# ON THRESHOLD CIRCUITS AND POLYNOMIAL COMPUTATION *

JOHN H. REIF† AND STEPHEN R. TATE†

**Abstract.** A *Threshold Circuit* consists of an acyclic digraph of unbounded fanin, where each node computes a threshold function or its negation. This paper investigates the computational power of Threshold Circuits. A surprising relationship is uncovered between Threshold Circuits and another class of unbounded fanin circuits which are denoted Finite Field $Z_{P(n)}$ Circuits, where each node computes either multiple sums or products of integers modulo a prime $P(n)$. In particular, it is proved that all functions computed by Threshold Circuits of size $S(n) \geq n$ and depth $D(n)$ can also be computed by $Z_{P(n)}$ Circuits of size $O(S(n)\log S(n) + nP(n)\log P(n))$ and depth $O(D(n))$. Furthermore, it is shown that all functions computed by $Z_{P(n)}$ Circuits of size $S(n)$ and depth $D(n)$ can be computed by Threshold Circuits of size $O(\frac{1}{\epsilon^2}(S(n)\log P(n))^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^5}D(n))$. These are the main results of this paper.

There are many useful and quite surprising consequences of this result. For example, integer reciprocal can be computed in size $n^{O(1)}$ and depth $O(1)$. More generally, any analytic function with a convergent rational polynomial power series (such as sine, cosine, exponentiation, square root, and logarithm) can be computed within accuracy $2^{-n^c}$, for any constant $c$, by Threshold Circuits of polynomial size and constant depth. In addition, integer and polynomial division, FFT, polynomial interpolation, Chinese Remaindering, all the elementary symmetric functions, banded matrix inverse, and triangular Toeplitz matrix inverse can be exactly computed by Threshold Circuits of polynomial size and constant depth. All these results and simulations hold for polytime uniform circuits. This paper also gives a corresponding simulation of logspace uniform $Z_{P(n)}$ Circuits by logspace uniform Threshold Circuits requiring an additional multiplying factor of $O(\log\log\log P(n))$ depth.

Finally, purely algebraic methods for lower bounds for $Z_{P(n)}$ Circuits are developed. Using degree arguments, a Depth Hierarchy Theorem for $Z_{P(n)}$ Circuits is proved: for any $S(n) \geq n$, $D(n) = O(S(n)^{c'})$ for some constant $c' < 1$, and prime $P(n)$ where $6(S(n)/D(n))^{D(n)} < P(n) \leq 2^n$, there exists explicitly constructible functions computable by $Z_{P(n)}$ Circuits of size $S(n)$ and depth $D(n)$, but provably not computable by $Z_{P(n)}$ Circuits of size $S(n)^c$ and depth $o(D(n))$ for any constant $c \geq 1$.

**Key words.** circuit complexity, threshold circuits, finite field circuits

**AMS(MOS) subject classification.** 68Q25, 68Q40

## 1. Introduction.

A *threshold$_k$ function* is a boolean function whose output is 1 depending on whether at least $k$ of its inputs have value 1. For example, a *threshold$_5$ function* is defined to be 1 if at least 5 inputs are 1. A *Threshold Circuit* is a boolean circuit in which each node computes a threshold function or its negation, and the nodes have unbounded fanin.

Many basic physical devices such as transistors and neurons can be modeled as threshold devices. Since an individual neuron may have very high fanin, a Threshold Circuit is a natural model for a neural net. For reasons described below, we will be particularly concerned with bounded depth Threshold Circuits.

Certainly any massively parallel computing device that uses a large number of relatively slow components must have small computational depth on a given computation if the overall computation is to be fast. For example, the reaction time of the lower brain for many nontrivial behavioral and recognition responses is less than .5 seconds, whereas the synapse response time of most neurons of the brain is at least .005 seconds; therefore, the depth of these particular computations can be no more

* This research was supported by National Science Foundation Grant CCR-8696134, by a grant from the Office of Naval Research under contract ONR-N00014-87-K-0310, and by Air Force contract AFSOR-87-0386.

† Department of Computer Science, Duke University, Durham, North Carolina 27706.

than 100. Nevertheless, in this small depth, many nontrivial functions are computed by the brain. Minsky and Papert were among the first investigators to observe the relationship between the lower brain and constant depth Threshold Circuits [15]. In particular, they developed a model for a learning device, known as a Perceptron, which is essentially a threshold circuit with constant depth.

There has been a considerable amount of renewed interest in models for the brain and for learning, and many of the recently proposed models are again essentially constant depth Threshold Circuits. Examples of these models include the Connectionist Models [5] and the Boltzmann Machine [1, 10]. Recently, Parberry and Schnitger proved that Boltzmann Machines can be simulated by constant depth Threshold Circuits [16].

This paper is a further theoretical investigation of bounded depth Threshold Circuits. In particular, we consider the following fundamental computational question: *What class of functions can be computed by bounded depth Threshold Circuits?*

This paper is organized as follows: In § 2, we give definitions of Threshold and Finite Field Circuits. In § 3, we give a precise statement of our results. In § 4, we give a simulation of Threshold Circuits by Finite Field Circuits. In § 5, we give simulations of polytime uniform Finite Field Circuits by polytime uniform Threshold Circuits, thus characterizing the functions computed by Threshold Circuits of depth $D(n)$ as a certain class of multivariate polynomial functions computed by Finite Field Circuits of depth $\Theta(D(n))$. In § 6, we give similar simulation results for logspace constructible circuits. In § 7 we prove a Hierarchy Theorem for size bounded Finite Field Circuits with increasing depth. In § 8, we conclude the paper with some open problems, conjectures, and some comments on how our theoretical results on Threshold Circuits might be applied to the construction of parallel arithmetic VLSI chips and to biological studies of learning in neuron nets by interpolation.

## 2. Circuit Definitions.

### 2.1. Circuits that Compute Boolean Functions.

Fix a value domain $\Sigma$. A *function basis* is a set $F$ of functions over domain $\Sigma^k$, for each $k \geq 0$. We assume a binary decoding function $decode_{n,n'} : \{0,1\}^n \rightarrow \Sigma^{n'}$ for decoding length $n$ binary strings into $n'$ values in $\Sigma$, and an encoding function $encode_{m',m} : \Sigma^{m'} \rightarrow \{0,1\}^m$, for binary encoding vectors of $m'$ values in $\Sigma$ into binary strings of length $m$. We will define circuits which take $n$ binary values as input, decode these inputs to an $n'$-tuple of values in $\Sigma$, make a computation using the functions in $F$, and then encode the outputs in binary.

A *circuit* $C_n$ over function basis $F$ is an oriented, acyclic digraph with a list of *input nodes* $v_1, \cdots, v_{n'}$, a list of *output nodes* $u_1, \cdots, u_{m'}$, and a $k$-adic function in $F$ labeling each noninput node with fanin $k \geq 0$. Given a binary input string $(x_1, \cdots, x_n) \in \{0,1\}^n$ we decode the input as $decode_{n,n'}(x_1, \cdots, x_n) = (y_1, \cdots, y_{n'})$ where $(y_1, \cdots, y_{n'}) \in \Sigma^{n'}$, and assign each input node $v_i$ a value $val(v_i) = y_i \in \Sigma$, for $i = 1, \cdots, n'$. For each other node $w$, with say $k$ predecessors $w_1, \cdots, w_k$, we recursively assign $w$ a value $val(w) = f(val(w_1), \cdots, val(w_k)) \in \Sigma$, where $f \in F$ is the $k$-adic function that labels node $w$. $C_n$ finally outputs the binary string given by $encode_{m',m}(val(u_1), \cdots, val(u_{m'})) \in \{0,1\}^m$ (where the output length $m$ is fixed for the circuit $C_n$). Thus $C_n$ computes a boolean function from $\{0,1\}^n$ to $\{0,1\}^m$.

We shall allow the circuits considered in this paper to have arbitrary fanin. The *size* of circuit $C_n$ is the number of edges of the circuit. The *depth* of circuit $C_n$ is the length of the longest path from any input node to an output node. A *circuit family* is an infinite list of circuits $\mathbf{C} = (C_1, C_2, \cdots, C_n, \cdots)$ where $C_n$ has $n$ binary inputs. $\mathbf{C}$ computes a family of boolean functions $(f_1, f_2, \cdots, f_n, \cdots)$, where $f_n$ is the function of $n$ binary inputs computed by circuit $C_n$. Let $\mathbf{C}$ have *size complexity* $S(n)$ and simultaneous *depth complexity* $D(n)$ if, $\forall n \geq 0$, circuit $C_n$ has size $\leq S(n)$ and depth $\leq D(n)$.

Circuit family $\mathbf{C}$ is *polytime* (*logspace*) *uniform* if there exists a Turing machine $M$ with $n^{O(1)}$ time bound ($O(\log n)$ space bound, respectively), such that given any $n \geq 1$ in unary, $M$ constructs an encoding of circuit $C_n$.

**2.2. Threshold Circuits.** A threshold function is a boolean function denoted $\delta_{k,\Delta} : \{0,1\}^k \rightarrow \{0,1\}$ such that

$$\delta_{k,\Delta}(x_1, \cdots, x_k) = \begin{cases} 1 & \text{if } \sum_{i=1}^{k} x_i \geq \Delta \\ 0 & \text{otherwise} \end{cases}$$

for $x_1, \cdots, x_k \in \{0,1\}$. Let $Th$ denote the set of all threshold functions and their negations. A *Threshold Circuit* is a circuit with function basis $Th$. Note that in the case of Threshold Circuits the value domain is $\Sigma = \{0,1\}$, so the number of input nodes is always the same as the number of boolean inputs, and *decode* and *encode* are simply the identity functions (i.e., no decoding of inputs or encoding of outputs is required). We let $Th(S(n), D(n))$ denote the collection of boolean function families computed by polytime uniform Threshold Circuits of size $O(S(n))$ and simultaneous depth $O(D(n))$. In addition, we will use the notation (logspace uniform) $Th(S(n), D(n))$ to denote the corresponding function families computed by logspace uniform Threshold Circuits. Note that with this notation, the class of all functions computed by Threshold Circuits having polynomial size and constant depth is $Th(n^{O(1)}, 1)$.

**2.3. Finite Field Circuits.** Let $p$ be a prime number. For finite field circuits, the value domain $\Sigma$ is $Z_p$, the finite field modulo $p$. We will let $FZ_p$ denote the set of functions consisting of $k$-adic addition and multiplication taken modulo $p$ for each $k \geq 1$, as well as a constant function giving value $y$, for each $y \in Z_p$. A (Finite Field) $Z_p$ Circuit $C_n$ is a circuit over function basis $FZ_p$. Let $b = \lfloor \log p \rfloor$. Given binary inputs $x_1, \cdots, x_n \in \{0,1\}$, we decode these inputs into $n' = \lceil n/b \rceil$ integer values $decode_{n,n'}(x_1, \cdots, x_n) = (y_1, \cdots, y_{n'})$, where the value $y_i \in Z_{2^b}$ is the number with binary encoding $x_{(i-1)b+1}, x_{(i-1)b+2}, \cdots, x_{\min(n,ib)}$. Note that the decoding of binary inputs yields only numbers in the range $\{0, 1, \cdots, 2^b - 1\} \subseteq Z_p$. The circuit $C_n$ then makes a computation over $FZ_p$ as described in § 2.1. If $u_1, \cdots, u_{m'}$ are the output nodes, then we encode the output as $encode_{m',m}(val(u_1), \cdots, val(u_{m'})) = B_1 \cdots B_{m'}$, where $B_i$ is the $t_i = \min(m - b(i-1), b)$ bit binary encoding of the integer residue of $val(u_i) \bmod 2^{t_i}$. We let $Z_{P(n)}(S(n), D(n))$ denote the collection of boolean function families computed by polytime uniform $Z_{P(n)}$ Circuit families $\mathbf{C} = (C_1, C_2, \cdots, C_n, \cdots)$ where each $C_n$ is a $Z_{P(n)}$ Circuit with size $O(S(n))$ and simultaneous depth $O(D(n))$. We will use the additional notation (logspace uniform) $Z_{P(n)}(S(n), D(n))$ to denote the corresponding function families computed by logspace uniform $Z_{P(n)}$ Circuits.

**3. Statement of Results.** In the following we let $P(n)$, $S(n)$, and $D(n)$ be any positive functions of $n$ such that $S(n) \geq n$, and $P(n)$ is prime for all $n$.

We will first give a simulation of (polytime uniform) Threshold Circuits by (polytime uniform) Finite Field Circuits:

THEOREM 3.1. *If $S(n) \le P(n) \le n^{O(1)}$ for all $n$, then*

$$Th(S(n), D(n)) \subseteq Z_{P(n)}(S(n)\log S(n) + nP(n)\log P(n), D(n)).$$

*Note:* Theorem 3.1 also holds for logspace uniform circuits.

Next we will give a simulation of (polytime uniform) Finite Field Circuits by (polytime uniform) Threshold Circuits:

THEOREM 3.2. $Z_{P(n)}(S(n), D(n)) \subseteq Th(\frac{1}{\epsilon^2}(S(n)\log P(n))^{1+\epsilon}, \frac{1}{\epsilon^5}D(n))$

The proof of theorem 3.2 requires that we build up families of Threshold Circuits for the basic problems of multiplication, iterated sum, and iterated product. The most costly problem we encounter is iterated product, and this is solved using techniques introduced for integer division [2, 8, 18].

As a consequence of Theorem 3.2, we have

COROLLARY 3.3. *Suppose an analytic function $f(x)$ has a convergent Taylor Series Expansion of form*

$$f(x) = \sum_{n=0}^{\infty} c_n(x - x_0)^n$$

*over an interval $|x - x_0| \le \epsilon$ where $0 < \epsilon < 1$, and the coefficients are rationals $c_n = \frac{a_n}{b_n}$ where $a_n, b_n$ are integers of magnitude $\le 2^{n^{O(1)}}$. Then polytime uniform Threshold Circuits of polynomial size and simultaneous constant depth (i.e., a function in $Th(n^{O(1)}, 1)$) can compute $f(x)$ over this interval within accuracy $2^{-n^c}$ for any constant $c \ge 1$.*

Note that Corollary 3.3 follows directly from Theorem 3.2 since a Finite Field $Z_{P(n)}$ Circuit of size $n^{O(1)}$ and depth $O(1)$ with $P(n) = 2^{n^{O(1)}}$ can simulate the rational arithmetic required to approximately evaluate $f(x)$.

Corollary 3.3 implies (see [18]) that $Th(n^{O(1)}, 1)$ contains a surprisingly rich class of elementary functions (which can be computed within accuracy $2^{-n^c}$) including: integer reciprocal, sine, cosine, exponential, logarithm, and square root, as well as exact computation of the following:

1. integer and polynomial quotient and remainder,
2. interpolation of rational polynomials,
3. banded matrix inverse, and
4. triangular Toeplitz matrix inverse.

These problems can all be efficiently reduced to integer products; also see [3, 4, 12, 18]. Theorems 3.1 and 3.2 yield the characterization:

COROLLARY 3.4. *For $S(n) \le P(n) \le n^{O(1)}$,*

$$\bigcup_{c \ge 1} Z_{P(n)}(S(n)^c, D(n)) = \bigcup_{c \ge 1} Th(S(n)^c, D(n))$$

For example, for $S(n) = n^{O(1)}$, $D(n) = O(1)$, $P(n) \le n^{O(1)}$, we get

$$Z_{P(n)}(n^{O(1)}, 1) = Th(n^{O(1)}, 1)$$

In other words, the class of functions computed by polytime uniform $Z_{P(n)}$ Circuits of polynomial size and constant depth is exactly the same as the class of functions computed by polytime uniform Threshold Circuits of polynomial size and constant depth.

Next, we will give a simulation of logspace uniform Finite Field Circuits by logspace uniform Threshold Circuits.

THEOREM 3.5.

$$\begin{aligned}&(\textit{logspace uniform})\ Z_{P(n)}(S(n), D(n)) \subseteq \\ &\qquad (\textit{logspace uniform})\ Th((S(n)\log(P(n)))^{O(1)}, D(n)\log\log\log P(n)).\end{aligned}$$

The proof of Theorem 3.5 uses techniques developed by Reif for integer division by uniform boolean circuits of bounded fanin, polynomial size, and $O(\log n \log\log n)$ depth [18]. Theorem 3.5 implies that (logspace uniform) $Th(n^{O(1)}, \log\log n)$ contains the various elementary functions listed above.

Finally, we derive some lower bound results for Finite Field Circuits using algebraic degree arguments.

THEOREM 3.6. *If $D(n) = O(S(n)^{c'})$ for some constant $c' < 1$, $D'(n) = o(D(n))$, and $6(S(n)/D(n))^{D(n)} < P(n) \le 2^n$, then there exists a function in $Z_{P(n)}(S(n), D(n))$ which is not in $\bigcup_{c\ge 1} Z_{P(n)}(S(n)^c, D'(n))$.*

Previously, Kung showed that degree bounded polynomials formed a hierarchy [13], but this did not immediately imply our result for $Z_{P(n)}$ circuits.

**4. Simulation of Threshold Circuits by Finite Field Circuits.** The key to our simulations will be the following:

LEMMA 4.1. *For prime $p$, and any function $f : Z_p \to Z_p$ there is a (polytime and logspace uniform) $Z_p$ Circuit of size $O(p\log p)$ and depth $O(1)$ which computes $f$.*

*Proof.* Any function $f : Z_p \to Z_p$ can be interpolated within $Z_p$ at all $p$ of its inputs, yielding a degree $p-1$ polynomial $p(x) = \sum_{i=0}^{p-1} c_i x^i$. In $O(p)$ size and O(1) depth, we can compute $x^{2^j}$ for $j = 0, \cdots, \lfloor \log p \rfloor$. From these values we can compute $x^i$ for each $i = 1, \cdots, p-1$ in $O(p \log p)$ size and O(1) depth. It follows that $f(x)$ is computable by a $Z_p$ circuit of size $O(p\log p)$ and depth $O(1)$. □

**4.1. Proof of Theorem 3.1.** Let $C_n$ be a polytime uniform Threshold Circuit of $n$ binary inputs $(x_1, \cdots, x_n) \in \{0,1\}^n$, where $C_n$ has size $S(n)$ and depth $D(n)$. For any prime $p = P(n) \ge S(n)$, we will construct a $Z_p$ Circuit $C'_n$ which will also take $n$ binary inputs $(x_1, \cdots, x_n) \in \{0,1\}^n$. Let $b = \lfloor \log p \rfloor$. By definition (see § 2.3), $C'_n$ must have $n' = \lceil \frac{n}{b} \rceil$ input nodes $v_1, \cdots, v_{n'}$ which are assigned integers $val(v_1) = y_1, \cdots, val(v_{n'}) = y_{n'}$, where $decode_{n,n'}(x_1, \cdots, x_n) = (y_1, \cdots, y_{n'})$. The first difficulty we must overcome is to compute within $C'_n$ the boolean encoding $x_{(i-1)b+1}, x_{(i-1)b+2}, \cdots, x_{\min(n,ib)} \in \{0,1\}$ of each integer $y_i$ (i.e., these boolean values must be computed by $C'_n$ from the $y_i$ values using only addition and multiplication modulo $p$). By Lemma 4.1, there exists a polynomial $f_j(y)$ of degree $\le p-1$ which when evaluated in $Z_p$ gives the boolean value of the $jth$ bit of $y \in Z_p$, so each $x_{(i-1)b+j} = f_j(y_i)$ can be computed in $C'_n$ using size $O(p\log p)$ and depth $O(1)$. The total size required here is $O(np\log p)$.

Next we must simulate in $C'_n$ a threshold function $\delta_{k,\Delta}$ of $k$ binary inputs, say $a_1, \cdots, a_k \in \{0,1\}$. This can be done by first computing the sum $s = \sum_{i=1}^{k} a_i$, and then finding the interpolating polynomial of degree $k-1$ that computes the function

$$\lambda_\Delta(s) = \begin{cases} 1 & s \ge \Delta \\ 0 & s < \Delta \end{cases}.$$

This interpolating polynomial can be evaluated in size $O(k\log k)$ and depth $O(1)$. The negation of $\delta_{k,\Delta}$ can be computed in $Z_p$ by a similar application of Lemma 4.1. This simulation of the threshold computations of $C_n$ requires the $Z_p$ Circuit $C'_n$ to have size $O(S(n)\log S(n))$ and depth $O(D(n))$. Finally, if $C_n$ has (boolean valued) output nodes $u_1,\cdots,u_m$, then we let $C'_n$ have output nodes $u'_1,\cdots,u'_{m'}$ where $m'=\lceil\frac{m}{b}\rceil$. For $i=1,\cdots,m'$ we compute the values $val(u'_i)=\sum_{j=1}^{t_i}2^j val(u_{(i-1)b+j})$ where $t_i=\min(b,m-b(i-1))$, so $encode_{m',m}(val(u'_1),\cdots,val(u'_{m'}))=(val(u_1),\cdots,val(u_m))$, and the (boolean) function computed by $C'_n$ is exactly the same as the function computed by $C_n$. The constructed $Z_{P(n)}$ Circuit $C'_n$ has $O(S(n)\log S(n)+nP(n)\log P(n))$ size and $O(D(n))$ depth, and $C'_n$ is polynomial time constructible thus completing the proof of Theorem 3.1. □

Note that if $C_n$ is logspace uniform, then $C'_n$ is also logspace uniform.

## 5. Simulation of Finite Field Circuits by Threshold Circuits.

**5.1. Computing Arithmetic Using Polytime Constructible Threshold Circuits.** The problem of finding the sum of a set of numbers is called the *iterated sum problem*. Pippenger has given a constant depth threshold circuit for multiplication, and the method used is the straight-forward reduction to iterated sum (i.e., the "grade-school method" of multiplication) [17]. Looking at just the iterated sum circuit, we see that Pippenger's circuit for adding $m$ values, each of $n$ bits, has size $O(nm^2)$ and depth $O(1)$. In the following lemma, we show how to produce a constant depth circuit for iterated sum with smaller size.

LEMMA 5.1. *Given any constant $\epsilon$ satisfying $0<\epsilon\leq 1$, there exists a circuit for computing the iterated sum of $m$ numbers, each of $n$ bits, (with $m\leq n^{O(1)}$) that has size $O(nm^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon})$.*

*Proof.* Since $m\leq n^{O(1)}$, it is trivial to show that the result of the iterated sum will have less than $cn$ bits for some constant $c$.

To calculated the iterated sum, we build a computation tree with maximum fanout $\lfloor m^\epsilon\rfloor$ and $m$ leaves. Placing the $m$ input values at the leaves, computation proceeds toward the root of the tree with each internal node computing the sum of its children. After all computations, the root contains the sum of all $m$ input values. It is easy to see that the desired tree has $O(m^{1-\epsilon})$ internal nodes, and a height of $O(\frac{1}{\epsilon})$. We use Pippenger's circuit at each internal node for a node size of $O(nm^{2\epsilon})$, so the total circuit size is $O(nm^{1+\epsilon})$. Since the depth of each node in the tree is constant, the total depth of the circuit is the same as the height of the tree, or $O(\frac{1}{\epsilon})$. □

Using this result, we can also construct small size circuits for discrete Fourier transform. Let $\mathrm{DFT}_M$ denote the discrete Fourier transform of an $M$-vector.

LEMMA 5.2. *Given any constant $\epsilon$ satisfying $0<\epsilon\leq 1$, we can construct a circuit for $\mathrm{DFT}_M(a_0,a_1,\cdots,a_{M-1}) \bmod 2^N+1$ (where $M$ and $N$ are both powers of 2 and $M\leq N$) that has size $O(\frac{1}{\epsilon}MN^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^2})$.*

*Proof.* Since $N$ and $M$ are powers of 2, let $N=2^n$ and $M=2^m$. We will first show $\mathrm{DFT}_M$ exists in the ring $Z_{2^N+1}$. If we let $\omega=2^{2N/M}$, then by taking $\omega^{M/2}=2^N\equiv -1 \pmod{2^N+1}$ it is easy to see that $\omega$ is a principle $M$th root of unity in $Z_{2^N+1}$. Also, since $M$ is a power of 2, we know that $M$ and $2^N+1$ are relatively prime; therefore, $M^{-1}$ exists in the ring. By these facts, the ring $Z_{2^N+1}$ supports DFTs on $M$-vectors.

We introduce a new constant $\delta=\frac{\sqrt{1+4\epsilon}-1}{2}$. We will construct a computation tree as we did in Lemma 5.1, but the fanout in this case will be $f=2^{\lfloor m\delta\rfloor}$. Let $v_0,v_1,\cdots,v_{f-1}$ be the children of the root, and assume each child recursively computes the $\frac{M}{f}$-vector $val(v_i)=(x_{i,0},x_{i,1},\cdots,x_{i,M/f-1})=\mathrm{DFT}_{M/f}(a_i,a_{f+i},\cdots,a_{M-f+i})$.

Note that these vectors exist since $\omega^f$ is a principle $\frac{M}{f}$th root of unity, and $\left(\frac{M}{f}\right)^{-1}$ exists in $Z_{2^N+1}$. From these vectors we can produce the vector $(y_0, y_1, \cdots, y_{M-1}) = \mathrm{DFT}_M(a_0, a_1, \cdots, a_{M-1})$ by calculating

$$y_i = \sum_{j=0}^{f-1} \omega^j x_{j,i} \bmod 2^N + 1. \tag{1}$$

The proof of correctness for (1) is straight-forward, and is not included in this paper. Equation (1) is a simple modular iterated sum, since multiplication by powers of $\omega$ is just a bit shift of zero cost. This process is repeated down the tree until there are less than $f$ values in each node. In general, if we label the root as level 0, we are calculating $\mathrm{DFT}_{M/f^i}$ at each node of level $i$ from its $f$ children. By using the iterated sum circuit of Lemma 5.1 (the reduction mod $2^N+1$ can be done after a non-modular iterated sum with a single subtraction), we can do this in size $O(\frac{M}{f^i}Nf^{1+\delta})$ for each node on level $i$. Since there are $f^i$ nodes on level $i$, the total size for all nodes of that level is $O(MNf^{1+\delta})$. There are $O(\frac{1}{\delta})$ levels, so the total size of the circuit is $O(\frac{1}{\delta}MNf^{1+\delta})$. Since $f$ is $O(N^\delta)$, the size can be written as $O(\frac{1}{\delta}MN^{1+\delta+\delta^2}) = O(\frac{1}{\epsilon}MN^{1+\epsilon})$. The depth of each level is $O(\frac{1}{\delta})$, so the total depth is $O(\frac{1}{\delta^2}) = O(\frac{1}{\epsilon^2})$. □

Using this circuit for discrete Fourier transform we can construct a constant depth multiplication circuit.

LEMMA 5.3. *Given any constant $\epsilon$ satisfying $0 < \epsilon \le 0.6$, we can construct a circuit for multiplication of two $N$ bit numbers that has size $O(\frac{1}{\epsilon}N^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^3})$.*

*Proof.* The circuits that we construct are actually for multiplying two $N$-bit numbers modulo $2^N+1$, where $N$ is a power of 2. For exact (non-modular) multiplication of $N'$ bit numbers, we use the same circuit with $N = 2^{\lceil \log N' \rceil + 1}$. It is easy to show that this will produce the exact answer.

We will denote the two input numbers by $a$ and $b$, and their product by $c$. Since $N$ is a power of 2, let $N = 2^n$, where $n$ is an integer. Letting $m = 2^{\lfloor \epsilon n \rfloor}$, we can write any $N$-bit number $a$ as an $m$-vector of blocks of $s = \frac{N}{m}$ bits, $a = (a_0, a_1, \cdots, a_{m-1})$; $a_0$ is the block of least significant bits. We can view this vector as a vector of polynomial coefficients, and define the polynomial $A(x) = \sum_{i=0}^{m-1} a_i x^i$. Note that $A(2^s) = a$. Defining a polynomial for $b$ in a similar way, the product polynomial $C(x) = A(x)B(x)$ will be such that $C(2^s) = c$.

We use discrete Fourier transforms for the polynomial multiplication, and since the product polynomial will have degree $2m-2$, we must calculate the transform of $2m$-vectors. (We could actually use wrapped convolutions on $m$-vectors, but nothing is gained over our asymptotic bounds.) Looking at the straight-forward method of polynomial multiplication, it is easy to bound $\max_{0\le i<2m}\{c_i\} < m2^{2s} < m(2^{2s}+1)$. Since $m$ and $2^{2s}+1$ must be relatively prime, we can calculate the coefficients of $C(x)$ modulo both $m$ and $2^{2s}+1$, and combine these results for the final answer modulo $m(2^{2s}+1)$. This ring includes as a subset the range of all possible results, so the result of these modular calculations is also the exact (non-modular) answer. The calculations modulo $m$ can be done using Lemma 5.1 and "grade-school multiplication", with a total size of $O(N^{1+\epsilon})$ as long as $\epsilon \le 0.6$. We will now concentrate on the cost of the calculations modulo $2^{2s}+1$.

We will again use a divide and conquer tree with the root labeled as level 0. The fanout of the tree is $2m$, and it should be obvious that on level $i$ we are computing products of $s_i = N\left(\frac{2}{m}\right)^i$ bit numbers. The $\mathrm{DFT}_{2m} \bmod (2^{2s_{i+1}}+1)$ required at this level can be done in size $O(\frac{1}{\epsilon}2m(2s_{i+1})^{1+\epsilon})$ by Lemma 5.2. On level $i$, there are $(2m)^i$ such DFTs to calculate, for a total size of $O(\frac{1}{\epsilon}4^{i+1}\left(\frac{2}{m}\right)^{(i+1)\epsilon}(2N)^{1+\epsilon})$. For sufficiently large $N$ (and therefore $m$) we have $\left(\frac{m}{2}\right)^{\epsilon} > 8$, so the size of level $i$ can be simplified to $O(\frac{1}{\epsilon}\left(\frac{1}{2}\right)^i N^{1+\epsilon})$. Summing over all levels we have a total size of $O(\frac{1}{\epsilon}N^{1+\epsilon})$.

The depth of each level in the tree is $O(\frac{1}{\epsilon^2})$ by Lemma 5.2, so the total depth of our multiplication circuit is $O(\frac{1}{\epsilon^3})$. □

*Note:* The requirement that $\epsilon \leq 0.6$ can be relaxed to $\epsilon \leq 1$ by simply creating a new constant $\delta = \frac{\epsilon}{2}$ and absorbing the constant factor increase in depth into the big-Oh notation; however, this is clearly just a notational manipulation and not an algorithmic improvement.

The problem of Chinese Remaindering can be stated as follows: given $m$ small primes $p_1, p_2, \cdots, p_m$ (actually, they only have to be pairwise relatively prime) and an $n$ bit number $a$, calculate the residue of $a \bmod p_i$ for all $1 \leq i \leq m$. Conversely, given the residues modulo each of the primes $r_1, r_2, \cdots, r_m$, we would like to calculate the least positive $a$ such that $a \equiv r_i \pmod{p_i}$ for all $1 \leq i \leq m$. We will only be interested in the case where $m \geq n$, and this fact simplifies the analysis.

LEMMA 5.4. *Given any constant $\epsilon$ satisfying $0 < \epsilon \leq 0.6$, we can construct a circuit for Chinese Remaindering (in both directions) with size $O(\frac{1}{\epsilon^2}m^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^4})$.*

*Proof.* The method of Chinese Remaindering is taken straight from [8], using the multiplication circuit of Lemma 5.3. The proof of the size and depth of the circuit is also analogous to that found in [8], and is not included in this paper. □

The last basic problem we will look at is that of iterated product over a finite field. An iterated product of $m$ values $a_1, a_2, \cdots, a_m$ over the field $Z_p$ is defined to be $\prod_{i=1}^m a_i \bmod p$.

LEMMA 5.5. *Given any constant $\epsilon$ satisfying $0 < \epsilon \leq 1$, we can construct a circuit for iterated product of $m$ numbers over the field $Z_p$ with size $O(\frac{1}{\epsilon^2}(m\log p)^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^5})$.*

*Proof.* Define a new constant $\delta = \frac{\epsilon}{5}$. We will perform the iterated product in a tree similar to the tree used for iterated sum. The tree will have fanout $m^\delta$, and will perform an iterated product of $m^\delta$ values in $Z_p$ at each node. The iterated product at each node is computed by performing a Chinese Remainder step, followed by calculating the iterated product over each of the smaller fields (using discrete logs, iterated sum, and powering), and finally a Chinese Remainering step to recover the full result. This produces the exact iterated product, and by multiplying by an $m^\delta \log p$ bit approximation to $1/p$, we can find the residue modulo $p$.

To insure there is no loss of information, we must be sure that $\prod p_i$ is greater than the maximum possible result. Specifically, we must insure that $\prod_{i=1}^s p_i > p^{m^\delta}$. By basic number theoretic results, we can achieve this with $s \leq p_s = \Theta(m^\delta \log p)$. Obviously, $s > \log p$, so the condition of Lemma 5.4 is satisfied, and we may construct the required Chinese Remaindering circuit with size $O(\frac{1}{\delta^2}m^{2\delta}(\log p)^{1+\delta})$ and depth $O(\frac{1}{\delta^4})$.

After performing the initial Chinese Remaindering step, we must perform an iterated product over each of the $p_i$. Since for all prime $p_i$, $Z_{p_i}$ is a cyclic group, there is a (not necessarily unique) generator — call it $g_i$ — that generates the entire group. Let $f_i(x) = g_i^x$; due to the fact that $g_i$ is a generator, this function is one-to-one and onto over $Z_{p_i}^*$. We make tables for $f_i(x)$ and $f_i^{-1}(x)$, each of size $O(p_i \log p_i)$. Within a particular field, there must be tables for all $m^\delta$ input values, so the total size taken up by tables for $p_i$ is $O(m^\delta p_i \log p_i)$.

The iterated product is calculated by taking the discrete logarithm of all input values ($f_i^{-1}(x)$, above), performing the iterated sum of these values modulo $p_i - 1$, then raising the generator to the resulting power in $Z_{p_i}$ (this is just $f_i(x)$, above). This is a fairly common method of performing iterated product (see, for example, [2]). The only part we haven't examined here is the iterated sum. By Lemma 5.1, we can calculate the exact iterated sum of $m^\delta$ numbers, each of $\log p_i$ bits, in size $O(m^{\delta+\delta^2} \log p_i)$ and depth $O(\frac{1}{\delta})$. With an $m^\delta \log p_i$ bit approximation to $(1/(p_i - 1))$, we can reduce this exact result to the result modulo $p_i - 1$ with a single multiplication. By Lemma 5.3, this takes size $O(\frac{1}{\delta} m^{\delta+\delta^2} (\log p_i)^{1+\delta})$ and depth $O(\frac{1}{\delta^3})$. Therefore the total complexity of calculating the iterated product of $m^\delta$ numbers modulo $p_i$ is $O(\frac{1}{\delta^2} m^{2\delta} p_i (\log p_i)^{1+\delta})$ size and $O(\frac{1}{\delta^4})$ depth.

Since this must be done for all $s$ prime fields, the total size complexity of iterated product of $m^\delta$ numbers is $s$ times the above value, plus the cost of Chinese Remaindering. Using the upper bounds for $s$ and $p_i$, the total size is $O(\frac{1}{\delta} m^{5\delta} (\log p)^{1+2\delta})$, and the total depth is $O(\frac{1}{\delta^4})$. With an $m^\delta \log p$ bit approximation to $(1/p)$, we can reduce this result (the exact iterated product) modulo $p$. The complexity of this multiplication is negligible compared to the rest of the circuit.

All the above results are for one node of the tree. Summing over all nodes and rewriting in terms of $\epsilon$, the total size is $O(\frac{1}{\delta^2} m^{1+5\delta} (\log p)^{1+2\delta}) = O(\frac{1}{\epsilon^2} (m \log p)^{1+\epsilon})$, and the depth is $O(\frac{1}{\epsilon^5})$. $\square$

*Note:* All Threshold Circuit families considered in this section can easily be seen to be constructed in polynomial time.

**5.2. Proof of Theorem 3.2.** Now we are ready to prove Theorem 3.2. Consider any polytime uniform $Z_{P(n)}$ Circuit $C_n$ with size $S(n)$ and depth $D(n)$. We wish to simulate $C_n$ by a Threshold Circuit $\hat{C}_n$. We will precompute an $S(n) \log P(n)$ bit approximation of the reciprocal of $P(n)$ so that a residue computation modulo $P(n)$ node of fanin $k$ can be done by just $O(k)$ additions and multiplications on $O(\log P(n))$ bit binary numbers, followed by a residue computation using an $O(k \log P(n))$ bit approximation to $\frac{1}{P(n)}$; therefore, each iterated sum or iterated product required at a node of $C_n$ can be done by applying Lemmas 5.1 and 5.5 using only size $O(\frac{1}{\epsilon^2}(k \log P(n))^{1+\epsilon})$ and depth $O(\frac{1}{\epsilon^5})$. The total size of the Threshold Circuit $\hat{C}_n$ is $O(\frac{1}{\epsilon^2}(S(n) \log P(n))^{1+\epsilon})$, and the depth is $O(\frac{1}{\epsilon^5} D(n))$; furthermore, the circuit family $\hat{C}$ is constructible in polynomial time, completing the proof of Theorem 3.2. $\square$

**6. Log Space Uniform Threshold Circuit Simulation of Arithmetic and Finite Field Circuits.** Let $a_1, \cdots, a_m \in Z_{2^n}$. Let $D(m,n)$ be the depth required to compute $\prod_{i=1}^m a_i \bmod (2^n + 1)$ using a (logspace uniform) Threshold Circuit of size $(mn)^{O(1)}$.

LEMMA 6.1. $D(m,n) \le D(m, O(mn)^{1/2}) + O(1)$.

*Proof.* We use a reduction of Reif from iterated product to discrete Fourier transform [18]. Assume without loss of generality that $n$ is a power of 2, and let $\hat{n} = O(mn)^{1/2}$ also be a power of 2. Given $a_1, \cdots, a_m \in Z_{2^n}$, we let $a_{i,j}$ (for $i = 1, \cdots, m$ and $j = 0, \cdots, \hat{n} - 1$) be integers in $Z_{2^{n/\hat{n}}}$ such that $a_i = \sum_{j=0}^{\hat{n}-1} a_{i,j} 2^{jn/\hat{n}}$. To calculate an iterated product, we first compute in the vector $(g_{i,0}, \cdots, g_{i,\hat{n}-1}) = DFT((a_{i,0}, 2a_{i,1}, \cdots, 2^{\hat{n}-1} a_{i,\hat{n}-1})^T)$ for $i = 1, \cdots, m$. By Lemma 5.2, we can easily compute these DFTs in polynomial size and constant depth. For $k = 0, \cdots, \hat{n} - 1$

compute in (logspace uniform) $Th((m\hat{n})^{O(1)}, D(m, \hat{n}))$ the iterated product $e_k = \prod_{i=1}^m g_{i,k} \bmod (2^{\hat{n}}+1)$. Finally, compute in (logspace uniform) $Th((nm)^{O(1)}, 1)$ the vector $(f_0, 2f_1, \cdots, 2^{\hat{n}-1}f_{\hat{n}-1}) = DFT^{-1}((e_0, \cdots, e_{\hat{n}-1})^T)$ and output $\sum_{i=0}^{\hat{n}-1} f_i 2^{in/\hat{n}} = \prod_{i=1}^m a_i \bmod (2^n+1)$. The total depth is the depth of the recursion plus a constant amount, as stated in the lemma. □

LEMMA 6.2. $D(m,n) \le O\left(\frac{\log m \log\log n}{\log n}\right)$.

*Proof.* For any $\epsilon$, $0 < \epsilon < \frac{1}{2}$, we can compute the iterated product of $m$ integers by first computing the $\lceil m/n^\epsilon \rceil$ iterated products of $n^\epsilon$ integers and repeating this $\lceil \frac{\log m}{\epsilon \log n} \rceil$ times, getting $D(m,n) \le \lceil \frac{\log m}{\epsilon \log n} \rceil (D(n^\epsilon, n) + O(1))$. Applying Lemma 6.1 and this recurrence a constant number of times, we get

$$D(n^\epsilon, n) \le D(n^\epsilon, n^{1/2}) + O(1) \le D(n^{\epsilon/2}, n^{1/2}) + O(1).$$

Finally, applying the above recurrence $\log\log n$ times, we get $D(n^\epsilon, n) \le O(\log\log n)$. Hence

$$D(m,n) = O(\frac{\log m}{\log n}) O(\log\log n),$$

which is the bound claimed in the lemma. □

**6.1. Proof of Theorem 3.5.** Note that Lemma 6.2 implies that iterated product of $n^{O(1)}$ integers with $n$ bits each is in (logspace uniform) $Th(n^{O(1)}, \log\log n)$. Since computing the $n$ bit approximation of the reciprocal of an $n$ bit number reduces to simply computing the iterated sum of $n$ iterated products of size $n$, we can also compute residues modulo a number with $n$ bits in (logspace uniform) $Th(n^{O(1)}, \log\log n)$. Theorem 3.5 immediately follows, since we must compute residues, iterated sums and iterated products of $n = O(\log P)$ bit numbers. □

**7. Lower Bounds by Degree Bounds.** The degree of a multi-variable polynomial $f(y_1, \cdots, y_k)$ is the maximum sum of the powers of the variables appearing in any term (monomial) of $f(y_1, \cdots, y_k)$.

LEMMA 7.1. *Suppose $f(y_1, \cdots, y_k)$ is a nonzero polynomial of degree $d$ over a finite field $Z_p$, and $A$ is a subset of $Z_p$ of size $\sigma$. If $\sigma > d$, then $\exists(a_1, \cdots, a_k) \in A^k$ such that $f(a_1, \cdots, a_k) \ne 0$.*

*Proof.* By induction on $k$. For the basis case $k = 1$, we have $f(y_1)$ which is only a single variable polynomial. It is well known that any nonzero polynomial $f(x)$ of degree $d$ over any field can have at most $d$ zeros in the field (see, for example [6]), and since $\sigma > d$, at least one $a \in A$ must give a nonzero value for $f(a)$.

We make the induction hypothesis that the lemma holds for all polynomials with $< k$ variables. Since $f(y_1, \cdots, y_k)$ is nonzero, $\exists(u_1, \cdots, u_k) \in (Z_p)^k$ such that $f(u_1, \cdots, u_k) \ne 0$. Hence $f(u_1, y_2, \cdots, y_k) = f'(y_2, \cdots, y_k)$ is not a zero polynomial, and by the induction hypothesis $\exists(a_2, \cdots, a_k) \in A^{k-1}$ such that $f'(a_2, \cdots, a_k) = f(u_1, a_2, \cdots, a_k) \ne 0$. Let $g(y_1) = f(y_1, a_2, \cdots, a_k)$. $g(x)$ is clearly a non-zero polynomial, so by the basis step there is an $a_1 \in A$ such that $g(a_1) \ne 0$, and we have constructed $(a_1, a_2, \cdots, a_k) \in A^k$ such that $f(a_1, \cdots, a_k) \ne 0$. □

*Note:* A similar lemma for polynomial identity testing in infinite fields was proved by Ibarra and Moran [11].

**7.1. Proof of Theorem 3.6.** Fix any positive integer functions $S(n)$ and $D(n)$, where $D(n) = O(S(n)^{c'})$ for some constant $c' < 1$, and $S(n) \ge n$. Now consider a sequence of primes $\{P(1), P(2), \cdots, P(n), \cdots\}$ where $6(S(n)/D(n))^{D(n)} < P(n) \le 2^n$.

We will construct a family of $Z_{P(n)}$ circuits $\mathbf{C} = (C_1, C_2, \cdots, C_n, \cdots)$ of size $S(n)$ and depth $D(n)$. In particular, we let $v_1, \cdots, v_{n'}$ be the input nodes of $C_n$, where $n' = \lceil n/b \rceil$ and $b = \lfloor \log P(n) \rfloor \leq n$. We also let $w_0 = v_1$ denote the first input node. Each level $L = 1, \cdots, D(n)$ of $C_n$ consists of a single "product" node $w_L$ with $\lfloor S(n)/D(n) \rfloor$ edges entering $w_L$ from node $w_{L-1}$, so that $val(w_L)$ is the $\lfloor S(n)/D(n) \rfloor$ power of $val(w_{L-1})$; $w_{D(n)}$ is the unique output node of $C_n$. Let $y_1 = val(v_1), \cdots, y_{n'} = val(v_{n'})$ be the input values, and let $\vec{y} = (y_1, \cdots, y_{n'})$. We have constructed $C_n$ of size $\leq S(n)$ and depth $D(n)$ so that its output is the $d_n = (\lfloor S(n)/D(n) \rfloor)^{D(n)}$ degree polynomial $f_n(\vec{y}) = val(w_{D(n)}) = (y_1)^{d_n}$. Note however that by definition, $C_n$ gets decoded input integers $y_1, \cdots, y_{n'}$ only over input domain $Z_{2^b}$, whereas the computation is over the entire Finite Field $Z_{P(n)}$. Furthermore, the binary encoded output value is the residue $f_n(\vec{y}) \bmod 2^b$.

Next consider any $Z_{P(n)}$ circuit family $\mathbf{C}' = (C'_1, C'_2, \cdots, C'_n, \cdots)$ where $C'_n$ has size $S(n)^c$, for some constant $c \geq 1$, and simultaneous depth $D'(n) = o(D(n))$. We can assume without loss of generality that $C'_n$ has only a single output node, which computes value $g_n(\vec{y})$, where $\vec{y} = (y_1, \cdots, y_{n'})$ are the decoded integer values of its input nodes. Again note that $g_n(\vec{y})$ has only input domain $Z_{2^b}$. We wish to show that there exists some $\vec{y} \in (Z_{2^b})^{n'}$ such that $f_n(\vec{y}) \neq g_n(\vec{y}) \bmod 2^b$. Observe that $g_n(\vec{y})$ is of degree $\leq \prod_{L=1}^{D'(n)} e_L$ where $e_L$ is the number of edges of $C'_n$ entering nodes of level $L$. This product form is maximized when each $e_L = S(n)^c/D'(n)$, and since $\sum_{L=1}^{D'(n)} e_L = S(n)^c$ we get an upper bound on the maximum possible degree of $g_n(\vec{y})$ as $(S(n)^c/D'(n))^{D'(n)}$.

Since $D'(n) = o(D(n))$, we have for infinitely many $n$, and any constant $c$,

$$d_n = (\lfloor S(n)/D(n) \rfloor)^{D(n)}$$

$$\geq (S(n)/(2D(n)))^{D(n)} > S(n)^{(1-c')D(n)}$$

$$> S(n)^{cD'(n)} > (S(n)^c/D'(n))^{D'(n)} \geq deg(g_n(\vec{y})).$$

Fix some such $n$. For this value of $n$, by the above derivation $d_n > deg(g_n(\vec{y}))$ and $d_n \leq (S(n)/D(n))^{D(n)} < P(n)/6$, so by Lemma 7.1 there exists some $\vec{y} \in (Z_{P(n)})^{n'}$ such that $f_n(\vec{y}) \neq g_n(\vec{y})$. However, this does not prove Theorem 3.6 because we must actually show there exists some $\vec{y} \in (Z_{2^b})^{n'}$ such that $f_n(\vec{y}) \neq g_n(\vec{y}) \bmod 2^b$.

We define a new function $h_n(\vec{y})$ by the equation

$$h_n(\vec{y}) = \left(f_n(\vec{y}) - g_n(\vec{y}) - 2^b\right)\left(f_n(\vec{y}) - g_n(\vec{y})\right)\left(f_n(\vec{y}) - g_n(\vec{y}) + 2^b\right)$$

Note that if $f_n(\vec{y}) = g_n(\vec{y}) \bmod 2^b$ for all $\vec{y} \in (Z_{2^b})^{n'}$, then $h_n(\vec{y}) = 0$ for all inputs $\vec{y} \in (Z_{2^b})^{n'}$.

The degree of $h_n(\vec{y})$ is easily seen to be $3d_n$, and it is also obvious that $h_n(\vec{y})$ is not identically zero. Let $A = Z_{2^b}$, and since we know that

$$\text{degree}(h_n(\vec{y})) = 3d_n < 3\left(\frac{S(n)}{D(n)}\right)^{D(n)} < \frac{P(n)}{2} < |A|,$$

we can use Lemma 7.1 to see that $h_n(\vec{y}) \neq 0$ for at least one $n'$-tuple $(a_1, a_2, \cdots, a_{n'}) \in (Z_{2^b})^{n'}$. Theorem 3.6 follows immediately. $\square$

## 8. Conclusions.

**8.1. Threshold Circuits for Arithmetic Units.** Division is by far the most costly operation for Arithmetic Units. Our polynomial size, constant depth Threshold Circuits for arithmetic indicate that Threshold Circuits might be quite useful in highly parallel Arithmetic Units for integer division and trigonometric computations. It is an interesting question whether a high fanin threshold gate can be manufactured in a reasonably small area on silicon chips. Constant fanin threshold gates are in fact used in current NMOS and CMOS technologies. In theory, fanin $k$ threshold gates can be constructed so that with sufficient area (growing no more than quadraticly with $k$) these gates can be driven in unit time. In particular, Mead and Conway describe how to construct tally circuits (for $k$ input threshold) in VLSI with total area $O(k^2)$ and time $O(1)$ for moderate $k$ using pass transistors [14, pp. 78–80]. The Microelectronics Center of North Carolina is investigating the use of new microelectronic devices that may be used for Threshold gates with large fanin. If this is feasible in practice, then VLSI Arithmetic Units might be designed using Threshold Circuits to run much faster than currently possible (i.e., compared with the standard bounded fanin boolean logic gates of conventional VLSI).

**8.2. On Learning and Interpolation in Neural Networks..** Our positive results concerning Threshold Circuits (in particular Theorem 3.2 and Corollary 3.3) show that Threshold Circuits of polynomial size and constant depth can compute high accuracy approximations to a large class of multivariate rational polynomials, and furthermore can interpolate rational polynomials with a constant number of variables. Learning by algebraic interpolation appears to be appropriate in certain constrained cases such as low level vision [7], and would likely be much more efficient than previously proposed methods for learning (such as found in [1] and [9]), which are essentially brute force. Nevertheless, even making the apparently reasonable assumption that certain portions of the lower brain act essentially as Threshold Circuits of constant depth does not necessarily imply that the lower brain is wired so as to compute approximations or interpolations of multivariate polynomials. However, our theoretical results do provide strong evidence of the *feasibility* of neuron nets which evaluate and interpolate such polynomial functions.

A neural biologist might, for example, make experimental tests to verify this by using a computer to monitor input-output response functions of neuron nets. Specifically, the lower brain very rapidly provides feedback control for certain muscles; this control appears to be smooth and nonlinear. Such easily observable responses would appear to be ideal to monitor and to interpolate. By using known randomized multivariate polynomial identity tests, such as those of Ibarra and Moran [11], one can with very high likelihood verify that the input-output response of a neuron net is a specific interpolated multivariate polynomial.

**8.3. Lower Bound Conjectures.** Finally, we make two lower bound conjectures concerning Threshold Circuits.

CONJECTURE 8.1. *For $D'(n) = o(D(n))$, there exists an $f \in \bigcup_{c\geq 1} Th(n^c, D(n))$ which is not in $\bigcup_{c\geq 1} Th(n^c, D'(n))$.*

Let DETERMINANT be the problem "given an $n \times n$ matrix $A$ with 0,1 elements, compute the determinant of $A$."

CONJECTURE 8.2. *DETERMINANT is not contained in $Th(n^c, 1)$ for any constant $c$.*

## REFERENCES


[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Sciences, 9 (1985), pp. 147–169.

[2] P. W. Beame, S. A. Cook, and H. J. Hoover, *Log depth circuits for division and related problems*, SIAM J. Comput., 15 (1986), pp. 994–1003.

[3] D. Bini, *Parallel solution of certain Toeplitz linear systems*, SIAM J. Comput., 13 (1984), pp. 268–276.

[4] D. Bini and V. Pan, *Fast parallel algorithms for polynomial division over arbitrary field of constants*, 1984. Note Interna, Departimento di Informatica, Universita di Pisa.

[5] J. A. Feldman and D. H. Ballard, *Connectionist models and their properties*, Cognitive Science, 6 (1982), pp. 205–254.

[6] J. B. Fraleigh, *A First Course in Abstract Algebra*, Addison-Wesley Publishing Company, Reading, MA, 1982.

[7] W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, M.I.T. Press, Cambridge, MA, 1981.

[8] J. Hastad and T. Leighton, *Division in $O(\log n)$ depth using $O(n^{1+\epsilon})$ processors*, 1986. Unpublished note.

[9] G. E. Hinton, T. Sejnowski, and D. A. Ackley, *Boltzmann machines: Constraint satisfaction networks that learn*, Tech. Rep. CMU-CS-84-119, Carnegie Mellon University, 1984.

[10] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences, USA, 79 (1982), pp. 2554–2558.

[11] O. H. Ibarra and S. Moran, *Probabilistic algorithms for deciding equivalence of straight-line programs*, JACM, 30 (1983), pp. 217–228.

[12] H. Jung, *On probabilistic tape complexity and fast circuits for matrix inversion problems*, in Proceedings of the 11th Colloquium on Automata, Languages, and Programming, 1984, pp. 281–291. Springer-Verlag LNCS vol. 172.

[13] H. T. Kung, *New algorithms and lower bounds for the parallel evaluation of certain rational expressions and recurrences*, JACM, 23 (1976), pp. 252–261.

[14] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.

[15] M. L. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, M.I.T. Press, Cambridge, MA, 1968.

[16] I. Parberry and G. Schnitger, *Parallel computation with threshold functions*, in Proceedings of Conference on Structure in Complexity Theory, 1986, pp. 272–290. Springer-Verlag LNCS vol. 223.

[17] N. Pippenger, *The complexity of computations by networks*, IBM J. Res. Dev., 31 (1987), pp. 235–243.

[18] J. H. Reif, *Logarithmic depth circuits for algebraic functions*, SIAM J. Comput., 15 (1986), pp. 231–242.