Depth-Efficient Threshold Circuits for Multiplication and Symmetric Function Computation *

Chi-Hsiang Yeh and Emmanouel A. Varvarigos

University of California, Santa Barbara, CA 93106-9560, USA

Abstract. The multiplication operation and the computation of symmetric functions are fundamental problems in arithmetic and algebraic computation. We describe unit-weight threshold circuits to perform the multiplication of two n-bit integers, which have fan-in k, edge complexity $O(n^{2+1/d})$, and depth $O(\log d + \log n/\log k)$, for any fixed integer $d > 0$. For a given fan-in, our constructions have considerably smaller depth (or edge complexity) than the best previous circuits of similar edge complexity (or depth, respectively). Similar results are also shown for the iterated addition operation and the computation of symmetric functions. In particular, we propose a unit-weight threshold circuit to compute the sum of m n-bit numbers that has fan-in k, edge complexity $O(nm^{1+1/d})$, and depth $O(\log d + \log m/\log k + \log n/\log k)$.

1 Introduction

The delay required to perform multiplication and iterated addition is crucial to the performance of many computationally intensive applications. The evaluation of symmetric Boolean functions is also another important problem that appears often in arithmetic and algebraic computations. In this paper, we propose some new unit-weight threshold circuits to perform multiplication and iterated addition, and compute symmetric functions. Our constructions provide efficient tradeoffs between fan-in, depth, and edge complexity, and they outperform the best previous results that use threshold or AND-OR circuits.

Threshold circuits have been proposed as a powerful computational model for arithmetic and other computations. A *linear threshold function* is defined as a Boolean function

$$\mathrm{Sgn}(F(X)) = \begin{cases} 1 \text{ if } F(X) \geq 0; \\ 0 \text{ if } F(X) < 0, \end{cases}$$

where $X = (x_1, ..., x_k) \in \{0,1\}^k$ are the input variables, and $F(X) = \sum_{i=1}^{k} w_i x_i + w_0$. The numbers $w_i, i = 1, 2, \ldots, k$, are called the *weights*, and w_0 is called the *bias* of the threshold function. A *threshold circuit* is defined as a computational network that consists of the acyclic interconnection of threshold gates, each of which computes a linear threshold function. The *depth* of a circuit is defined as the length of the longest path from any input to any output node of the circuit.

* Research supported partially by NSF under grant NSF-RIA-08930554.

The *edge complexity* (or *size*) is defined as the number of edges in the circuit, while its *fan-in* is the largest fan-in of all the gates contained in it. The computational model used in this paper assumes unit-weight threshold circuits, where the weights $w_i \in \{1, -1\}$, for all $i = 1, 2, ..., k$. Unit-weight threshold circuits are considerably less expensive to implement than threshold circuits of large weight.

Recent papers [1, 2, 5, 3, 7, 8, 11] have begun to study the class of functions that are computable by threshold circuits that have constant depth and use a polynomial number of edges. The class of these functions is commonly called class TC^0. Several important operations, such as iterated addition, multiplication, and the computation of symmetric functions, have been shown to belong to TC^0, but they do not belong to AC^0. Beame et al [4] and Siu et al [13] proposed efficient threshold circuits to execute these operations using threshold circuits of constant depth. In this paper, we propose several new threshold circuits to perform iterated addition and multiplication, and compute general symmetric functions, which have depths that are considerably smaller than those in [4, 13]. In particular, in Section 2 we present efficient threshold circuits to compute iterated addition. In Section 3, we use our results on iterated addition to obtain efficient threshold circuits to compute general symmetric functions. Finally, in Section 4 we present efficient methods for the construction of fast multipliers. The depth of the proposed multiplier is smaller than that of the best previously known results [13].

2 Iterated Addition

In this section, we focus on the (m, n) iterated addition problem, which is the problem of computing the sum of m n-bit integers. A related problem is the (m, n) sum-reduction problem, where we want to produce two integers whose sum is equal to the sum of the original m numbers. Both problems have been considered repeatedly in the literature, and many constructions have been proposed to execute them [4, 6, 9, 13, 17].

2.1 The (m, n) Sum-Reduction Problem

Given n one-bit numbers, a $(n, \lceil \log_2(n+1) \rceil)$-*counter* is a circuit that produces the $\lceil \log_2(n+1) \rceil$-bit binary representation of the sum of the n bits [16]. Parallel counters are important in our constructions, since they will be used as subblocks in the circuits that we will propose for the sum-reduction problem. The following lemma gives an efficient construction of parallel counters.

Lemma 1. *A $(n, \lceil \log_2(n+1) \rceil)$-counter can be constructed using a unit-weight threshold circuit of depth 2, edge complexity $n^2 + O(n)$, and fan-in n.*

Proof. We let $x_i, i = 1, 2, \ldots, n$, be the n one-bit numbers that have to be added, and $d = \lceil \log_2(n+1) \rceil$ be the number of output bits. We also denote by

$Z = (z_{d-1}, \ldots, z_0)_2$ the binary representation of $\sum_{i=1}^{n} x_i$. We then have

$$z_j = \begin{cases} 1 & \text{if } r2^j \leq \sum_{i=1}^{n} x_i < (r+1)2^j \text{ for some odd integer } r \in [1, r_j], \\ 0 & \text{otherwise,} \end{cases}$$

for $j = 0, 1, \ldots, d-1$, where $r_j = 2\lceil \lfloor n/2^j \rfloor / 2 \rceil - 1$.

To compute z_j, $j = 0, 1, ..., d-1$, we first compare $\sum_{i=1}^{n} x_i$ with the integers $1, 2, \ldots, n$, and then determine whether $\sum_{i=1}^{n} x_i$ falls in one of the intervals that make $z_j = 1$. We define

$$c_i = \begin{cases} \text{Sgn}\left(\sum_{j=1}^{n} x_j - i\right), & \text{for } i = 1, 2, \ldots, n, \\ 0, & \text{for } i > n. \end{cases}$$

The results c_i, $i = 1, 2, \ldots, n$, of the comparisons can be found using n unit-weight threshold gates of fan-in n. We then have

$$z_j = \text{Sgn}\left(\sum_{r=1,3,5,\ldots,r_j} c_{r2^j} - c_{(r+1)2^j} - 1\right), \text{ for } j = 0, 1, \ldots, d-1, \qquad (1)$$

The binary number $z_j, j = 0, 1, \ldots, d-1$, require d unit-weight threshold gates to compute, with total edge complexity $\sum_{j=0}^{d-1} \lfloor n/2^j \rfloor \leq 2n-1$. Therefore, Z can be computed using a unit-weight threshold circuit of depth 2, edge complexity no more than $n^2 + 2n - 1 = n^2 + O(n)$, and fan-in equal to n. □

We are now in a position to present circuits for the sum reduction problem. Using the techniques developed in [6, 9], we can reduce the number of operands that have to be added from $m = pr$ to $p\lceil \log_2(r+1) \rceil)$ by using $(r, \lceil \log_2(r+1) \rceil)$-counters. This reduction will be used repeatedly to reduce the number of operands. Note that the larger $r/\lceil \log_2(r+1) \rceil$ can be made, given our fan-in constraints, the faster we will be able to execute the sum-reduction problem. We define the function $f(t)$ as the unique integer $x \leq t$ that satisfies the condition

$$\begin{cases} x/\lceil \log_2(x+1) \rceil \geq y/\lceil \log_2(y+1) \rceil & \text{for all } y \in [x, t], \\ x/\lceil \log_2(x+1) \rceil > y/\lceil \log_2(y+1) \rceil & \text{for all } y < x, \end{cases}$$

The following lemma will be useful in our analysis.

Lemma 2. *The (m, n) sum-reduction problem can be executed using a unit-weight threshold circuit of depth 2, edge complexity $O(nm^3/\log m)$, and fan-in $g(m) = m \cdot 2^{\lceil \log_2(m-1) \rceil}$.*

Proof. We let x_i, $i = 1, 2, \ldots, m$, be the n-bit numbers that have to be added. The proof will be done using the "block-save technique" [12], which is a variant of the well known "carry-save adder" [17]. If we let $k = \lceil \log_2(m-1) \rceil$, then the sum $\sum_{i=1}^{m} x_i$ is no more than $2k$ bits long, since it is equal to at most $m \cdot (2^k - 1) \leq 2^{2k} - 1$.

We partition the n binary bits of the given m integers into $\lceil n/k \rceil$ groups based on their position, so that each group has k bits, except for the group containing the most significant bits, which may have less than k bits. We replace

the binary bits in group $0, 2, 4, \ldots$ with zeros and denote by S_{odd} the sum of the resultant "odd-group" part of the m integers. Similarly, we replace the binary bits in group $1, 3, 5, \ldots$ with zeros, and denote by S_{even} the sum of the resultant "even-group" part of the m integers. Since the sum $S_{odd} + S_{even}$ is equal to the sum of the given m n-bit numbers, the sum-reduction problem can be solved by finding S_{odd} and S_{even}. To do so, we simply compute the sum of each group separately and then concatenate the resultant sums of all the odd groups to get S_{odd} and of all the even groups to get S_{even}. Note that because of the definition of k, the binary representation of the sum of group i, $i = 0, 1, ..., \lceil n/k \rceil$, will not overlap with the binary representation of the sum of group $i + 2$.

One way to compute the sum of m integers using unit-weight threshold gates is to find one-bit integers whose sum is equal to the given m integers. A j-bit binary number $(x_{i,j-1}, \ldots, x_{i,p}, \ldots, x_{i,1}, x_{i,0})_2$ is equal to the sum of the following $2^j - 1$ one-bit integers,

$$x_{i,0}, x_{i,1}, x_{i,1}, \ldots, \underbrace{x_{i,p}, x_{i,p}, \ldots, x_{i,p}, x_{i,p}}_{2^p}, \ldots, \underbrace{x_{i,j-1}, x_{i,j-1}, \ldots, x_{i,j-1}, x_{i,j-1}}_{2^{j-1}},$$

where we have replaced bit $x_{i,p}$ in position p, $0 \leq p \leq j-1$, by 2^p one-bit integers, each of which is equal to $x_{i,p}$. We can then use a method similar to the one in the proof of Lemma 1 to find S_{odd} (or S_{even}, respectively). In particular, in order to compute the j^{th} bit, z_j, of S_{odd} for $j = 0, 1, \ldots, 2k-1$, we let $x_{j,i}$, $i = 1, 2, \ldots, q_j \equiv m \cdot (2^j - 1)$, be q_j one-bit integers whose sum is equal to the sum of the least significant j bits of the m integers in the group. We compare $\sum_{i=1}^{q_j} x_{j,i}, j = 1, 2, ..., 2k-1$, with the numbers $i2^j, i = 1, 2, \ldots, m-1$, using $m-1$ unit-weight threshold gates of fan-in q_j, and get

$$c_{j,i} = \mathrm{Sgn}\left(\sum_{p=1}^{q_j} x_{j,p} - i2^j\right), \text{ for } i = 1, 2, \ldots, m-1, j = 1, 2, ..., 2k-1. \quad (2)$$

We also set $c_{j,i} = 0$, for $i \geq m$, $j \neq 0$, and $c_{0,m} = \mathrm{Sgn}\left(\sum_{p=1}^{q_0} x_{0,p} - m\right)$. For $j = 0, 1, \ldots, 2k-1$, we have

$$z_j = \begin{cases} 1 & \text{if } r2^j \leq \sum_{i=1}^{n} x_{j,i} < (r+1)2^j \text{ for some odd integer } r \in [1, r_j], \\ 0 & \text{otherwise,} \end{cases}$$

where $r_j = 2\lceil \lfloor n/2^j \rfloor / 2 \rceil - 1$. Since

$$z_j = \mathrm{Sgn}\left(\sum_{r=1,3,5,\ldots,r_j} c_{r2^j} - c_{(r+1)2^j} - 1\right), \; j = 0, 1, 2, ..., k-1, \quad (3)$$

the bits z_j can be computed using $2k-1$ threshold gates of fan-in at most m. The number of edges required to compute the sum of a group is dominated by the number of edges used in layer 1, which is $mq_0 + \sum_{j=1}^{k-1}(m-1)q_j + (2m-1)q_{k-1} = O(m^3)$. The maximum number of edges used in a threshold gate is $g(m) = q_j = m(2^k - 1) = m \cdot 2^{\lceil \log_2 m \rceil}$, for $j \geq k$. The number of edges required for all $\lceil n/k \rceil$ groups is, therefore, $O(nm^3/k) = O(nm^3/\log m)$. □

The main idea of the following theorem is to use Lemma 1 repeatedly to reduce the number of operands to a small number, and then use Lemma 2 to obtain the final result.

Theorem 3. *The (m,n) sum-reduction problem can be executed using a unit-weight threshold circuit of depth no more than*

$$2\left(\lceil \log_2(d+1) - \log_2\log_2 m + \log_2\log_2 f(k)\rceil + \lceil \log_2 m/(\log_2 f(k) - \log_2\lceil \log_2 f(k)\rceil)\rceil + \Delta(k) + 1\right),$$

edge complexity $O(nm^{1+1/d})$, and fan-in k, for any fixed integer $d > 0$.

Proof. Let d be any fixed positive integer. The (m,n) sum-reduction will be performed in three phases:
Phase 1: This phase is subdivided into q stages, where

$$q = \lceil \log_2(d+1) - (\log_2\log_2 m - \log_2\log_2 f(k))\rceil. \tag{4}$$

We let $r_1 = f\left(m^{1/d}\right)$ and $r_i = f\left(m^{2^{i-1}/(d+1)}\right)$, for $i = 2,3,...,q$. At stage i, $i = 1,2,\ldots,q$, we use $(r_i, \lceil \log_2(r_i+1)\rceil)$-counters to reduce the number of operands that have to be added.
Phase 2: In this phase we use $(f(k), \lceil \log_2(f(k)+1)\rceil)$-counters to continue reducing the number of operands for another x stages until there are only

$$m_{q+x} \leq \min\left(O(m^{\frac{d+1}{4d}}\log m), g^{-1}(k)\right) \tag{5}$$

operands left, where the function $g^{-1}(k)$ is the inverse of the function $g(m)$ defined in the proof of Lemma 1.
Phase 3: At the beginning of phase 3 we are left with m_{q+x} operands that have to be added. The $(m_{q+x}, \log m_{q+x})$ sum-reduction problem can now be solved using Lemma 2. From Lemma 1, Phase 1 can be executed using a unit-weight threshold circuit of depth $2q$ and edge complexity $O(nm^{1+1/d})$, which is dominated by stage 1 for sufficiently large m. Since the fan-in of the counters in stage i is r_i, which is nondecreasing, the largest fan-in of the threshold gates used in Phase 1 is equal to r_q, where $r_q \leq m^{2^{q-1}/(d+1)} < f(k) \leq k$. From Lemma 1, the x stages of Phase 2 have depth equal to $2x$, fan-in equal to $f(k)$, and edge complexity $o(nm^{1+1/d})$. The depth required to implement Phase 3 is equal to 2. We denote by m_i the number of operands left after stage i that have to be added to obtain the result. It can be seen that an upper bound on m_i, for $i > q$, is given by

$$m_i \leq \lceil m_{i-1}/f(k)\rceil \cdot \lceil \log_2(f(k)+1)\rceil. \tag{6}$$

Since the number m_{q+x} of operands left after Phase 2 is $O(m^{\frac{d+1}{4d}}\log m)$, Phase 3 has edge complexity at most $O(nm^{1+1/d})$ from Lemma 2. Since m_{q+x} is no more than $g^{-1}(k)$, the fan-in of the circuit that implement Phase 3 is at most equal to k. Thus, the threshold circuit constructed above for the (m,n) reduction problem has depth $2(q+x+1)$, fan-in no more than k, and edge complexity at most $O(nm^{1+1/d})$.

To find the upper bound on the depth of the circuit, we distinguish two cases, depending on the magnitude of the fan-in k.

Case 1: $g^{-1}(k) = o(m^{\frac{d+1}{3d}} \log m)$, which corresponds to the case where k is not large. To derive an upper bound on x, we define three new functions $N(t), M(t)$, and $\Delta(k)$ in the following way.

The function $N(t)$ is defined recursively as

$$N(t) = (N(t-1)/f(k) + 1)\lceil \log_2(f(k)+1) \rceil, \tag{7}$$

where $N(0) = m$. Function $N(t)$ is always larger than the number of operands m_{q+t} left after stage $q+t$ as can be shown by an inductive argument. In particular, we note that $N(0) = m > m_q$, and assume (as the inductive step) that $N(t-1) > m_{q+t}$. We then have

$$N(t-1)/f(k) + 1 \geq \lceil N(t-1)/f(k) \rceil \geq \lceil m_{q+t-1}/f(k) \rceil, \tag{8}$$

which, combined with Eq. (6), gives

$$N(t) > m_{q+t}, \tag{9}$$

completing the induction. From the definition of $N(t)$ we obtain, after some algebraic manipulation, that

$$N(t) = m \cdot r^t + \frac{1-r^t}{1-r} \lceil \log_2(f(k)+1) \rceil, \tag{10}$$

where $r = \lceil \log_2(f(k)+1) \rceil / f(k)$. We also define the function $M(t)$ through the recursion

$$M(t) = (f(k) - \lceil \log_2(f(k)+1) \rceil) \lceil M(t-1)/\lceil \log_2(f(k)+1) \rceil \rceil + M(t-1),$$

where $M(0) = g^{-1}(k)$. It can be shown that the function $M(t)$ is the maximum number of operands that can be reduced to no more than $g^{-1}(k)$ operands within t steps using $(f(k), \lceil \log_2(f(k)+1) \rceil)$ parallel counters of fan-in bounded by $f(k) \leq k$ (we omit the details).

We define the function $\Delta(k)$ as the smallest integer satisfying

$$M(\Delta(k)) \geq \lceil \log_2(f(k)+1) \rceil / (1-r) + 1, \tag{11}$$

By numerically computing function $\Delta(k)$ for all k, we get

$$\Delta(k) = \begin{cases} 0 & \text{for } 56 \leq k < 146, \\ 1 & \text{for } 5 \leq k < 56, \quad \text{or } 146 \leq k < 150, \\ 2 & \text{for } 3 \leq k \leq 4. \end{cases} \tag{12}$$

It can also be shown that $\Delta(k) = 0$ for $k \geq 150$ (we omit the details). We let $\tilde{x} = \lceil \log_2 m / (\log_2 k - \log_2 \lceil \log_2(k+1) \rceil) \rceil$; the integer $\tilde{x}$ can be viewed as an approximation to x. We can derive an upper bound on $N(\tilde{x})$, where $N(\tilde{x})$ is

an upper bound on the number of operands $m_{q+\tilde{x}}$ left after $q+\tilde{x}$ stages. In particular, using Eqs. (9) and (10), we obtain

$$m_{q+\tilde{x}} < N(\tilde{x}) = m \cdot r^{\tilde{x}} + \frac{1-r^{\tilde{x}}}{1-r}\lceil \log_2(f(k)+1)\rceil < m \cdot r^{\tilde{x}} + \frac{1}{1-r}\lceil \log_2(f(k)+1)\rceil,$$

which combined with Eqs. (2) and (11) implies $m_{q+\tilde{x}} < M(\Delta(k))$. Therefore, after the first $q+\tilde{x}$ stages are performed, an additional $\Delta(k)$ stages, at most, suffice to reduce the number of operands to be less than or equal to $g^{-1}(k)$ [i.e., $m_{q+\tilde{x}+\Delta(k)} < N(\tilde{x}+\Delta(k)) \leq g^{-1}(k)$]. Therefore, we have

$$x \leq \tilde{x} + \Delta(k) = \lceil \log_2 m/(\log_2 k - \log_2\lceil \log_2(k+1)\rceil)\rceil + \Delta(k). \tag{13}$$

Case 2: $g^{-1}(k) = \Omega(m^{\frac{d+1}{3d}} \log m)$, which corresponds to k being large. It can be shown that $x = 1$ or 2 stages suffice to reduce the number of operands that have to be added to be in the order of $O(m^{\frac{d+1}{3d}} \log m)$. As a result, the inequality

$$x \leq \lceil \log_2 m/(\log_2 f(k) - \log_2\lceil \log_2 f(k)\rceil)\rceil + \Delta(k) \tag{14}$$

still holds.

Since the depth of each stage is equal to two, an upper bound on the total depth of the circuit is given by

$$\begin{aligned} 2(q+x+1) \leq 2\,(&\lceil \log_2(d+1) - \log_2\log_2 m + \log_2\log_2 f(k)\rceil + \Delta(k) + 1 \\ &+ \lceil \log_2 m/(\log_2 f(k) - \log_2\lceil \log_2 f(k)\rceil)\rceil)\,. \end{aligned} \tag{15}$$

□

The depth of our circuit is smaller than the depth of the circuit given in [4] for $k = m$ (note that the results in [4] were developed only for the case $k = m$), and is approximately 1/3 of that given in [4] for large value of m. Theorem 3 also provides a way to trade depth for edge-complexity for any integer fan-in k not exceeding m. Such flexibility is also provided in the (n,n) sum-reduction circuit presented in [13], whose depth is (according to our calculations) about $7\log_2 d\,\lceil \log_2 n/(\log_2 k - \log_2\lceil \log_2 k\rceil) + 1\rceil$.

2.2 The (m,n) Iterated Addition Problem

In this subsection, we turn our attention to the (m,n) iterated addition problems, which is the problem of computing the sum of m n-bit integers.

Lemma 4. *The sum of two n-bit integers can be computed using an AND-OR circuit of depth $2\lceil \log_2 n/\log_2 k\rceil + 3$, edge complexity $O(n^2)$, and fan-in k, assuming $k \geq \log_2 n$.*

Proof. Omitted.

For fan-in $k < \log n$, the depth of the circuit in Lemma 4 may be larger than that given in the lemma by a small constant.

Theorem 5. *The (m,n) iterated addition can be computed using a unit-weight threshold circuit of depth at most equal to*

$$2\left(\lceil \log_2(d+1) - (\log_2 \log_2 m - \log_2 \log_2 f(k))\rceil + \lceil \log_2 m/(\log_2 f(k) - \log_2\lceil \log_2 f(k)\rceil)\rceil + \Delta(k) + \lceil \log_2 n/\log_2 k\rceil + 2\right),$$

edge complexity $O(nm^{1+1/d})$, and fan-in k, for any fixed integer $d > 0$, assuming $k \geq \max(\log_2 n, 6)$.

Proof. We first use the (m,n) sum-reduction circuit of Theorem 3 to reduce the number of operands from m to two, and then compute the sum of the two numbers using the adder of Lemma 4. Since the fan-in required in the last layer of the sum-reduction circuit is no more than $g^{-1}(k)$ (which is about $\sqrt{k}$), we can combine this layer with the first layer of the adder that follows to reduce the total depth by one. The fan-ins of the gates in the combined layer are still no more than $2g^{-1}(k) \leq k$, for $k > 5$. The rest of the proof follows from Theorem 3 and Lemma 4. □

3 Symmetric Functions

In this section we propose some new threshold circuits to efficiently compute general symmetric functions. A Boolean function f is said to be *symmetric* if $f(x_1,\ldots,x_n) = f(x_{\pi(1)},\ldots,x_{\pi(n)})$ for any permutation $(x_{\pi(1)},\ldots,x_{\pi(n)})$ of $(x_1,\ldots,x_n)$. Note that a symmetric function is completely specified by the number of ones in its inputs (that is, by the sum $\sum_{i=1}^{n} x_i$).

We first restate some known facts [10, 14] in Lemmas 6 and 7.

Lemma 6. *The sum of two n-bit integers can be computed using a depth-3 AND-OR circuit of edge complexity $O(n^3)$ and fan-in n.*

Lemma 7. *Any Boolean function of n inputs can be computed in a depth 3 AND-OR circuit of gate-complexity $O(2^{n/2})$ and fan-in $O(2^{n/2})$.*

We can modify Lemma 7 to derive the following restricted fan-in version.

Lemma 8. *Any Boolean function of n inputs can be computed in a AND-OR circuit of depth $\lceil \frac{n}{2\log_2 k}\rceil + 1$, edge complexity $2^n + O(2^{n/2+1})$, and fan-in k.*

Proof. Omitted.

We are now in a position to prove the main result of this section.

Theorem 9. *Any symmetric function of n inputs can be computed using a unit-weight threshold circuit of depth no more than*

$$2\left(\lceil \log_2(d+1) - \log_2 \log_2 n + \log_2 \log_2 f(k)\rceil + \lceil \log_2 n/(\log_2 f(k) - \log_2\lceil \log_2 f(k)\rceil)\rceil + \Delta(k) + \lceil \log_2 n/\log_2 k\rceil/2\right) + 5,$$

edge complexity $O(n^{1+1/d})$, and fan-in k, for any fixed integer $d > 0$, where $\Delta(k) \leq 1$ and assuming $k \geq \max(\log_2 n, 6)$.

Proof. We can compute a symmetric function in two phases:
Phase 1: We use Theorem 3 and Lemma 6 to find the sum $\sum_{i=1}^{n} x_i$ of the inputs. Note that the output value of the symmetric function is completely determined by $\sum_{i=1}^{n} x_i$.
Phase 2: The symmetric function is transformed into a Boolean function of $\lceil \log_2(n+1) \rceil$ variables (the bits in the binary representation of $\sum_{i=1}^{n} x_i$) by Phase 1. Therefore, we can use Lemma 8 to find the desired result. The analysis is omitted. □

4 Multiplication

The results obtained in Section 2 for iterated addition give rise to a fast and edge-efficient multiplier that uses threshold gates of restricted fan-in. This multiplier is described in the following theorem.

Theorem 10. *The product of two n-bit integers can be computed using a unit-weight threshold circuit of depth at most equal to*

$$2\left(\lceil \log_2(d+1) - \log_2 \log_2 n + \log_2 \log_2 f(k) \rceil + \lceil \log_2 n/(\log_2 f(k) - \log_2 \lceil \log_2 f(k) \rceil \rceil + \Delta(k) + \lceil (\log_2 n + \lceil \log_2(n-1) \rceil - 1))/\log_2 k \rceil\right) + 5,$$

edge complexity $O(n^{2+1/d})$, and fan-in k, for any fixed integer $d > 0$, assuming $k \geq \max(\log_2 n, 6)$.

Proof. We let $X = (x_{n-1}, \ldots, x_1, x_0)_2$ and $Y = (y_{n-1}, \ldots, y_1, y_0)_2$ be the two integers that have to be multiplied. We will transform the problem into the problem of finding the sum of n n-bit numbers by using the "grade-school" method. The binary numbers

$$p_{j,i+j} \stackrel{\text{def}}{=} x_i \wedge y_j = \mathrm{Sgn}(x_i + y_j - 2), \text{ for } i = 0, 1, \ldots, n-1, \; j = 0, 1, \ldots, n-1.$$

can be computed using a unit-weight threshold circuit of depth one. We then have

$$X \cdot Y = \sum_{j=0}^{n-1} P_j, \tag{16}$$

where $P_j = (p_{j,\lceil \log_2 m_i \rceil + j - 1}, \cdots, p_{j,j}, \underbrace{0, \ldots, 0}_{j})_2$, for $j = 0, 1, \ldots, n-1$. The construction is completed by observing that the summation in Eq. (16) corresponds to an $(n, 2n-1)$ iterated addition, and using Theorem 5. □

Siu et al [13] have given threshold circuits of restricted fan-in to perform multiplication. Theorem 10 improves on the results in [13], by reducing the required depth by a factor of about $7 \log_2 d/2$ when $\log m/\log k$ is large, by a factor of about $7 \log_2 m/(2 \log_2 k - 2 \log_2 \log_2 k)$ when d is large, and by a factor of about $7 \log_2 d/4$ when $\log_2 d \approx \log_2 m/(\log_2 k - \log_2 \log_2 k)$ is large.

5 Conclusion

We have proposed several threshold circuits to perform iterated addition and multiplication, and compute symmetric functions. Our constructions give efficient tradeoff methods among the edge complexity, the circuit depth, and the maximum fan-in through the flexibility provided in the choice of the fan-in k, and in the choice of the parameter d. Our circuits appear to be considerably more depth-effective than the best previous circuit for similar edge complexity and fan-in (or, alternatively, considerably more cost-effective for similar circuit depths). A natural continuation of our work would be to construct efficient threshold circuits to execute other elementary operations such as division, powering, iterated product, and polynomial computation.

References

1. Allender, E.: A note on the power of threshold circuit. Proc. Symp. Foundations of Computer Science (1989) 580–585
2. Alon, N., Bruck, J.: Explicit constructions of depth-2 majority circuits for comparison and addition. SIAM J. Disc. Math. (1994) 1–8
3. Bruck, J.: Harmonic analysis of polynomial threshold functions. SIAM J. Disc. Math. (1990) 168–177
4. Beame, P., Brisson, E., Ladner, R.: The complexity of computing symmetric functions using threshold circuits. Theoretical Computer Science (1992) 253–265
5. Boppana, R.: Threshold functions and bounded depth monotone circuits. ACM Symp. Theor. Computing (1984) 475–479
6. Dadda, L.: Some schemes for parallel multipliers. Alta Freq. (1965) 349–356
7. Hajnal, A., Mass, W., Pudlák, P., Szegedy, M., Turán, G.: Threshold circuits of bounded depth. Proc. Symp. Foundations of Computer Science (1987) 99–110
8. Håstad, J., Goldmann, M.: On the power of small-depth threshold circuits. Computational Complexity (1991) 113–129
9. Ho, I.T., Chen, T.C.: Iterated addition by residue threshold functions and their representation by array logic. IEEE Trans. Computers **C-22** (1973) 762–767
10. Redkin, N.P.: Synthesis of threshold element networks for certain classes of boolean functions. Kibernetika. **5** (1970) 6–9
11. Reif, J., Tate, S.R.: On threshold circuits and polynomial computation. SIAM J. Comput. **21** (1992) 896–908
12. Siu, K.Y., Roychowdhury, V.P., Kailath, T.: Depth-size tradeoffs for neural computation. IEEE Trans. Computers **40** (1991) 1402–1412
13. Siu, K.Y., Roychowdhury, V.P., Kailath, T.: Toward Massively Parallel Design of Multipliers. J. Parallel Distributed Computing (1995) 86–93
14. Siu, K.Y., Roychowdhury, V.P., Kailath, T.: Discrete Neural Computation, A Theoretical Foundation. Prentice Hall, New Jersey (1994)
15. Spielman, D.A.: Computing arbitrary symmetric functions. Technical Report TR-906, Dept. of Computer Science, Yale University (1992)
16. Swartzlander, E.E.: Parallel Counters. IEEE Trans. Computers **29** (1973) 1021–1024
17. Wallace, C.S.: A suggestion for a fast multiplier. IEEE Trans. Computers **EC-13** (1964) 14–17