The Computational Complexity of Universal Hashing

Yishay Mansour* Noam Nisan † Prasoon Tiwari‡

July 14, 2002

Abstract

Any implementation of Carter-Wegman universal hashing from n-bit strings to m-bit strings requires a time-space tradeoff of $TS = \Omega(nm)$. The bound holds in the general boolean branching program model, and thus in essentially any model of computation. As a corollary, computing $a + b * c$ in any field F requires a quadratic time-space tradeoff, and the bound holds for any representation of the elements of the field.

Other lower bounds on the complexity of any implementation of universal hashing are given as well: Quadratic AT^2 bound for VLSI implementation; $\Omega(\log n)$ parallel time bound on a CREW PRAM; and exponential size for constant depth circuits.

1 Introduction

Carter and Wegman [9] defined the notion of a *universal family of hash functions.*

*Laboratory for Computer science, MIT, 545 Tech. Sq., Cambridge, MA 02139. Supported by an IBM graduate fellowship.

†Dept. of Computer Science, Hebrew University, Jerusalem, Israel. Research was done while the author was at Laboratory for Computer science, MIT, 545 Tech. Sq., Cambridge, MA 02139. Partially supported by NSF 865727-CCR and ARO DALL03-86-K-017.

‡This work was performed while the author was at IBM Research Division, T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. Author's current address: Department of Computer Science, University of Wisconsin–Madison, 1210 W. Dayton Street, Madison, WI 53706, USA.

Definition 1 *Let A and B be two sets, and let H be a family of functions from A to B. H is called a* universal family of hash functions *if for every $x_1 \neq x_2 \in A$ and $y_1, y_2 \in B$ we have that*

$$\mathrm{Prob}_{h \in H}[h(x_1) = y_1 \ \wedge \ h(x_2) = y_2] = 1/|B|^2.$$

Apart from the value of such families of hash function for various hashing purposes as proposed in [9], many new applications have been found. For example, Sipser [20] used universal hashing to obtain simulation of BPP in the polynomial time hierarchy; Goldwasser and Sipser [13] used it for the simulation of Interactive Proof systems by Arthur-Merlin games; Impagliazzo and Zuckerman [14] used universal hashing for the amplification of the probability of success of randomized algorithms; and Nisan [18] used universal hashing to construct pseudorandom generators for space-bounded computation.

Previous research has been directed towards finding efficient implementation of universal hashing, as well as applying universal hashing in varying applications. Perhaps the only exception is Siegel [19], who studied both upper and lower bounds on the tradeoffs between the storage requirement and the number of random bits versus the computation time of $\log n$-wise independent hash functions, in the algebraic RAM model.

In this paper, we study the inherent *computational complexity* of universal hashing schemes. We give several lower bounds on the computational complexity that any implementation of a universal family of hash functions must incur. Our main result is a tight, quadratic, time-space tradeoff on the general model of boolean branching programs.

Borodin and Cook [6] defined and justified the boolean branching program as a general model of computation. It imposes no structure on the computation, allows any pattern of access to the input bits, and also allows the output bits to be computed in any order. The model is strong enough to efficiently simulate any other reasonable model of computation, such as multitape Turing machines or RAMs. In Section 3, we recall the definition of the model, and the notions of time and space in this model.

With the exception of [10], first time-space tradeoffs were obtained on structured models of compuation. Tompa [23] contains a discussion of these results. In the case of non-structured models, Cobham [10] shows a time-space tradeoff for any computational device having one head-read only input tape. The boolean branching program removes this restriction to accessing the input bits. The first non-trivial time-space tradeoffs in this model were given by Borodin and Cook [6] who proved that sorting requires a nearly quadratic tradeoff. Subsequently, Abrahamson [1] and Yesha [25] proved tradeoffs for several algebraic problems, and Beame [3] proved a tight tradeoff for sorting and related problems. In contrast to earlier results that provide tradeoffs for specific problems, we prove tradeoffs for a whole family of problems.

Theorem 2 *Let H be a universal family of hash function $h : \{1 \ldots N\} \rightarrow \{1 \ldots M\}$. Any branching program which on input $x \in \{1 \ldots N\}$ and $h \in H$ outputs $h(x)$ requires a time-space tradeoff of $TS = \Omega(nm)$, where $n = \log N$ (is the length of the representation of x) and $m = \log M$ (is the length of the representation of the output, $h(x)$).*

There are universal famalies of hash functions for which Theorem 2 is tight. (See, for example, Corollary 3.) As an immediate consequence of this theorem we get several of the known time-space tradeoffs for algebraic problems. Two of these are stated in the corollaries below.

Corollary 3 ([1]) *In the boolean branching program model, performing the convolution[1] of an n-bit string with an $(n+m-1)$-bit string to obtain an m bit string requires $TS = \Omega(nm)$.*

Corollary 4 ([25]) *In the boolean branching program model, multiplying two $n \times n$ matrices over $GF(2)$ requires $TS = \Omega(n^3)$.*

The the lower bound in Corollary 3 is tight [1]. In contrast, the bound in Corollary 4 is not tight; a (better) tight bound has been established by Abrahamson [1].

Perhaps the most interesting corollary of our theorem regards the complexity of performing arithmetic in fields. Consider the field $GF(p)$ where p is some prime number. It is not difficult to see that performing addition in this field can be done in simultaneously linear time and logarithmic space; multiplication is, however, more difficult. If one wishes to perform multiplication easily, one may do it by changing the *representation* of the numbers in $GF(p)$; each element in the field will be represented by its discrete logarithm (according to some fixed generator). In this representation it is easy to perform multiplication in linear time and logarithmic space, but now addition

[1] Convolution is defined in Section 3.

is difficult. Our results show that this tradeoff is unavoidable in any non-redundant representation[2] [7] of the elements of $GF(p)$, the combination of these operations is difficult.

Theorem 5 *Let F be a finite field, and let $n = \lceil \log |F| \rceil$. Fix an arbitrary representation of the elements of F by n-bit strings. Then any branching program which takes $a, b, c \in F$ as inputs, and outputs $a + b * c$, requires a time-space tradeoff of $TS = \Omega(n^2)$.*

It is interesting to note that the fact that F is a field is essential. For certain values of n and a carefully chosen representation, it is possible to perform $(a + b * c) \bmod n$ in linear time and logarithmic space. (For example, let n be a product of many "small" distinct primes and choose the representation by the Chinese Remainder Theorem.)

This paper contains other lower bounds on the complexity of any implementation of universal hashing. We first consider area-time tradeoffs in the VLSI model, and prove the following AT^2 lower bound.

Theorem 6 *Let H be a universal family of hash function $h : \{1 \ldots N\} \to \{1 \ldots N\}$. Any VLSI implementation which on input $x \in \{1 \ldots N\}$ and $h \in H$ outputs $h(x)$ requires a area-time tradeoff of $AT^2 = \Omega(n^2)$, where $n = \log N$.*

Again, this theorem implies some of the known AT^2 lower bounds, and in particular Yao's result [24] giving a quadratic AT^2 lower bound for the computation of $a + b * c$, in any representation.

We then make the observation that universal hash functions must have high "average sensitivity", a fact which in conjunction with several known results gives the following lower bounds:

Theorem 7 *Let H be a universal family of hash function $h : \{1 \ldots N\} \to \{1 \ldots M\}$. Then there exists a function $h \in H$ such that*

1. *h requires exponential size constant depth circuits.*
2. *h requires $\Omega(\log n)$ time to compute on a CREW PRAM.*
3. *$\Omega(m)$ of the output bits of h requires $\Omega(n^2)$ boolean formula size (over the basis consisting of $\{\vee, \wedge, \neg\}$).*

[2] A non-redundant representation is one where each element of $GF(p)$ has a unique representation.

where $n = \log N$ is the input size, and $m = \log M$ is the output size.

Another interesting resource tradeoff is that between communication and space. A communication-space tradeoff problem was posed in Tiwari [22]. First results in this direction were obtained on the algebraic straight-line model of communicating processors by Lam, Tiwari, and Tompa [16]. They proved a tight communication-space tradeoff for convolution and matrix multiplication. Recently, Beame, Tompa, and Yan [4] have used our main lemma in order to prove a tradeoff between communication and space on a general model of communicating processors.

The rest of this paper is organized as following. Section 2 gives the formal definition of universal hash functions and their properties. Section 3 defines the branching program model. Section 4 gives the lower bound for branching programs. Section 5 gives the VLSI lower bounds. Section 6 gives the remaining lower bounds. Section 7 states some open problems.

2 Universal hash functions

Definition 8 (Carter-Wegman) *Let A and B be two sets, and let H be a family of functions from A to B. H is called a universal family of hash functions if for every $x_1 \neq x_2 \in A$ and $y_1, y_2 \in B$ we have that*

$$\mathrm{Prob}_{h \in H}[h(x_1) = y_1 \ \wedge \ h(x_2) = y_2] = 1/|B|^2.$$

There are several known efficient constructions of universal families of hash functions. We mention two especially interesting, well known families. Note that in both of these constructions the length of the description of a function in H is linear in sum of the lengths of the description of an element from the domain A and from the range B.

1. Let F be an arbitrary finite field and let $A = B = F$. The family H that consists of all linear functions on F is a universal family of hash functions. More specifically, $H = \{ax + b | a, b \in F\}$, and each function $h \in H$ is uniquely represented by two elements a, b.

2. Let $A = \{0,1\}^n$ and $B = \{0,1\}^m$. For $x \in \{0,1\}^n$ and $y \in \{0,1\}^{n+m-1}$ we define the convolution of y and x, $y \circ x$, to be the m-bit string z whose i'th bit is given by $z_i = \sum_{j=1}^{n} x_j y_{i+j-1} \pmod 2$. For two m-bit strings x and y let $x \oplus y$ denote the bitwise exclusive-or of the two strings. Then the following family is a universal family of hash functions

$$H = \{(a \circ x) \oplus b | a \in \{0,1\}^{n+m-1},\ b \in \{0,1\}^m\}$$

The fact that the first is a family of universal hash functions can be observed from the fact that for any $x_1 \neq x_2, y_1, y_2$, there is a unique solution for a and b.

The fact that the convolution is a universal hash function is slightly less immediate, and for completeness the proof is sketched below.

Claim 9 *The following is a universal family of hash functions*

$$H = \{(a \circ x) \oplus b | a \in \{0,1\}^{n+m-1},\ b \in \{0,1\}^m\}$$

Proof: It suffices to show that the size of the set

$$\{< a, b > |(a \circ x_1) \oplus b = y_1 \wedge (a \circ x_2) \oplus b = y_2\}$$

is independent of the choices of x_1, x_2, y_1, y_2. Fix $x_1 \neq x_2$ and y_1, y_2 and note that for any choice of a such that $a \circ (x_1 \oplus x_2) = y_1 \oplus y_2$, there is a unique choice of b such that $(a \circ x_1) \oplus b = y_1$ and $(a \circ x_2) \oplus b = y_2$ (and no choice of b for other values of a). Thus, it suffices to count the number of choices for a satisfying the above condition.

Let $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$, we have that $x \neq \vec{0}$. Consider the set $\{a | a \circ x = y\}$. It is the solution space of m equations. Since $x \neq \vec{0}$, the equations are independent, and thus it defines a hyperspace of dimension exactly $n - 1$ in Z_2^{n+m-1}. $\square$

3 Model of Branching Programs

Consider a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$. A *binary branching program* computing f is a rooted directed graph with each internal nonsink node labeled with an integer i in the range from 1 to n. Each nonsink node has outdegree two, and the two out-edges are labeled zero and one. Some edges are also labeled with a pair of the form (j, b), where $j \in \{1, 2, \ldots, m\}$ and $b \in \{0,1\}$. A *computation* of the branching program on an input $x_1, \ldots, x_n$, is a path in the graph specified by the following conditions: (i) the path starts at the root, (ii) at vertex v with label i, the outgoing edge with label x_i is selected, and (iii) the path terminates on arrival at a sink node. A computation produces as output the edge-labels associated with its path.

We require that if a pair (j, b) is produced as an output, then b equals the j-th bit of $f(x_1, x_2, \ldots, x_n)$. Note that every input defines a unique computation path.

The *time complexity* of a branching program is the length of its longest computation, and the *space complexity* of a branching program is the logarithm of the number of nodes in the graph. Every branching program, with time T and space S, can be transformed to a branching program whose graph is acyclic (no directed cycles) and leveled (the underline graph is leveled), and whose time is T and space is at most $S + \log T$ (see [8]). For this reason, we restrict our discussion to acyclic and leveled branching programs.

4 A Time-Space Tradeoff for Hash Functions

For the lower bound we use a version of the proof technique introduced by Borodin and Cook [6].

Definition 10 *Consider a function $f : \{0,1\}^l \rightarrow \{0,1\}^m$. Let $A \subset \{0,1\}^l$ be the set of all l-bit strings that contain the same substring on some fixed set of n bit positions. (Note that $| A |= 2^{l-n}$.) For $k \leq m$, let $B \subset \{0,1\}^m$ be the set of all m-bit strings that contain the same substring on some fixed set of k bit positions. (Note that $| B |= 2^{m-k}$.) If, for every such pair of sets A and B, upon choosing an input from A uniformly, the probability of getting an element of B as the output is bounded by $2^{-\alpha k+\beta}$, then the function f is said to have the* randomness property *with parameters n, α, and β.*

We show (following [6]) that a function that has the randomness property requires $TS = \Omega(\alpha nm)$.

The following claim is immediate from the randomness property.

Claim 11 *If a function f has the randomness property with parameters n, α, and β, then any branching program of depth at most n, when given a random input x, outputs $k \leq n$ bits of $f(x)$ correctly for at most $2^{-\alpha k+\beta}$ fraction of the inputs.*

Proof: Clearly, along any computation path, the branching program can read at most n inputs. Let $y_{i_1} \ldots y_{i_k}$ be the outputs produced along a particular path. (Note that these bits, i.e., their value and index, are completely determined by the path.) Since this output is based only on some n input bits, the randomness property of f ensures that for any k output bits, there is no fixed assignment that has probability more than $2^{-\alpha k+\beta}$ of being correct. Therefore, the correct output is produced with probability at most $2^{-\alpha k+\beta}$. $\square$

Lemma 12 *Any branching program that computes a function $f : \{0,1\}^l \rightarrow \{0,1\}^m$ that has the randomness property with parameters n, α, and β, requires $(S+\beta)(T+n) \geq \alpha nm$, or, when α and β are constants, $TS \geq \Omega(mn)$.*

Proof: Claim 11 implies that no n inputs determine the output completely. Therefore, $T \geq n$, and consequently, $S \geq \log n$. Without loss of generality we may assume that the branching program is leveled and acyclic. Partition the branching program into $\lceil T/n \rceil$ blocks such that each block, except possibly the last one, contains n consecutive levels. Each block consists of a number of disjoint branching "subprograms". Note that there are at most 2^S such subprograms. In addition, each input can be associated with a block (and a particular subprogram in that block) that produces at least $m/\lceil T/n \rceil$ ($\leq n$, distinct) outputs (for that input).

For each subprogram we count the number of inputs for which it produces at least $m/\lceil T/n \rceil$ distinct outputs. By Claim 11, any n-level branching program can produce at least $m/\lceil T/n \rceil$ ($\leq n$, distinct) outputs for at most $2^l 2^{-\alpha m/\lceil T/n \rceil+\beta}$ of the inputs. On the other hand, for each input there is some subprogram that outputs at least $m/\lceil T/n \rceil$ distinct outputs. This implies that the number of inputs (i.e. 2^l) is bounded by the sum, over all subprograms, the number of inputs for which that subprogram produces at least $m/\lceil T/n \rceil$ distinct outputs. Since the number of subprograms is bounded by 2^S,

$$2^l \leq 2^S (2^l 2^{-\alpha m/\lceil T/n \rceil+\beta}).$$

This implies $(S+\beta)(T+n) \geq \Omega(\alpha mn)$. $\square$

The construction, to this point, follows the general outline of [6], and appeared with some variations in [1, 25, 3]. We still have to show the main part of the argument, that the universal hash functions have the randomness property.

Consider the function $f(h,x) = h(x)$ where h is from a collection of universal hash functions. The following Lemma gives the main argument that f has the *randomness property* required for proving the lower bound. This lemma is a stronger variant of a lemma that appears in [18]. It is also interesting to compare this lemma to a result of Lindsey on the distribution of -1's and 1's in submatrices of Hadamard matrices [2, 12].

Lemma 13 *Let $H = \{h : I \to O\}$ be a collection of universal hash functions. Let $A \subset I$, $B \subset O$ and $C \subset H$. Then,*

$$|\mathrm{Prob}_{x\in A, h\in C}[h(x) \in B] - p| \le \sqrt{\frac{|H|}{|A||C|}p(1-p)}$$

where $p = |B|/|O|$.

Proof: Define $M_{h,x}$ to be 1 if $h(x) \in B$ and 0 otherwise. Then,

$$|\mathrm{Prob}_{x\in A, h\in C}[h(x) \in B] - p| =$$

$$|E_{h\in C}E_{x\in A}(M_{h,x} - p)|.$$

Clearly, using Cauchy-Schwartz inequality,

$$|E_{h\in C}E_{x\in A}(M_{h,x} - p)| \le \sqrt{E_{h\in C}\{E_{x\in A}(M_{h,x} - p)\}^2}.$$

Rather than averaging over $h \in C$, we want to average over $h \in H$. Clearly, this can not decrease the expectation by a factor of more than $|H|/|C|$. Hence,

$$\sqrt{E_{h\in C}\{E_{x\in A}(M_{h,x} - p)\}^2} \le \sqrt{\frac{|H|}{|C|}E_{h\in H}\{E_{x\in A}(M_{h,x} - p)\}^2}$$

We will concentrate on the expression on the right-hand side, and compute it exactly.

$$\begin{aligned}
E_{h\in H}\{E_{x\in A}(M_{h,x} - p)\}^2 &= E_{h\in H}\{(E_{x_1\in A}(M_{h,x_1} - p))(E_{x_2\in A}(M_{h,x_2} - p))\} \\
&= E_{h\in H}\{E_{x_1\in A}E_{x_2\in A}(M_{h,x_1}M_{h,x_2} - pM_{h,x_1} - pM_{h,x_2} + p^2)\} \\
&= E_{x_1\in A}E_{x_2\in A}E_{h\in H}(M_{h,x_1}M_{h,x_2} - pM_{h,x_1} - pM_{h,x_2} + p^2)
\end{aligned}$$

For any x, $E_{h\in H}M_{h,x} = p$, therefore $E_{x\in A}E_{h\in H}M_{h,x} = p$. Furthermore, $E_{x_1\in A}E_{x_2\in A}E_{h\in H}M_{h,x_1}M_{h,x_2}$ can be computed by evaluating $E_{h\in H}M_{h,x_1}M_{h,x_2}$ in the two cases discussed below:

Case I: $x_1 = x_2$**.** In this case $E_{h\in H}M_{h,x_1}M_{h,x_2} = E_{h\in H}M_{h,x_1} = p$. This case occurs with probability $1/|A|$.

Case II: $x_1 \neq x_2$**.** By the properties (Definition 1) of the universal hash functions, $E_{h\in H}M_{h,x_1}M_{h,x_2} = p^2$. This case occurs with probability $1 - 1/|A|$.

Therefore,

$$\begin{aligned} E_{x_1 \in A} E_{x_2 \in A} E_{h \in H} M_{h,x_1} M_{h,x_2} &= (1/|A|)p + (1 - 1/|A|)p^2 \\ &= (p - p^2)/|A| + p^2. \end{aligned}$$

Combining with the previous terms, we get

$$\begin{aligned} E_{h \in H} \{E_{x \in H}(M_{h,x} - p)\}^2 &= (p - p^2)/|A| + p^2 - 2p^2 + p^2 \\ &= (p - p^2)/|A| \end{aligned}$$

To conclude, we have shown that,

$$|\mathrm{Prob}_{x \in A, h \in C}[h(x) \in B] - p| \leq \sqrt{\frac{|H|}{|A||C|} p(1-p)}$$

□

It remains to establish a relationship between the above lemma and the *randomness property.*

Lemma 14 *Let $H = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a universal family of hash functions. Then, the function $f(h,x) = h(x)$ has the randomness property with parameters n, $\alpha = \frac{1}{2}$, and $\beta = 1$.*

Proof: We need to show that, given any n bits specifying the argument of f, (i.e., specifying h and x,) the probability that any k bits of $f(h,x) = h(x)$ take on a specified value (output) is at most $2^{-\alpha k}$. Assume that, of the n bits given to us, l_1 bits are from x, and the remaining $l_2 = n - l_1$ bits are from h. Let A be the set of all possible consistent extensions of the input, and C be the set of all possible extensions of the output. Then $|H|/|C| \leq 2^{l_2}$, and $1/|A| \leq 2^{-n} 2^{l_1}$. For $l_1 + l_2 \leq n$, Lemma 13 implies that the probability of obtaining a fixed assignment to any $k \leq m$ bits of the output is bounded by $\sqrt{2^{-k}(1 - 2^{-k})} \leq 2^{-k/2}$. This implies that f has the randomness property parameters n, $\frac{1}{2}$, and 1. □

Lemma 14 is stated for hash functions whose input set is $I = \{0,1\}^n$ and output set is $O = \{0,1\}^m$. The same results hold for any $I \subset \{0,1\}^n$ and for $O \subset \{0,1\}^m$ if $|O| \geq 2^{m-c}$ for some constant c. We did not resolve the case where the output set is sparse in $\{0,1\}^m$, e.g. $|O| = 2^{m/2}$.

Note that Lemma 14 does not use the full power of Lemma 13. Lemma 13 allows us to argue about *any* large subset, rather than subsets that are defined by fixing certain bits. This is an essential difference between our lower bound technique and previous techniques.

Combining Lemma 14 with Lemma 12, we prove our main result.

Theorem 2 *Let H be a universal family of hash function $h : \{1 \ldots N\} \rightarrow \{1 \ldots M\}$. Any branching program which on input $x \in \{1 \ldots N\}$ and $h \in H$ outputs $h(x)$ requires a time-space tradeoff of $TS = \Omega(nm)$, where $n = \log N$ is the length of the representation of x and $m = \log M$ is the length of the representation of the output, $h(x)$.*

We can now establish the following corollaries.

Corollary 3 *Performing the convolution of an n-bit string with an $(n + m - 1)$-bit string to obtain an m bit string requires $TS = \Omega(nm)$.*

Proof: Let f be a function producing m-bit results. Then, if $f(y)$ can be computed by a branching program with time T and space S than $f(y) \oplus b$ can be computed by a branching program with time at most $2m + T$ and space at most $2\log m + S$, where b is an m-bit constant. By Claim 9, $H = \{(a \circ x) \oplus b | a \in \{0,1\}^{n+m-1}, b \in \{0,1\}^m\}$ is a universal family of hash functions, and the function $f(a,b,x) = (a \circ x) \oplus b$ requires $TS = \Omega(nm)$. Therefore, the convolution $a \circ x$ $(= f(a,b,x) \oplus b)$ requires $TS = \Omega(nm)$. □

Corollary 4 *Multiplying two $n \times n$ matrices over $GF(2)$ requires $TS = \Omega(n^3)$.*

Proof: There is a reduction to matrix multiplication from the convolution of an $n + n^2 - 1$ bit string with an n bit string; using standard arguments, one can reduce the convolution to multiplying two $2n \times 2n$ matrices. □

We now restate Theorem 5. It gives a time-space tradeoff for computing $a + b * c$ over any finite field, provided that inputs are presented in a *succinct* representation.

Theorem 5 *Let F be a finite field, and let $n = \lceil \log |F| \rceil$. Fix an arbitrary representation of the elements of F by n-bit strings. Then any branching program which takes $a, b, c \in F$ as inputs, and outputs $a + b * c$, requires a time-space tradeoff of $TS = \Omega(n^2)$.*

Proof: This is immediate from the fact that $H = \{ax + b | a, b \in F\}$ is a universal family of hash functions. □

5 An Area-Time Tradeoff for VLSI

The model that we are considering is Thompson Grid Model (see [21]) of VLSI Chip Layout. The model assumes that a chip consists of a grid with evenly spaced vertical and horizontal tracks. Circuit nodes are placed on the intersection of vertical and horizontal tracks. The *area* of a chip is defined to be the the number of grid-points it contains.

Definition 15 *For a string y of n bits, let $y_{|n/2}$ denote the substring consisting of the first $n/2$ bits of y.*

Lemma 16 *Let H be a family of hash functions from $\{0,1\}^n$ to $\{0,1\}^n$ and $x_1, x_2, \ldots, x_k \in \{0,1\}^{n/2}$ be distinct elements. For $k < 2^{n/4}$, there is a function $h \in H$, such that $h(\vec{0}x_i)_{|n/2} \neq h(\vec{0}x_j)_{|n/2}$ for all $i \neq j$, where $\vec{0}$ is a string of $\frac{n}{2}$ zeros.*

Proof: For any $y, z \in \{0,1\}^{n/2}$, $y \neq z$, the definition of universal hash functions implies that

$$\text{Prob}_{h \in H}[h(\vec{0}y)_{|n/2} = h(\vec{0}z)_{|n/2}] = 2^{-n/2}.$$

In order to prove the lemma, choose uniformly a function $h \in H$. The probability that there is a pair x_i,x_j $(i \neq j)$ such that the $h(\vec{0}x_1)_{|n/2} = h(\vec{0}x_2)_{|n/2}$, is bounded by $k^2 2^{-n/2}$. Therefore, for $k < 2^{n/4}$, there is a function $h \in H$, such that $h(\vec{0}x_i)_{|n/2} \neq h(\vec{0}x_j)_{|n/2}$ for all $i \neq j$. $\square$

Using standard arguments of information transfer, we show the following lower bound.

Theorem 6 *Let H be a universal family of hash function $h : \{1 \ldots N\} \to \{1 \ldots N\}$. Any VLSI implementation which on input $x \in \{1 \ldots N\}$ and $h \in H$ outputs $h(x)$ requires a area-time tradeoff of $AT^2 = \Omega(n^2)$, where $n = \log N$.*

Proof: Partition the chip into two parts with $O(\sqrt{A})$ bisection length, each having (approximately) half the bits of the input. Without loss of generality, assume that the first part contains the first $n/2$ bits of the input, as well as the first $n/2$ bits of the output. For all $1 \leq i \leq n/2$ fix $x_i = 0$. Consider the set of those inputs that have a string of zeros as the first $n/2$ positions. By Lemma 16, there is a function $h \in H$ such that $|\{h(\vec{0}x)_{|n/2} : x \in \{0,1\}^{n/2}\}| \geq 2^{n/4}$. Therefore the information transfer between the two parts must be $\Omega(n)$. We complete the proof using the well known results of [21]: In order to allow the transfer of $\Omega(n)$ bits through the bisection of size $O(\sqrt{A})$, the time is at least $\Omega(n/\sqrt{A})$ which implies that $AT^2 = \Omega(n^2)$. $\Box$

As stated, we assumed that the hashing was performed on the integers $1 \ldots N$. Following [24], one can ask how the results differ if we allow the inputs and outputs to be represented by arbitrary (but unique) numbers. For this question we obtain essentially the same tradeoffs as in [24]. In the case that the representation is $O(\log |F|)$ bits long, where $|F|$ is the number of elements in F, our results still hold (using, essentially, the same proof). However, when the representation is allowed to be sparse, we can only prove bounds when the inputs are all on the boundary of the chip.

Theorem 17 *Let $H = \{h : I \to O\}$ be a universal family of hash functions. Any VLSI implementation that receives as inputs h and x on the boundary of the circuit, and outputs $h(x)$ requires $AT^2 = \Omega(mn)$, where $n = \log |I|$ and $m = \log |O|$.*

Proof: Let the circuit be a $w \times d$ rectangle, where $w \geq d$. Assume that the representation of the input is l bits long. Since all the inputs appear on the boundary of the circuit, $wT \geq l/4$.

Divide the circuit into $2l/n$ subrectangles, along the edge w, such that every subrectangle has at most $n/2$ inputs. Following the same line of argument as in Theorem 6, we show that there is a subrectangle, such that if we fix its inputs to zero, there is a function $h \in H$ that has $2^{cmn/l}$ distinct outputs in this subrectangle, for some constant c. This implies that $dT = \Omega(mn/l)$. Since $A = wd$, then $AT^2 = (wT)(dT) = (l/4)(mn/l) = \Omega(mn)$. $\Box$

Using the same argument for a general chip, one can derive an $A^2T^3 = \Omega(m^2 n)$ lower bound.

6 Other Lower bounds

Definition 18 *Let $f(x_1, \ldots x_n)$ be a boolean function. The influence of x_i on f is*

$$\mathrm{Inf}_i(f) = \mathrm{Prob}\left[f(x_1, ..., x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, ..., x_{i-1}, 1, x_{i+1}, \ldots, x_n)\right]$$

Where the probability is uniform over all choices of $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. The average sensitivity of f is $s(f) = \sum_i \mathrm{Inf}_i(f)$.

Lemma 19 *Let H be a universal family of hash function $h : \{1 \dots N\} \rightarrow \{1 \dots M\}$, and denote by h_i the i'th output bit of h. Then there exists a function $h \in H$ such that for at least $m/4$ bits i: $s(h_i) \geq n/4$.*

Proof: Fix any bit $1 \leq i \leq m$. By the definition of universal hashing it is clear that for any $1 \leq j \leq n$: $E_{h \in H}[\mathrm{Inf}_j(h_i)] = 1/2$. We thus have that $E_{h \in H}[s(h_i)] = n/2$. The Markov inequality implies that for at least a quarter of the functions $h \in H$, $s(h_i) \geq n/4$. An averaging argument completes the proof of the lemma. □

Using known results, we get several bounds.

Theorem 7 *Let H be a universal family of hash function $h : \{1 \dots N\} \rightarrow \{1 \dots M\}$. Then there exists a function $h \in H$ such that*

1. *h requires exponential size constant depth circuits.*
2. *h requires $\Omega(\log n)$ time to compute on a CREW PRAM.*
3. *$\Omega(m)$ of the output bits of h requires $\Omega(n^2)$ boolean formula size (over the basis consisting of $\{\vee, \wedge, \neg\}$).*

where $n = \log N$ is the input size, and $m = \log M$ is the output size.

Proof: [17] shows that $exp(s(f)^{1/d})$ is a lower bound on the size of depth d circuits computing f. [11] shows that $\log s(f)$ is a lower bound on the CREW parallel time. (In fact, they give the bound in terms of the stronger worst case sensitivity.) Krapchenko's lower bound technique [15] shows that $s(f)^2$ is a lower bound on the boolean formula complexity. (See also [5]). □

7 Open problems

We would like to conjecture about the complexity of universal hash function in the circuit and Turing machine models. We conjecture that any implementation of universal hashing has superlinear circuit and Turing machine complexity.

Conjecture 20 *Let H be a universal family of hash function $h : \{1 \dots N\} \rightarrow \{1 \dots N\}$.*

1. *Any boolean circuit which on input $x \in \{1 \dots N\}$ and $h \in H$ outputs $h(x)$ requires $\Omega(n \log n)$ size.*

2. *Any multitape Turing machine which on input $x \in \{1 \ldots N\}$ and $h \in H$ outputs $h(x)$ requires $\Omega(n \log n)$ time.*

Where $n = \log N$ is the length of the representation of x.

Acknowledgment

We would like to thank Paul Beame, Martin Tompa, and Avi Wigderson for helpful discussions.

References

[1] K. Abrahamson. Time-space tradeoffs for branching programs constructed with those for straight line programs. In 27^{th} *Annual Symposium on Foundations of Computer Science, Toronto, Ontario, Canada*, pages 402–409, October 1986.

[2] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 337–347, 1986.

[3] P. Beame. A general sequential time space tradeoff for finding unique elements. In *Proceedings of the* 21^{st} *Annual ACM Symposium on Theory of Computing, Seattle, Washington*, pages 197–203, May 1989.

[4] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs in the boolean model. manuscript.

[5] R. Boppana and M. Sipser. The complexity of finite functions. Handbook of Theoretical Computer Science.

[6] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 1(2):287–297, 1982.

[7] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58:113–136, 1983.

[8] A. Borodin, M. Fischer, D. Kirkpatrik, N. Lynch, and M. Tompa. A time-space tradeoff for sorting on non-oblivious machines. *J. Comp. and Syst. Sci.*, 22(3):351–364, October 1981.

[9] L. Carter and M. Wegman. Universal hash functions. *J. Comp. and Syst. Sci.*, 18(2):143–154, 1979.

[10] A. Cobham. The recognition problem for the set of perfect squares, conference record. In *IEEE 7th Annual Symposium on Switching and Automata Theory*, pages 78–87, 1966.

[11] S. Cook, C. Dwork, and R. Reischuk. Upper and lower bounds for parallel random access machines without simultaneous writes. *J. Comp. and Syst. Sci.*, February 1986.

[12] P. Erdös and J. Spencer. *Probabilistic Methods in Combinatorics*. Academic Press, 1974.

[13] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the* 18^{th} *Annual ACM Symposium on Theory of Computing, Berkeley, California*, pages 59–68, May 1986.

[14] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In 30^{th} *Annual Symposium on Foundations of Computer Science, Reseach Triangle Park, NC*, pages 248–253, October 1989.

[15] V. Krapchenko. A method of determining lower bounds for the complexity of π schemes. *Math. Notes Acad. Sci. USSR*, pages 474–479, 1971.

[16] T. Lam, P. Tiwari, and M. Tompa. Tradeoffs between communication and space. In *Proceedings of the* 21^{st} *Annual ACM Symposium on Theory of Computing, Seattle, Washington*, pages 217–226, May 1989. (To appear in JCSS.).

[17] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform and learnability. In 30^{th} *Annual Symposium on Foundations of Computer Science, Reseach Triangle Park, NC*, pages 574–579, October 1989.

[18] N. Nisan. Pseudorandom generators for space-bounded computation. In *Proceedings of the* 22^{st} *Annual ACM Symposium on Theory of Computing, Baltimore, Maryland*, pages 204–212, May 1990.

[19] A. Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In 30^{th} *Annual Symposium on Foundations of Computer Science, Reseach Triangle Park, NC*, pages 20–25, October 1989.

[20] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the* 15^{th} *Annual ACM Symposium on Theory of Computing, Boston, Massachusetts*, pages 330–335, May 1983.

[21] Thompson. A complexity theory for VLSI. Technical report, Carnegie Mellon University, 1980. Ph.D. thesis, Dept. of Computer Science.

[22] P. Tiwari. *The communication complexity of distributed computing and a parallel algorithm for polynomial roots.* PhD thesis, University of Illinois at Urbana-Champaign, 1986.

[23] M. Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. *Journal of Comput. Syst. and Sci.*, 20:118–132, 1980.

[24] A. C. Yao. The entropic limitations on VLSI computations. In *Proceedings of the* 13^{th} *Annual ACM Symposium on Theory of Computing, Milwaukee, Wisconsin*, pages 308–311, May 1981.

[25] Y. Yesha. A time-space tradeoff for matrix multiplication and the discrete fourier transform on a general sequential random access computer. *J. Comp. and Syst. Sci.*, 29:183–197, 1984.