$

*where $a$, $b$ are positive constants.*

**Proof.** We describe a randomized one-way communication protocol $\Phi$ for $q$-computing the "pointer" function $g$ as follows. Let $\sigma = \sigma_1, \ldots, \sigma_{l(n)}$ be an input sequence of player $I$ and $\omega = \omega_1, \ldots, \omega_{k(n)}$ — an input sequence of player $II$. Let $t(n) = a\log(bl(n))$. We define constants $a, b$ later in a proper way. Player $I$ runs branching program $P$ on his part of inputs $t(n)$ times and sends $t(n)$ nodes $v_1, \ldots, v_{t(n)}$ which were reached by $P$ during the computations to the player $II$. The goal of player $II$ is to determine the input string $\sigma$ of player $I$ with probability no less than $q$ (more precisely player $II$ determines a string $\sigma'$ such that probability of the event $\sigma' = \sigma$ is no less than $q$). Then, player $II$ having his part of input can outputs the correct result with probability no less than $q$. Let $B_0 := \{0,1\}^{l(n)}$. In each step $i \geq 1$, $II$ reduces a set $B_{i-1}$ and in the last step $l(n)$ of procedure $II$ gets a set $B_{l(n)} = \{\sigma'\}$. Player $II$ after getting $v_1, \ldots, v_{t(n)}$ determines $\sigma'$ by a randomized binary search procedure as follows.

**Step 1.** Take a "middle" input sequence $\sigma^1$ (sequence $\sigma^1$ determines the middle column of the communication matrix $CM(f)$. Columns of $CM(f)$ are ordered in a natural order of input strings, that is $\mathbf{0} = (0, \ldots, 0), \ldots, \mathbf{1} = (1, \ldots, 1)$).

Run $P$ on $\sigma^1$ $t(n)$ times starting from nodes $v_1, \ldots, v_{t(n)}$ and take the majority result $\Delta_1 \in \{0,1\}$. Using $\Delta_1$, select a set $B_1$ of potential inputs of player $I$ (the set of sequences that determine the upper half of rows of $CM(f)$ or the set of sequences that determine the lower half of rows of $CM(f)$). $|B_1| = 2^{l(n)}/2$.

**Step 2.** If $\Delta_1 = 1$ then select a "middle" input sequence $\sigma^2$ between $\sigma^1$ and **1** else — between **0** and $\sigma^1$.

Run $P$ on $\sigma^2$ $t(n)$ times starting from nodes $v_1, \ldots, v_{t(n)}$ and take the majority result $\Delta_2 \in \{0,1\}$. Using $\Delta_2$, select a set $B_2 \subset B_1$ of potential inputs of player $I$. $|B_2| = |B_1|/2$.

After $l(n)$ steps procedure stops by selecting a set $B_{l(n)}$ that consists of unique input sequence $\sigma'$. Player $II$ outputs the result $g(\sigma', \omega)$. Clearly we have

$$C(\Phi) \leq t(n) \log size(P)).$$

The following counting arguments show that protocol $\Phi$ $q$-computes $g$.

For a string $\gamma \in \{0,1\}^{l(n)}$ that determines a column of matrix $CM(f)$ denote by $Pr(\gamma)$ a probability of getting the correct result $\Delta$ by the binary search procedure above. Then the probability $Pr(\sigma' = \sigma)$ of correctly determining an input of player $I$ is

$$Pr(\sigma' = \sigma) = Pr(\sigma^1) \ldots Pr(\sigma^{l(n)}).$$

The probability $1 - Pr(\gamma)$ of getting error $\Delta$ is no more than $(1/c(p))^{t(n)}$ for some constant $c(p) > 1$ depending on probability $p$ of correct computation of $P$ (see, e.g., [14]). By choosing a constant $a$ in a proper way we get

$$1 - Pr(\gamma) \leq 1/(bl(n)).$$

¿From the above it follows that

$$Pr(\sigma' = \sigma) \geq (1 - 1/(bl(n)))^{l(n)}.$$

Using the fact that function $(1-1/x)^{x/b}$ is monotonically increasing to $(1/e)^{1/b}$ for $x \to \infty$ we get for properly selected constant $b > 1$ and for $n$ large enough

$$Pr(\sigma' = \sigma) \geq q.$$

∎

We formulate now the last lemma.

**Lemma 3** *For arbitrary $q \in (1/2, 1)$ and arbitrary $\delta > 0$ and for every $n$ large enough, we have*

$$PC_q(g) \geq (l(n) - o(l(n)))(1 - (1+\delta)H(q)).$$

*where $H(q) = -q\log q - (1-q)\log(1-q)$ is Shannon entropy.*

See [2] for the proof of the lower bound of the lemma.

# 4 Generalization and concluding remarks

Note that in the proof technique used in the section above for ordered read-once branching programs we used the following essential fact. The set of variables of $P$ can be partitioned (according to the ordering $\tau$ of $P$) into two parts $L$ and $W$ (of approximately equal sizes) such that for any computation path of $P$ the following is true. If a variable from $W$ is tested, then no variable from $L$ can be tested in the rest of this path. This means that the statement of the theorem 3 is true also for other common models of branching programs we define below.

Define a *balanced partitioning* as any partition of a set $X$ (more precisely the sequence of sets) into subsets $X_1$ and $X_2$ of $|X_1| = \Theta(|X_2|)$.

**Definition 1** *Call branching program $P$ a $\pi$-balanced-weak-ordered branching program if it respects a balanced partition $\pi$ of its variables $X$ into two parts $X_1$ and $X_2$ such that if an edge leads from an $x_i$-node to an $x_j$-node, where $x_i \in X_t$ and $x_j \in X_m$, then the condition $t \leq m$ has to be fulfilled.*

*Call branching program $P$ an balanced-weak-ordered if it is $\pi$-balanced-weak-ordered for some partition $\pi$ of the set of variables of $P$ into two sets.*

Our theorem 3 can be generalized as follows.

**Theorem 5** *Let for $p \in (1/2, 1)$ the function $MULT(X, Y)$ be $p$-computed by randomized balanced-weak-ordered branching program $P$. Then*

$$size(P) \geq 2^{\Omega(n/\log n)}.$$

# Open problems

It is an interesting open problem to prove a lower bound for integer multiplication on randomized branching programs with 1) limited number of inputs readings, and 2) without any condition on ordering of variables. We conjecture that the corresponding lower bounds are also exponential.

**Acknowledgment** We would like to thank Anna Gál, Stephen Ponzio, Sasha Razborov, Thomas Thierauf and Andy Yao for helpful discussion on the subject of the paper.

# References

[1] F. Ablayev, Lower bounds for one-way probabilistic communication complexity *in Proceedings of the ICALP'93, Lecture Notes in Computer Science, Springer-Verlag*, 700, (1993), 241-252.

[2] F. Ablayev, Randomization and nondeterminism are incomparable for ordered read-once branching programs, *in Proceedings of the ICALP'97, Lecture Notes in Computer Science, Springer-Verlag*, 1256, (1997), 195-202.

[3] F. Ablayev and M. Karpinski, On the power of randomized branching programs, *in Proceedings of the ICALP'96, Lecture Notes in Computer Science, Springer-Verlag*, 1099, (1996), 348-356.

[4] F. Ablayev and M. Karpinski, On the power of randomized ordered branching programs, Research Report 85181-CS, University of Bonn, 1997.

[5] A. Borodin, A. Razborov, and R. Smolensky, On lower bounds for read-$k$-times branching programs, *Computational Complexity*, 3, (1993), 1-18

[6] R. Bryant, Graph-based algorithms for boolean function manipulation *IEEE Trans. Comput.*, C-35, (8), (1986), 677-691.

[7] R. Bryant, On the complexity of VLSI implementations and graph representations of boolean functions with applications to integer multiplication, *IEEE Trans. Comput.*, 40 (2), (1991), 205-213.

[8] R. Bryant, Symbolic boolean manipulation with ordered binary decision diagrams, *ACM Computing Surveys*, 24, No. 3, (1992), 293-318.

[9] R. Buss, The graph of multiplication is equivalent to counting, *Information Processing Letters*, 41, (1992), 199-201.

[10] R. Gallager, Information theory and reliable communication, *Wiley, New York*, 1968.

[11] J. Gergov, Time-space tradeoffs for integer multiplication on various types of input oblivious sequential machines, *Information Processing Letters*, 51, (1994), 265-269.

[12] S. Jukna, The graph of integer multiplication is hard for read-$k$-times networks, *TR 95-10 Mathematik/Informatik University of Trier*, 1995.

[13] E. Kushilevitz and N. Nisan, Communication complexity, *Cambridge University Press*, 1997.

[14] R. Motwani and P. Raghavan, Randomized Algorithms, *Cambridge University Press*, 1995.

[15] S. Ponzio, A lower bound for integer multiplication with read-once branching programs, *Proceedings of the 27-th STOC*, (1995), 130-139.

[16] S. Ponzio, Restricted branching programs and hardware verification, *Technical Report*, MIT/LCS-TR-633, MIT, 1995

[17] A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, *in Proceedings of the FCT'91, Lecture Notes in Computer Science, Springer-Verlag*, 529, (1991), 47–60.

[18] T. Thierauf, *personal communication*, 1997.

[19] P. Savicky, S. Zak, A large lower bound for 1-branching programs, *Electronic Colloquium on Computational Complexity*, Revision 01 of TR96-036, (1996), available at `http://www.eccc.uni-trier.de/eccc/`

[20] M. Sauerhoff, personal communication, 1997.

[21] M. Sauerhoff, On nondeterminism versus randomness for read-once branching programs *Electronic Colloquium on Computational Complexity*, TR97-030, (1997), available at `http://www.eccc.uni-trier.de/eccc/`

[22] I. Wegener, Efficient data structure for Boolean functions, *Discrete Mathematics*, 136, (1994), 347-372.

[23] A.C. Yao, Some Complexity Questions Related to Distributive Computing, *in Proc. of the 11th Annual ACM Symposium on the Theory of Computing*, (1979), 209-213.

[24] A.C. Yao, Lower bounds by probabilistic arguments, *in Proc. of the 27th Annual IEEE Symposium on Foundations of Computer Science* (1983), 420-428.