SIAM J. DISCRETE MATH.
Vol. 7, No. 2, pp. 284–292, May 1994

014

ON OPTIMAL DEPTH THRESHOLD CIRCUITS FOR MULTIPLICATION AND RELATED PROBLEMS *

KAI-YEUNG SIU† AND VWANI P. ROYCHOWDHURY‡

Abstract. Let $\widehat{LT}_d$ denote the class of functions that can be computed by depth-d threshold circuits with polynomial size and polynomially bounded integer weights. Using the results in [M. Goldman, J. Håstad, and A. Razborov, in *Proc. 7th Annual Conference on Structure in Complexity Theory*], [M. Goldman and M. Karpinski, *Constructing depth $d+1$ majority circuits that simulate depth d threshold circuits*, unpublished] we show that multiple sum is in $\widehat{LT}_2$, and multiplication and division are in $\widehat{LT}_3$. Moreover, it follows from the lower-bound results in [A. Hajnal et al., *IEEE Sympos. Foundations of Comput. Sci.*, 28 (1987), pp. 99–110], [T. Hofmeister and P. Pudlák, *Forschungbericht Nr.* 477 Uni Dortmund, 1992] that these threshold circuits are optimal in circuit depth. The authors also indicate that these techniques can be applied to construct polynomial-size depth-3 threshold circuits for powering and depth-4 threshold circuits for multiple product.

Key words. threshold circuits, linear threshold functions, multiplication, division, arithmetic functions

AMS subject classifications. 68Q15, 68Q05, 68Rxx

1. Introduction.

In this paper, we consider the power of small-depth threshold circuits in computing arithmetic functions such as multiple sum, multiplication, division, and powering. Threshold circuits are unbounded fan-in Boolean circuits in which each gate computes a *linear threshold function.* A linear threshold function $f(X)$ is a Boolean function such that

$$f(X) = \operatorname{sgn}(F(X)) = \begin{cases} 1 & \text{if } F(X) \geq 0, \\ 0 & \text{if } F(X) < 0, \end{cases}$$

where

$$F(X) = \sum_{i=1}^{n} w_i \cdot x_i + w_0.$$

The real coefficients w_i are commonly referred to as the *weights* of the threshold function. It is well known that the weights can be chosen to be integers [11].

However, the magnitudes of the integers can be exponentially large in the number of inputs. The *size* of a circuit is the number of gates. If the number of gates in a threshold circuit is polynomially bounded, then so is the number of wires in the circuit, and vice versa. Unless otherwise specified, we assume in the following discussions that all the weights in the threshold circuits are integers (possibly exponential) and that the sizes of the threshold circuits are polynomially bounded.

* Received by the editors April 1, 1992; accepted for publication (in revised form) November 22, 1992. This work was supported in part by the Joint Services Program at Stanford University (United States Army, Navy, and Air Force) contract DAAL03-88-C-0011 and the Department of the Navy (NAVELEX) contract N00039-84-C-0211, NASA Headquarters, Center for Aeronautics and Space Information Sciences grant NAGW-419-S6.

† Department of Electrical and Computer Engineering, University of California, Irvine, California 92717 (siu@balboa.eng.uci.edu). The work of this author was supported in part by Irvine Faculty Research grant 91/92-27.

‡ School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907 (vwani@drum.ecn.purdue.edu). The work of this author was supported in part by the General Motors Faculty fellowship from the School of Engineering at Purdue University.

An important open problem in circuit complexity theory is to find an explicit function that cannot be computed in constant-depth polynomial-size threshold circuits. The first attempt toward solving this problem was made by Hajnal et al. [6]. They showed that the INNER PRODUCT MOD 2_n function $(x_1 \wedge y_1) \oplus \cdots \oplus (x_n \wedge y_n)$ can be computed in linear-size depth-3 threshold circuit but requires exponential-size depth-2 threshold circuit to compute, when the weights are polynomially bounded integers. This result gives a separation of the class of depth-2 polynomial-size threshold circuits from the class of depth-3 polynomial-size threshold circuits, when the weights are polynomially bounded.

Using the notation of [12], let us denote the class of depth-d polynomial-size threshold circuits where the weights are polynomially bounded by $\widehat{LT}_d$ and the corresponding class where the weights are unrestricted by LT_d. Then the exponential lower bound result on INNER PRODUCT MOD 2_n in [6] implies that $\widehat{LT}_2 \subsetneq \widehat{LT}_3$. As another consequence of this lower bound result, we can show that multiplication and division [9] of two n-bit integers also require exponential-size depth-2 threshold circuits with polynomially bounded weights to compute. In [7] Håstad and Goldmann proved an exponential lower bound on the size of depth-3 threshold circuits with the restriction that the bottom fan-in of the circuit is small. However, no exponential lower bound result on the size of depth-2 threshold circuits is known when there is no restriction on the size of the weights.

It is implicit in [3] that a constant-depth threshold circuit with arbitrary weights can be simulated by another constant-depth threshold circuit with polynomially bounded weights at the expense of at most a polynomial increase in size. To study the exact relationship between the depths of threshold circuits with arbitrary weights and with polynomially bounded weights, Siu and Bruck [12] showed that any depth-d threshold circuit can be simulated by a depth-$(2d+1)$ threshold circuit with polynomially bounded weights (both have polynomial size); that is, $LT_d \subset \widehat{LT}_{2d+1}$. The result was substantially strengthened by Goldmann, Håstad, and Razborov [4]; they showed that, in fact, $LT_d \subset \widehat{LT}_{d+1}$. While the proof techniques in [4] are not constructive, Goldmann and Karpinski [5] later gave an explicit construction of the circuits in [4].

Threshold circuits are powerful as a model of computation. In fact, many common arithmetic functions have been shown to be computable in small-depth threshold circuits. It was first shown in [12] that multiple sum and multiplication can be computed in $\widehat{LT}_3$ and $\widehat{LT}_4$, respectively. This result was also independently discovered later by Hofmeister, Hohberg, and Köhling [8] with much improvement on the circuit size. More recently, it was shown in [13] that small-depth threshold circuits can be constructed for division and related problems. In particular, division and powering are in $\widehat{LT}_4$ and multiple product (iterated multiplication) is in $\widehat{LT}_5$. The question of whether multiplication of two n-bit integers can be computed in $\widehat{LT}_3$ had remained open since the work of Hajnal et al. [6].

In this paper, we demonstrate that applications of the results in [4], [5] yield a depth-3 threshold circuit of polynomially bounded weights for multiplication; i.e., multiplication is in $\widehat{LT}_3$. It is clear from the result in [6] that such threshold circuit is optimal in depth. Moreover, similar techniques can be applied to show that division and powering can be computed in $\widehat{LT}_3$ and multiple product can be computed in $\widehat{LT}_4$. The result in [9] also implied that our division circuit is optimal in depth.

The rest of the paper is outlined as follows. We first describe a depth-2 threshold circuit for multiple sum. Using this result, a depth-3 threshold circuit for multiplication follows easily. We then indicate how to apply the result for multiple sum to

obtain depth-3 threshold circuits for division and powering and depth-4 threshold circuit for multiple product. Since the techniques for deriving the results on division related problems are similar to those in [13], we only sketch the proof and indicate how the results in [13] can be improved using the results in [4], [5]. In the final section, we conclude with some open problems.

2. Main Results.

DEFINITION 1. *Given n n-bit integers, $z_i = \sum_{j=0}^{n-1} z_{i,j}2^j$, $i = 1, \ldots, n$, $z_{i,j} \in \{0,1\}$, we define* multiple sum *to be the problem of computing the $(n + \log n)$-bit sum $\sum_{i=1}^{n} z_i$ of the n integers.*

The above problem is also referred to as *iterated addition* in the literature.

DEFINITION 2. *Given two n-bit integers, $x = \sum_{j=0}^{n-1} x_j 2^j$ and $y = \sum_{j=0}^{n-1} y_j 2^j$, we define* multiplication *to be the problem of computing the $(2n)$-bit product of x and y.*

It is easy to see that, if multiple sum can be computed in $\widehat{LT}_2$, then multiplication can be computed in $\widehat{LT}_3$. We first prove the result on multiple sum. Our result hinges on the results in [4], [5]. The key observation is that multiple sum can be computed as a sum of polynomially many linear threshold (LT_1) functions (with exponential weights). Let us first state the results [4], [5].

LEMMA 2.1 (see [4], [5]). *Let $\widehat{LT}_d$ denote the class of depth-d polynomial-size threshold circuits where the weights at the output gate are polynomially bounded integers (with no restriction on the weights of the other gates). Then $\widetilde{LT}_d = \widehat{LT}_d$ for any fixed integer $d \geq 1$.*

The following lemma is a generalization of a result in [10]. Informally, the result says that, if a function is 1 when a weighted sum (possibly exponential) of its inputs lies in one of polynomially many intervals, and is 0 otherwise, then the function can be computed as a sum of polynomially many LT_1 functions.

LEMMA 2.2. *Let $S = \sum_{i=1}^{n} w_i x_i$ and $f(X)$ be a function such that $f = 1$ if $S \in [l_i, u_i]$ for $i = 1, \ldots, N$ and $f = 0$ otherwise, where N is polynomially bounded in n. Then f can be computed as a sum of polynomially many LT_1 functions, and thus $f \in \widetilde{LT}_2$.*

Proof. For $j = 1, \ldots, N$, let

$$y_{l_j} = \operatorname{sgn}\left\{\sum_{i=1}^{n} w_i x_i - l_j\right\} \quad \text{and} \quad y_{u_j} = \operatorname{sgn}\left\{u_j - \sum_{i=1}^{n} w_i x_i\right\}.$$

We claim that

$$f(X) = \sum_{j=1}^{N} (y_{l_j} + y_{u_j}) - N,$$

and therefore

$$f(X) = \operatorname{sgn}\left\{\sum_{j=1}^{N} (y_{l_j} + y_{u_j}) - N - 1\right\}.$$

Note the following: If, for $j = 1, \ldots, N$, $\sum_{i=1}^{n} w_i x_i \notin [l_j, u_j]$, then $y_{l_j} + y_{u_j} = 1$ for all j. Thus, $\sum_{j=1}^{N} (y_{l_j} + y_{u_j}) - N = 0$. On the other hand, if $\sum_{i=1}^{n} w_i x_i \in [l_j, u_j]$ for some $j \in \{1, \ldots, N\}$, then $y_{l_j} + y_{u_j} = 2$ and $y_{l_i} + y_{u_i} = 1$ for $i \neq j$. Thus, $\sum_{i=1}^{N} (y_{l_i} + y_{u_i}) - N = N + 1 - N = 1$. □

Combining the above two lemmas yields a depth-2 threshold circuit for multiple sum.

THEOREM 2.3. *Multiple sum is in* $\widehat{LT}_2$.

Proof. Given n n-bit integers $z_i = \sum_{j=0}^{n-1} z_{i,j}2^j$, $i = 1, \ldots, n$, the sum $\tilde{S} = \sum_{i=1}^{n} z_i$ can be represented as an $(n + \log n)$-bit integer, $\tilde{S} = \sum_{i=0}^{n+\log n-1} \tilde{s}_i 2^i$. Clearly, the kth bit of $\tilde{S}$, $\tilde{s}_{k-1}$ is the same as the kth bit of the sum of the first k-bits of the z_i's, i.e., the kth bit of $\sum_{i=1}^{n} \sum_{j=0}^{k-1} z_{i,j}2^j$. Thus, to prove the theorem, it suffices to show that the kth bit of n k-bit integers can be computed in $\widehat{LT}_2$, for $k = 1, \ldots, n + \log n$. We first construct a depth-2 threshold circuit where the threshold gates in the first level have exponential weights.

Let $S = \sum_{i=1}^{n} \sum_{j=0}^{k-1} 2^j z_{i,j} = \sum_{l=0}^{\log n+k-1} 2^l s_l$ be the sum of n k-bit integers. Note that the kth bit of S, s_{k-1} is 1 if $S \in I_{j,k} = [j2^{k-1}, (j+1)2^{k-1} - 1]$ for $j = 1, 3, 5, \ldots, 2^{\log n+1} - 1$ and 0 otherwise. Since there are only polynomially many intervals $I_{j,k}$, it follows from Lemma 2.2 the kth bit can be computed in $\widehat{LT}_2$. Now apply Lemma 2.1 for $d = 2$; thus the kth bit can be computed in $\widehat{LT}_2$. □

It is also easy to see that multiple sum cannot be computed in LT_1. Simply observe that the first bit of the sum is the parity function, which does not belong to LT_1. Thus the above threshold circuit for multiple sum has minimum possible depth.

THEOREM 2.4. *Multiplication is in* $\widehat{LT}_3$.

Proof. Let the two integers be $x = x_{n-1}x_{n-2}\ldots x_0$, $y = y_{n-1}y_{n-2}\ldots y_0$. The first level of our circuit outputs the n $(2n)$-bit integers $z_i = z_{i_{2n-1}} z_{i_{2n-2}} \ldots z_{i_0}$, for $i = 0, \ldots, n-1$, where

$$z_i = \underbrace{0\ldots0}_{n-i}(x_{n-1} \wedge y_i)(x_{n-2} \wedge y_i)\ldots(x_0 \wedge y_i)\underbrace{0\ldots0}_{i}.$$

This level requires $O(n^2)$ gates. It is easy to see that the product of x and y is simply the sum of the z_i's. By Theorem 2.3, the sum of the z_i's can be computed using two more levels of polynomially many threshold gates (with polynomially bounded weights). □

We can further apply the results in [4], [5] to construct small-depth threshold circuits for division, powering, and multiple product. Let us give a formal definition of these problems.

DEFINITION 3. *Let Z be an n-bit integer ≥ 0. We define* powering *to be the n^2-bit representation of Z^n.*

DEFINITION 4. *Given n n-bit integers z_i, $i = 1, \ldots, n$, we define* multiple product *to be the n^2-bit representation of $\prod_{i=1}^{n} z_i$.*

The above problem is also called *iterated product* or *iterated multiplication* in the literature.

Suppose that we want to compute the quotient of two integers. Some quotient in binary representation might require infinitely many bits; however, a circuit can only compute the most significant bits of the quotient. If a number has both finite and infinite binary representation (for example, 0.1 = 0.0111...), we always express the number in its finite binary representation. We are interested in computing the truncated quotient, defined below.

DEFINITION 5. *Let X and $Y \geq 1$ be two input n bit integers. Let $X/Y = \sum_{i=-\infty}^{n-1} z_i 2^i$ be the quotient of X divided by Y. We define* $\mathrm{DIV}_k(X/Y)$ *to be* X/Y

truncated *to the* $(n+k)$*-bit number, i.e.,*

$$\mathrm{DIV}_k(X/Y) = \sum_{i=-k}^{n-1} z_i 2^i.$$

In particular, $\mathrm{DIV}_0(X/Y)$ is $\lfloor X/Y \rfloor$, the greatest integer $\leq X/Y$.

Using the results in [2], [12], it was shown in [13] that powering and division can be computed in $\widehat{LT}_4$ and that multiple product can be computed in $\widehat{LT}_5$. Combining these results with the results in [4], [5], we can reduce the depths of these circuits by one. We only indicate the key steps in the construction of the circuits in [13]. For other details of the proof, see [13]. Let us rephrase the results in [4], [5] as the following lemma.

LEMMA 2.5 (see [4], [5]). *Let* $f(X) \in LT_1$. *Then, for any* $k > 0$, *there exist* m *functions* $t_1(X), \ldots, t_m(X) \in \widehat{LT}_1$ *such that, for all* X,

$$\left| f(X) - \frac{1}{N} \sum_{j=1}^{m} t_j(X) \right| \leq n^{-k},$$

where m *and* N *are integers bounded by a polynomial in* n.

By Lemma 2.2, each bit in the sum of multiple sum can be computed as a sum of polynomially many LT_1 functions. Combining this result with the above lemma yields the following result.

LEMMA 2.6. *Let* s_i *be any of the outputs in multiple sum. Then, for any* $\hat{k} > 0$, *there exist* $\hat{t}_j(X) \in \widehat{LT}_1$ *such that*

$$\left| s_i - \frac{1}{\hat{N}} \sum_{j=1}^{\hat{m}} \hat{t}_j(X) \right| \leq n^{-\hat{k}},$$

where $\hat{m}$ *and* $\hat{N}$ *are integers bounded by a polynomial in* n.

The following lemma, which was shown in [13], states that, if t_1 and t_2 can be closely approximated by polynomially many $\widehat{LT}_k$ functions, so is their product $t_1 \wedge t_2$.

LEMMA 2.7. *Suppose that, for* $i = 1, 2$ *and for every* $c > 0$, *there exist integers* m_i, w_{i_j}, *and* N *that are bounded by a polynomial in* n *such that, for all inputs* X,

$$\left| t_i(X) - \frac{1}{N} \sum_{j=1}^{m_i} w_{i_j} t_{i_j}(X) \right| = O(n^{-c}),$$

where each $t_{i_j} \in \widehat{LT}_k$. *Then there exist integers* $\tilde{m}$, $\tilde{w}_j$, *and* $\tilde{N}$ *that are bounded by a polynomial in* n *such that*

$$\left| t_1(X) \wedge t_2(X) - \frac{1}{\tilde{N}} \sum_{j=1}^{\tilde{m}} \tilde{w}_j \tilde{t}_j(X) \right| = O(n^{-c}),$$

where each $\tilde{t}_j \in \widehat{LT}_k$.

To avoid cumbersome explanations in the following discussions, we say informally that every LT_1 function and each output bit in multiple sum can be *closely approximated* by a sum of polynomially many $\widehat{LT}_1$ functions, in the sense of Lemmas 2.5 and 2.6.

THEOREM 2.8. *Powering is in* $\widehat{LT}_3$.

Proof. Let X be an n-bit integer and $Z = X^n$. Let p_i denote the ith prime number and let $\pi(k)$ denote the number of primes $\leq k$. Let

$$P_n = \prod_{i=1}^{\pi(n^2)} p_i$$

be the product of all primes $\leq n^2$. Then we can show that $Z < 2^{n^2} < P_n$, and thus $(Z \bmod P_n) = Z$.

Using the Chinese Remainder Theorem, we can compute Z with the following steps:

1. For $i = 1, ..., n^2$, compute in parallel the values $r_i = Z \bmod p_i$;
2. $\tilde{Z} = \sum_{i=1}^{\pi(n^2)} r_i \cdot m_i$;
3. $Z = (Z \bmod P_n) = (\tilde{Z} \bmod P_n)$;

where, in step 2 above, the m_i are fixed integers (possibly exponentially large), and therefore step 2 is, in fact, multiple sum. Moreover, we can show that $\tilde{Z} \leq n^4 \cdot P_n$, and hence $Z = (\tilde{Z} \bmod P_n) = \tilde{Z} - k \cdot P_n$ for some k, where $0 \leq k \leq n^4$. For each $k \in \{0, \ldots, n^4\}$, let

$$\begin{aligned} EQ_k(Z) &= \operatorname{sgn}\{\tilde{Z} - k \cdot P_n\} + \operatorname{sgn}\{(k+1)P_n - \tilde{Z} - 1\} - 1 \\ &= \begin{cases} 1 & \text{if } Z = (\tilde{Z} \bmod P_n) = \tilde{Z} - k \cdot P_n, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Let z_{jk} be the jth bit of $\tilde{Z} - k \cdot P_n$. Then the jth bit of Z is

$$\bigvee_{0 \leq k \leq n^4} (EQ_k(Z) \wedge z_{jk}).$$

We can compute the values r_i in step 1, above, as a sum of polynomially many $\widehat{LT}_1$ functions. By Lemmas 2.6 and 2.5, each z_{jk} and each $EQ_k(Z)$ can be *closely approximated* by a sum of polynomially many $\widehat{LT}_1$ functions with variables r_i. Thus $EQ_k(Z)$ and z_{jk} can be *closely approximated* as a sum of the outputs from polynomially many depth-2 threshold circuits whose inputs are the variables X. By Lemma 2.7, it follows that $(EQ_k(Z) \wedge z_{jk})$ can also be closely approximated as a sum of the outputs from polynomially many depth-2 threshold circuits. Hence, each of the outputs $\bigvee_{0 \leq k \leq n^4}(EQ_k(Z) \wedge z_{jk})$ can be computed in a depth-3 threshold circuit. □

Remark 1. In [13] each $EQ_k(Z)$ and z_{jk} is closely approximated as a sum of outputs from polynomially many depth-3 threshold circuits ($\widehat{LT}_3$). Lemmas 2.6 and 2.5 enable us to save one level of threshold gates in computing them.

THEOREM 2.9. *Multiple product is in* $\widehat{LT}_4$.

Proof. Let $Z = \prod_{j=1}^{n} z_j$, where each z_i is an n-bit integer. The proof is very similar to the proof of Theorem 2.8. We can compute Z using the same three steps as in Theorem 2.8. The only difference is that now each $r_i = Z \bmod p_i$ is computed as a sum of polynomially many depth-2 threshold circuits ($\widehat{LT}_2$), one more level of threshold gates than the circuit for powering. □

THEOREM 2.10. $\mathrm{DIV}_k(x/y)$ *is in* $\widehat{LT}_3$.

Proof. Note that $\mathrm{DIV}_k(x/y) = 2^{-k}\mathrm{DIV}_0(2^k x/y)$; so it suffices to prove our claim for the case where $k = 0$. The resulting threshold circuits for the general case when

k is polynomial in n have the same depth and the size will increase by a polynomial factor.

The underlying idea is to compute an *overapproximation* $\tilde{a}$ to x/y such that $x/y \leq \tilde{a} \leq x/y + 2^{-(n+1)}$. Then we can show that $\lfloor \tilde{a} \rfloor = \lfloor x/y \rfloor$.

Since x/y is equal to the product of x and y^{-1}, it is enough to get an overapproximation $\tilde{y}^{-1}$ of y^{-1} with error $\leq 2^{-(2n+1)}$. Then we can compute the approximation $q = x \cdot \tilde{y}^{-1}$ to x/y with an error $\leq x2^{-(2n+1)} \leq 2^{-(n+1)}$ with a small-depth threshold circuit.

To construct an overapproximation of y^{-1}, let $j \geq 1$ be the integer such that $2^{j-1} \leq y < 2^j$. Note that $|1 - y2^{-j}| \leq \frac{1}{2}$, and we can express y^{-1} as a series expansion

$$y^{-1} = 2^{-j} \cdot (1 - (1 - y2^{-j}))^{-1} = 2^{-j} \sum_{i=0}^{\infty} (1 - y2^{-j})^i.$$

If we put

$$\tilde{y}^{-1} = 2^{-j} \sum_{i=0}^{2n} (1 - y2^{-j})^i,$$

then the difference

$$0 \leq (y^{-1} - \tilde{y}^{-1}) \leq 2^{-j} \sum_{i=(2n+1)}^{\infty} 2^{-i} \leq 2^{-(2n+1)}.$$

Since $x < 2^n$, we have

$$0 \leq (xy^{-1} - x\tilde{y}^{-1}) < 2^{-(n+1)}.$$

Suppose for the moment that we can find the integer $j \geq 1$ such that $2^{j-1} \leq y < 2^j$. Now we can rewrite

$$x\tilde{y}^{-1} = \frac{1}{2^{j(2n+2)}} \sum_{i=0}^{2n+1} 2^{j(2n+1-i)} x(2^j - y)^i.$$

Let $Z_j = \sum_{i=0}^{2n+1} 2^{j(2n+1-i)} x(2^j - y)^i$. Then $x\tilde{y}^{-1} = (1/2^{j(2n+2)})Z_j$, a shifting of the bits in Z_j.

Again, we can compute Z_j via the Chinese Remainder Theorem as follows:

1. For $i = 1, \ldots, N$, compute in parallel the values $r_{i,j} = Z_j \bmod p_i$;
2. $\tilde{Z}_j = \sum_{i=1}^{N} r_{i,j} \cdot m_i$;
3. $Z_j = (Z_j \bmod P_N) = (\tilde{Z}_j \bmod P_N)$;

where N is a sufficiently large integer such that the product of the first N primes $\prod_{i=1}^{N} p_i = P_N > Z_j$ for all $j = 1, \ldots, n$. Moreover, we can show that $\tilde{Z}_j \leq n^\alpha P_N$ for some $\alpha > 0$, and hence $Z_j = (\tilde{Z}_j \bmod P_N) = \tilde{Z}_j - kP_n$ for some k, where $0 \leq k \leq n^\alpha$. For each $k \in \{0, ..., n^\alpha\}$, let

$$\begin{aligned} EQ_k(Z_j) &= \operatorname{sgn}\{\tilde{Z}_j - kP_N\} + \operatorname{sgn}\{(k+1)P_N - \tilde{Z}_j - 1\} - 1 \\ &= \begin{cases} 1 & \text{if } Z_j = (\tilde{Z}_j \bmod P_N) = \tilde{Z}_j - kP_N, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Let $\sum_l z_{j,k,l}2^l = (1/2^{j(2n+2)})(\tilde{Z}_j - kP_N)$. If $EQ_{k^*}(Z_j) = 1$, i.e., $(\tilde{Z}_j \bmod P_N) = \tilde{Z}_j - k^*P_N$, then

$$\mathrm{DIV}_0(x/y) = \sum_{l=0}^{n-1} z_{j,k^*,l}2^l.$$

Thus the ith bit of $\mathrm{DIV}_0(x/y)$ can be computed as

$$\bigvee_{1\le k\le n^\alpha} (EQ_k(Z_j) \wedge z_{j,k,i}).$$

The above expression is based on the assumption that we can find the unique integer $j \ge 1$ such that $2^{j-1} \le y < 2^j$. We can compute such integer j in parallel without increasing the depth of the circuit. To see this, for each $j \in \{1, \ldots, n\}$, let

$$I_j = \mathrm{sgn}\{y - 2^{j-1}\} + \mathrm{sgn}\{2^j - y - 1\} - 1 = \begin{cases} 1 & \text{if } 2^{j-1} \le y < 2^j, \\ 0 & \text{otherwise.} \end{cases}$$

Then the ith bit of $\mathrm{DIV}_0(x/y)$ is

$$\bigvee_{1\le j\le n} \bigvee_{1\le k\le n^\alpha} (I_j \wedge EQ_k(Z_j) \wedge z_{j,k,i}).$$

Now apply the same argument as in Theorem 2.8; we can show that each $(I_j \wedge EQ_k(Z_j) \wedge z_{j,k,i})$ can be closely approximated by a sum of outputs from polynomially many $\widehat{LT}_2$ functions. Hence the final result can be computed in $\widehat{LT}_3$. □

Remark 2. In [13] each $(I_j \wedge EQ_k(Z_j) \wedge z_{j,k,i})$ is closely approximated by a sum of outputs from polynomially many depth-3 threshold circuits ($\widehat{LT}_3$). Here again Lemmas 2.6 and 2.5 enable us to save one level of threshold gates in computing them.

3. Concluding remarks. We have demonstrated optimal-depth threshold circuits for multiplication, multiple sum, and division. We also indicated how the techniques can be applied to obtain depth-3 threshold circuits for powering and depth-4 threshold circuit for multiple product. These results are improvements on the depths of the circuits constructed in [13]. Moreover, the construction of these circuits can all be made explicit using the results in [5].

There are a few open problems, below, related to the results in this paper:

1. What is the minimal size of a depth-3 threshold circuit for multiplication?
2. Can INNER PRODUCT MOD 2_n and multiplication be computed in LT_2? A negative answer to this question will provide the separation $LT_2 \subsetneq \widehat{LT}_3$.

REFERENCES

[1] N. Alon and J. Bruck, *Explicit Constructions of Depth-2 Majority Circuits for Comparison and Addition*, IBM Research Report, RJ 8300, August 1991.

[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] A. K. Chandra, L. Stockmeyer, and U. Vishkin, *Constant depth reducibility*, SIAM J. Comput., 13 (1984), pp. 423–439.

[4] M. Goldmann, J. Håstad, and A. Razborov, *Majority gates vs. general weighted threshold gates*, in Proc. 7th Annual Conference on Structure in Complexity Theory Conference, 1992, pp. 2–13.

[5] M. GOLDMANN AND M. KARPINSKI, *Simulating threshold circuits by majority circuits*, in Proc. of the 25th annual ACM Symposium on the Theory of Computing (STOC), San Diego, CA, May 1993, pp. 551–560.

[6] A. HAJNAL, W. MAASS, P. PUDLÁK, M. SZEGEDY, AND G. TURÁN, *Threshold circuits of bounded depth*, IEEE Sympos. Foundations of Comput. Sci., 28 (1987), pp. 99–110.

[7] J. HÅSTAD AND M. GOLDMANN, *On the power of small-depth threshold circuits*, Computational Complexity, 1 (1991), pp. 113-129.

[8] T. HOFMEISTER, W. HOHBERG, AND S. KÖHLING, *Some notes on threshold circuits and multiplication in depth* 4, Inform. Process. Lett., 39 (1991), pp. 219–225.

[9] T. HOFMEISTER AND P. PUDLÁK, *A proof that division is not in* TC^0_2, Forschungsbericht Nr. 447, Uni Dortmund, 1992.

[10] S. MUROGA, *The principle of majority decision logic elements and the complexity of their circuits*, Internat. Conf. on Information Processing, Paris, France, June 1959.

[11] S. MUROGA, I. TODA, AND S. TAKASU, *Theory of majority decision elements*, J. Franklin Inst., 271 (1961), pp. 376–418.

[12] K.-Y. SIU AND J. BRUCK, *On the power of threshold circuits with small weights*, SIAM J. Discrete Math., 4 (1991), pp. 423–435.

[13] K.-Y. SIU, J. BRUCK, T. KAILATH, AND T. HOFMEISTER, *Depth-efficient neural networks for division and related problems*, IEEE Trans. Inform. Theory, 39 (1933), pp. 946–956.