Toward Massively Parallel Design of Multipliers[1]

KAI-YEUNG SIU,*,[2] VWANI ROYCHOWDHURY,†,[3] AND THOMAS KAILATH‡,[4]

*Department of Electrical and Computer Engineering, University of California at Irvine, Irvine, California 92717;
†School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907; and
‡Information Systems Laboratory, Stanford University, Stanford, California 94305

Current efficient multipliers have computation delays that increase as $O(\log n)$, where n is the number of bits in the inputs. In fact, this asymptotic time complexity for multiplication cannot be improved if the fan-in of the circuits is bounded above by a constant independent of n. In this paper, we study the potential gain in the speed of arithmetic computation provided by massive parallelism (i.e., unbounded fan-in/fan-out) and examine the tradeoffs between the speed and the amount of hardware in *large but restricted fan-in threshold circuits*. Threshold circuits are chosen because the potential speed-up in AND–OR circuits for multiplication is minimal even if the fan-in is allowed to be arbitrarily large. It is well known in complexity theory that any arbitrary fan-in AND–OR circuit for multiplication of two n-bit integers must have $\Omega(\log n/\log\log n)$ computation delays if the circuit size is bounded by a polynomial in n. Threshold circuits, on the other hand, are more powerful and we present here threshold circuits for multiplication with fan-in bounded by m, size $O(2^{-d/2}n^2m^{1/(2^d-1)}\sqrt{\log m})$, and depth $O(d \log n/\log m)$ for every fixed integer $d > 0$. Similar results are also shown for symmetric functions such as the parity. With the rapid advance of VLSI technology and the promise of large scale implementations of cheap and reliable threshold gates, our results could be used for the design of massively parallel high-speed multipliers. © 1995 Academic Press, Inc.

1. INTRODUCTION

Many computationally intensive applications such as real-time signal processing rely crucially on the availability of a fast arithmetic logic unit. Any improvement in the speed of an adder or a multiplier can result in an enhanced performance of the corresponding system. Because of such fundamental considerations, several efficient designs of multipliers [4, 12–14, 18, 19] have been proposed (see, for example, [17] for a survey of such results).

One basic assumption in these conventional multiplier designs is that the *fan-in* (i.e., number of inputs to each gate) is bounded above by a constant independent of the input size n. Very often, each gate in the circuit under consideration simply has only two inputs. If such constraint on the fan-in of the multiplier circuits is imposed, then the design of the Ofman–Wallace tree [12] or the algorithm of Schonhage and Strassen [14] achieve asymptotic optimality in terms of time complexity. Indeed, any design based on a bounded fan-in circuit model that yields a circuit with $O(\log n)$ computation delays or/and $O(n)$ gate counts must be asymptotically optimal in terms of time complexity or/and circuit size, respectively.

1.1. Potential Speed-Up by Massive Parallelism

The previous discussion leaves open the question of the potential gain in the speed of arithmetic computations when large fan-in circuits are allowed. To answer this question, theoretical computer scientists have used the model of *unbounded fan-in* combinational circuits to understand fundamental issues of parallel computation. To be more specific, this computational model should be referred to as *unbounded fan-in* parallelism, since the number of inputs to each gate in the circuit is not bounded by a constant and is allowed to be arbitrarily large. The theoretical study of unbounded fan-in parallelism may give us insights into devising faster algorithms for various computational problems than would be possible with bounded fan-in parallelism. The objective of such study is to determine what can be saved with respect to hardware and time if the fan-in can be increased arbitrarily.

Clearly, the particular weighting of circuit depth, fan-in, and size that gives a realistic measure of a network's cost and speed depends on the technology used to build it. One case where circuit depth would seem to be the most important parameter and where large fan-in is feasible is when the circuit is implemented using optical devices. We refer those who are interested in the implementation of such optical devices to [1].

1.2. Threshold Circuits—Motivation

A motivation in our study of threshold circuits comes from the understanding of the computational limitation of

[1] This work is supported in part by the Joint Services Program at Stanford University (U.S. Army, U.S. Navy, U.S. Air Force) under Contract DAAL03-91-C-0010, and the NASA Headquarters, Center for Aeronautics and Space Information Sciences, under Grant NAGW-419. K.-Y. Siu's work is also partly supported by the NSF Young Investigator Awards Program under Grant MIP-9357553. V. P. Roychowdhury's work is also partly supported by NSF Grant ECS-9308814 and by NSF Grant ASC-9211073.

[2] E-mail: siu@ece.uci.edu.

[3] E-mail: vwani@ecn.purdue.edu.

[4] E-mail: tk@isl.standford.edu.

AND–OR circuits. It is known [5] that for multiplication, having large fan-in cannot in principle improve the speed of AND–OR circuits substantially. Therefore, if such a fundamental limitation is to be overcome by massive parallelism, a different model must be used. Threshold circuits become an attractive alternative.

Many logic optimization problems implicitly contain as subproblems the mapping of symmetric functions onto general purpose programmable devices. However, circuit designers have observed that they are unable to implement most symmetric functions in a small chip area with, for example, a few cascade of programmable logic arrays. This can be explained by the result that [5] many symmetric functions such as the parity require unbounded fan-in AND–OR circuits of $\Omega(\log n/\log\log n)$ depth to compute, if the circuit size is bounded by a polynomial in n, similar to the case of multipliers. Since the conventional designs of bounded fan-in circuits for multiplication [14] and for symmetric functions [10] require only $O(\log n)$ depth, we conclude that no significant speed-up in AND–OR circuits can be achieved by having large fan-in gates.

On the other hand, it is known [11] that any symmetric function of n variables can be computed by a depth-2 threshold circuit with $O(n^2)$ size. Moreover, it has been shown in [3] that all basic arithmetic operations such as multiplication of two n bit integers can be carried out in constant depth threshold circuits with size that increases as a polynomial in n (i.e., $O(n^c)$ for some constant $c > 0$). The constants in the depth of these circuits were significantly improved in [15, 15a]. Together with the above lower bound results, we can conclude that a threshold circuit based design can lead to a more efficient implementation of arithmetic and symmetric functions than an AND–OR circuit based design.

1.3. Depth–Size and Fan-in Tradeoffs in Massive Parallelism

Although unbounded fan-in threshold circuits have been designed in the literature, in many instances, the fan-in of the circuit under study increases at least linearly $\Omega(n)$ with the input size. However, the available fan-in might not be large enough to meet the requirements of a given circuit and a modified circuit might need to be designed with restricted fan-in. In the context of arithmetic circuits, issues such as how much can be gained in speed when the fan-in is not arbitrary but is restricted to, say, a small fractional power in n (i.e., n^ε for small $\varepsilon > 0$) are not well understood.

We address this problem in the paper and *derive results on the tradeoffs between the circuit size and the computation delay with a simultaneous bound on the maximum allowable fan-in (fan-out)* for circuits performing multiplication and computation of symmetric functions. For example, one would like to know the smallest possible circuitry and/or computation delay in a multiplier when the maximum allowable fan-in is $O(\sqrt{n})$ instead of $O(n)$. This problem is almost completely solved in this paper and it is shown that such circuits could have constant depth. Similar results are derived for circuits implementing symmetric Boolean functions.

Our goal in this paper is to establish the theoretical foundations for the design of efficient small depth threshold circuits for multiplication and symmetric functions. With the rapid advance of VLSI technology and the promise of large scale implementations of cheap and reliable threshold gates, we believe that our approach and results will provide circuit designers more insights into devising faster circuits.

2. DEFINITIONS AND SUMMARY OF RESULTS

For convenience of presentation, we define two important notions associated with a circuit: the *size* and the *depth*. The size and the depth give a measure of the amount of hardware and the speed of the circuit, respectively.

DEFINITION 1 (Size and Depth). In a combinational circuit, the size is the total number of edges and the depth is the number of levels.

The notion of size as defined here is not the only notion accepted in the literature. In fact, as in our related work [16], many authors define the size as the number of gates in the circuit.

DEFINITION 2 (Threshold Gate/Circuit). A *threshold gate* computes a Boolean function $f(X)$ of its inputs $X = (x_1, \ldots, x_n) \in \{0, 1\}^n$ 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 $F(X) = \sum_{i=1}^{n} w_i \cdot x_i + w_0$ and the *weights* w_i are real-valued. A *threshold circuit* is a combinational logic circuit in which each basic component is a threshold gate.

Besides multiplication, we shall study the design of threshold circuits for the class of symmetric functions.

DEFINITION 3 (Symmetric Function). A Boolean function $f\colon \{1, 0\}^n \rightarrow \{1, 0\}$ is said to be *symmetric* if

$$f(x_1, \ldots, x_n) = f(x_{(1)}, \ldots, x_{(n)})$$

for any permutation $(x_{(1)}, \ldots, x_{(n)})$ of $(x_1, \ldots, x_n)$, or equivalently, the function depends only on the sum of its input values $\sum_{i=1}^{n} x_i$.

A common example of a symmetric function is the *parity* function, for which the output is 1 if the sum of its inputs is odd, and 0 otherwise.

It turns out that an efficient design of a general symmetric function leads directly to a fast multiplier design. We are able to derive depth-size tradeoffs results, with a simultaneous bound on maximum allowable fan-in, for circuits computing symmetric functions. We should note here that apart from the usefulness of these results in the context of efficient multiplier circuits, these results could be equally useful in efficient implementations of other logic functions which use symmetric functions as basic blocks.

In Section 3, we first describe a threshold circuit that computes the parity function of n variables. Using the "periodic" structure of the parity function, our design yields a threshold circuit of fan-in m, depth $O(d \log n/\log m)$, size $O(nm^{1/(2^d-1)})$, and $O(nm^{-1/(2^d-1)})$ threshold gates for every fixed integer $d > 0$ (Theorem 1). We also show how to design a threshold circuit for parity with fewer gates but more edges. In particular, we obtain a threshold circuit of size $O(dnm^{1/d})$, depth $O(d \log n/\log m)$, fan-in bounded by m, and $O(dn/m^{1-1/d})$ threshold gates (Theorem 2).

Then we present a more general circuit design for arbitrary symmetric functions. More precisely, we prove the following result:

THEOREM 3. *Any symmetric function of n inputs can be computed using a threshold circuit of size bounded by* $O(2^{-d/2}nm^{1/(2^d-1)}\sqrt{\log m})$ *or* $O(n \log n)$ *if* $m^{1/(2^d-1)}\sqrt{\log m}) = O(\log n)$, *and depth* $O(d \log n/\log m)$ *and fan-in bounded by m, for every fixed integer* $d > 0$.

Using the results on symmetric functions in Section 3.2, we give a construction of a fast multiplier in Section 3.3. Similar to the Ofman–Wallace tree design, our multiplier is based on the fast computation of multiple addition. Our algorithm is a generalization of the carry–save technique and is a refinement of the results in [3, 15]. In particular, we obtain the following result:

THEOREM 4. *The product of two n bit integers can be computed using a threshold circuit of size* $O(2^{-d/2}n^2m^{1/(2^d-1)}\sqrt{\log m})$, *depth* $O(d \log n/\log m)$, *and fan-in bounded by m, for every fixed integer* $d > 0$.

Section 4 concludes with some open problems and directions for future research.

3. THRESHOLD CIRCUITS FOR SYMMETRIC FUNCTIONS AND MULTIPLICATION

Since a symmetric function depends only on the sum of its inputs, it is natural to approach the problem with an efficient way of computing the sum of n inputs. Thus it is not surprising that the design of circuits for symmetric functions is closely related to the design of multipliers. In fact, the idea of computing the sum of n inputs using threshold gates has already been used in [6] and our results are generalizations of this work. (For other results regarding the design of multipliers, see the comprehensive survey in [17].) Before we present the general design for an arbitrary symmetric function, we find it more instructive to present the design for a special case, the parity function.

3.1. A Parity Circuit

Besides addition and multiplication, the parity function is perhaps the most well known function in computational complexity theory. In fact, the lower bound result on the limitation of AND–OR circuits for multiplication was obtained by using the lower bound result on the parity function [5].

It follows from a result of Muroga [11] that the parity function can be computed using a depth-2 threshold circuit of size $n^2 + O(n)$. The size of the circuit was later improved by Minnick [9] to $n^2/2 + O(n)$ (with the same depth), using an ingenious "telescopic" technique. In fact, their results hold for arbitrary symmetric functions. But the fan-in of their circuits can be as large as $\Omega(n)$. In our earlier work [16], we generalized the construction in [9] to obtain a depth-d threshold circuit for parity with a near optimal number of gates, for every integer $d > 2$. Here we shall give two circuit designs; one design is for minimizing the number of *edges* and the other for minimizing the number of gates, with respect to the depth and the fan-in of the circuit. We shall use the circuits in [2, 16] as the basic building blocks in our construction for the parity circuits.

To give the basic idea of how threshold gates can overcome the limitation of AND–OR gates, we first review the classical construction of Muroga [11] in computing symmetric functions.

A Depth-2 Construction

LEMMA 1 [11]. *Let* $f(x_1, x_2, \ldots, x_n)$ *be a symmetric function. Then f can be computed using a depth-2 threshold circuit of size* $O(n^2)$.

Proof. Since f only depends on the sum of its inputs $\sum_{i=1}^n x_i$, there exists a set of s subintervals in $[0, n]$, say $[k_1, \tilde{k}_1], [k_2, \tilde{k}_2], \ldots, [k_s, \tilde{k}_s]$ (k_j's, $\tilde{k}_j$'s are integers and possibly $k_j = \tilde{k}_j$) such that

$$f(x_1, x_2, \ldots, x_n) = 1 \text{ iff } \sum_{i=1}^n x_i \in [k_j, \tilde{k}_j] \text{ for some } j.$$

On the first level, there are $2s$ threshold gates computing

$$y_{k_j} = \operatorname{sgn}\left\{\sum_{i=1}^n x_i - k_j\right\}; \tilde{y}_{k_j}$$
$$= \operatorname{sgn}\left\{\tilde{k}_j - \sum_{i=1}^n x_i\right\} \quad \text{for } j = 1, \ldots, s.$$

On the second level, the output gate computes

$$f(x_1, x_2, \ldots, x_n) = \operatorname{sgn}\left\{\sum_{j=1}^s (y_{k_j} + \tilde{y}_{k_j}) - s - 1\right\}.$$

To see that the output gate gives the correct value of the function, note the following:

If for all j, $\sum_{i=1}^{n} x_i \notin [k_j, \tilde{k}_j]$, then $y_{k_j} + \tilde{y}_{k_j} = 1$

for all $j = 1, \ldots, s$.

$$\text{Thus, } \operatorname{sgn}\left\{\sum_{j=1}^{s} (y_{k_j} + \tilde{y}_{k_j}) - s - 1\right\} = \operatorname{sgn}\{s - s - 1\} = 0.$$

On the other hand, if $\sum_{i=1}^{n} x_i \in [k_j, \tilde{k}_j]$

for some $j \in \{1, \ldots, s\}$

then $y_{k_j} + \tilde{y}_{k_j} = 2$ and $y_{k_i} + \tilde{y}_{k_i} = 1$ for $i \neq j$.

$$\text{Thus, } \operatorname{sgn}\left\{\sum_{i=1}^{s} (y_{k_i} + \tilde{y}_{k_i}) - s - 1\right\} = \operatorname{sgn}\{s + 1 - s - 1\} = 1.$$

Since s is at most $n/2$, there are at most n threshold gates in the first level. Each gate in the circuit has fan-in at most n. Thus the circuit size is $O(n^2)$. ■

Efficient Depth-d Constructions with Restricted Fan-in

If the fan-in is constrained to be not more than $m \ll n$, then we cannot directly apply the previous construction. Instead, we shall use the results in [2, 16] to construct a small constant-depth circuit to be used as a submodule in our designs. By doing so, we have reduced both the fan-in requirement and the size of the circuit.

THEOREM 1. *The parity function of n inputs can be computed using a threshold circuit of size $O(nm^{1/(2^d-1)})$, depth $O(d \log n/\log m)$, and fan-in bounded by m. Moreover, the number of threshold gates in the circuit is $O(nm^{-1/(2^d-1)})$, for every fixed integer $d > 0$.*

Proof. In [2], it was shown that for every $d > 1$, the parity function of n inputs can be computed by a threshold circuit of size $O(n^{1+1/(2^d-1)})$ and depth $O(d)$ (with fan-in bounded by n). The number of threshold gates is $O(n^{1-1/(2^d-1)})$. Our threshold circuit has the structure of an m-ary tree: each node in the tree corresponds to a subcircuit of size $O(m^{1+1/(2^d-1)})$ and depth $O(d)$ that computes the parity of its m inputs, which are the outputs from the nodes in the next lower level. At the bottom level, there are n/m nodes each computing the parity of m inputs (the inputs are the "leaves" of the m-ary tree). The single node at the top level of the tree outputs the parity of the n inputs to the circuit. If l is the number of levels of the m-ary tree, then clearly the number of leaves is m^l. Thus $l = O(\log n/\log m)$. Since there are $O(n/m)$ nodes in the tree and each node requires a depth-$O(d)$ size-$O(m^{1+1/(2^d-1)})$ threshold circuit of $O(m^{1-1/(2^d-1)})$ gates, the total circuit size and depth are $O(nm^{1/(2^d-1)})$ and $O(d \log n/\log m)$, respectively. The number of threshold gates is $O(n/m \cdot m^{1-1/(2^d-1)}) = O(nm^{-1/(2^d-1)})$. ■

It is clear that if the number of gates in the threshold circuit corresponding to each node in the above circuit can be reduced, then the overall gate count will also be reduced. Depending on the technology of implementing threshold gates, sometimes we might prefer a threshold circuit with fewer threshold gates at the expense of more edges. In fact, we can construct a threshold circuit with fewer threshold gates than the one in Theorem 1, with a small increase in the number of edges while having the same depth. We shall state the following result without proof. We refer the interested readers to our other paper [16] for the details.

LEMMA 2. *For every integer $d > 0$, we can compute the parity of m inputs using a depth-$(d + 1)$ threshold circuit of size $O(dm^{1+1/d})$ and fan-in bounded by $O(m)$. The number of threshold gates in the circuit is $O(dm^{1/d})$.*

As a result of Lemma 2, we have the following theorem:

THEOREM 2. *For every $d > 0$, the parity function of n inputs can be computed using a threshold circuit of size $O(dnm^{1/d})$, depth $O(d \log n/\log m)$ and fan-in bounded by m. Moreover, the number of threshold gates in the circuit is $O(dn/m^{1-1/d})$.*

Proof. The proof is almost the same as in Theorem 1, except that now each node in the m-ary tree corresponds to a depth $(d + 1)$ threshold circuit of size $O(dm^{1+1/d})$ and with $O(dm^{1/d})$ threshold gates. Since the number of nodes $O(n/m)$ and the number of levels $O(\log n/\log m)$ of the underlying m-ary tree remains the same, the circuit size and depth become $O(dnm^{1/d})$ and $O(d \log n/\log m)$, respectively. The total number of threshold gates in the circuit is $O(n/m \cdot dm^{1/d}) = O(dn/m^{1-1/d})$. ■

3.2. Computing a Sum of *n* Bits

The design of the parity circuit shown in the previous section relies heavily on the "periodic" structure of the parity function. For a general symmetric function, the parity circuit design based on an m-ary tree structure does not seem to apply.

On the combinational complexity of a general symmetric function, the first significant result was obtained by Muller and Preparata [10]. They showed that any n-input symmetric function can be constructed using a bounded fan-in circuit of size $O(n)$ and depth $O(\log n)$. For a bounded fan-in circuit, this construction is (within a constant factor) optimal both in circuit size and depth. For an unbounded fan-in AND–OR circuit, a technique in [3] shows that the depth of the circuit for symmetric function can be improved to $O(\log n/\log \log n)$ by having arbitrarily large fan-in gates, an improvement which can be shown to be the best possible (asymptotically).

In our related work [16], we attempted to minimize the gate count and have obtained a depth-3 threshold circuit of size $O(n^{3/2})$ (but $O(\sqrt{n})$ number of gates) for any n-input symmetric function. However, the threshold circuit has a maximum fan-in of $n + \sqrt{n}$. It can be shown that the number of threshold gates in this circuit cannot be reduced significantly without increasing the depth.

Here we shall constrain the fan-in of our circuit to be m and construct a threshold circuit for an arbitrary symmetric function that is efficient in circuit size and depth. The construction consists of two parts. In the first part, we design a threshold circuit that would output the log n-bit sum of its n inputs. The construction of the second part is easy. Since a symmetric function only depends on the sum of its inputs, which has been encoded into log n bits from the first part, we simply construct a look-up table for the value of the function, which costs at most $O(n \log n)$ circuitry.

It is easy to see from the proof of Lemma 1, one can construct a depth-2 size-$O(m^2)$ threshold circuit to compute the $O(\log m)$-bit sum of the m inputs $(x_1, x_2, \ldots, x_m)$. Actually, the size of the circuit can be significantly improved, as stated in the following lemma, which is a special case of a result in [2]. To avoid cumbersome notation, we shall assume that the sum of m bits can be expressed as a log m-bit integer (e.g., Lemmas 3 and 5).

LEMMA 3 (Beame *et al.* [2]). *Given m inputs $(x_1, x_2, \ldots, x_m)$, the log m-bit sum $\sum_{i=1}^{m} x_i$ can be computed using a threshold circuit of size $O(2^{-d/2}m^{1+1/(2^d-1)}\sqrt{\log m})$, depth $7d$, and fan-in bounded by m, for every fixed integer $d > 0$.*

A major part of our design for symmetric functions and multiplication is a threshold circuit with fan-in bounded by m that efficiently computes the sum of many integers. The basic idea is a generalization of the classical carry–save technique or the Ofman–Wallace tree. The main difficulty in computing the sum of n integers, where $n \gg m$, is to compute the carry bits in parallel. The conventional carry–save technique reduces the sum of three integers to the sum of two integers in one step, where one of the resulting two integers consists only of the carry bits. Our technique generalizes this idea and reduces the sum of m N-bit integers to the sum of log m $(N + \log m)$-bit integers using a small depth threshold circuit. Similar techniques have been used in [15]. For an expository discussion of carry–save adders or the column compression technique, see the textbook of Kuck [8]. Using the generalized carry-save techniques and the result of Lemma 5, we can derive the following lemma; the proof follows by adding the corresponding bits of the m given integers.

LEMMA 4. *Given m N-bit integers: $x_i = 2^{N-1}x_{i_{N-1}} + \cdots + 2x_{i_1} + x_{i_0}$, $i = 1, \ldots, m$. The sum $\sum_{i=1}^{m} x_i$ can be reduced to a sum of* log m *integers of size at most* $(N + \log m)$*-bits, using a depth $7d$ threshold circuit of size $O(2^{-d/2}Nm^{1+1/(2^d-1)}\sqrt{\log m})$ and fan-in bounded by m.*

Now we can construct a threshold circuit to compute the sum of n inputs using the small depth circuits in Lemma 3 and Lemma 4 as submodules.

LEMMA 5. *Given n inputs $(x_1, x_2, \ldots, x_n)$, the $O(\log n)$-bit sum $\sum_{i=1}^{n} x_i$ can be computed using a threshold circuit of size $O(2^{-d/2}nm^{1/(2^d-1)}\sqrt{\log m})$, depth $O(d \log n/\log m)$, and fan-in bounded by m for every fixed integer $d > 0$.*

Proof. For convenience, let $f_d(m) = 2^{-d/2}m^{1+1/(2^d-1)}\sqrt{\log m}$. Divide the inputs into n/m groups, each having m inputs. Compute the $(\log m)$-bit sum of each group in parallel using the circuits in Lemma 3. This step requires $7d$ levels of threshold gates and altogether $O(f_d(m) \cdot n/m) = O(2^{-d/2}nm^{1/(2^d-1)}\sqrt{\log m})$ circuit size.

After the first step, we obtain n/m integers of size $(\log m)$ bits: $x_i = 2^{\log m - 1} x_{i,\log m-1} + \cdots + 2x_{i,1} + x_{i,0}$, $i = 1, \ldots, n/m$. Further divide these n/m integers into n/m^2 groups, each having m integers of size log m bits. By Lemma 4, the sum in each group can be reduced to a sum of log m integers each of 2 log m bits. This step requires $7d$ levels of threshold gates and $O(f_d(m) \log m) = O(2^{-d/2}m^{1+1/(2^d-1)} \log^{3/2} m)$ circuitry for each group. If we perform the reduction in parallel for all n/m^2 groups of integers computed from the first step, then $O(n/m^2 \cdot 2^{-d/2}m^{1+1/(2^d-1)} \log^{3/2} m) = O(n \log^{3/2} m)$ circuitry is needed. Since each group of m integers is reduced to a group of log m integers, we have $n \log m/m^2$ $(2 \log m)$-bit integers after the second step.

This reduction procedure is iterated until the sum is reduced to the sum of $<m$ integers each of $O(\log n)$ bits. At every step, the integers are divided into groups of m integers and the reduction procedure is applied to every group. By Lemma 4, the sum of m integers of size $(k \log m)$ bits in each group can be reduced to the sum of log m integers of size $(k + 1) \log m$-bits, using a threshold circuit of depth-$7d$ and size $O(kf_d(m) \log m) = O(2^{-d/2}km^{1+1/(2^d-1)} \log^{3/2} m)$. In general, after the kth step $(k > 1)$, we will have $O(n(\log m)^{k-1}/m^k)$ remaining integers of size k log m bits, and the circuitry required in the kth step is $O(f_d(m) \cdot (k - 1) \log m \cdot n(\log m)^{k-2}/m^k) = O(nf_d(m) \cdot k \cdot (\log m)^{k-1}/m^k)$. The reduction procedure is iterated until $O(n(\log m)^{k-1}/m^k)$ drops below m, say $\bar{m}$, or when $k = O(\log n/\log m)$. Then apply Lemma 4 again with m replaced by $\bar{m}$ and N replaced by $O(\log n)$ so that there are $<$ log m remaining integers each of $O(\log n)$ bits. This would require a depth-$7d$ threshold circuit with $O(f_d(m) \log n)$ circuitry. Since each step requires $7d$ levels of threshold gates, so there are $O(d \log n/\log m)$ levels and the $O((n/m)f_d(m)) = O(2^{-d/2}nm^{1/(2^d-1)}\sqrt{\log m})$ circuitry in the first step dominates the circuit size after this sequence of reductions.

It remains to be shown how we can efficiently compute the sum of log m $O(\log n)$-bit integers: $z_1, \ldots, z_{\log m}$. Partition each $O(\log n)$-bit integer z_i into $O(\log n/\log \log m)$

consecutive blocks $\tilde{s}_{i,j}$ with each block having log log m bits, for $i = 1, \ldots, \log m$ and $j = 0, \ldots, O(\log n/\log\log m)$. Note that each block $\tilde{s}_{i,j}$ represents an integer $\leq 2^{\log\log m} - 1 = \log m - 1$. Thus the sum of all jth blocks, $\tilde{s}_j = \sum_{i=1}^{\log m} \tilde{s}_{i,j}$, can be formed as the sum of $(\log m)(\log m - 1) = O(\log^2 m)$ 1-bit number after each has been decoded, i.e., converted to unary notation. By Lemma 3, the sum of all jth blocks $\sum_{i=1}^{\log m} \tilde{s}_{i,j}$ can be computed using a threshold circuit of size $O(f_d(\log^2 m))$, depth $7d$, and fan-in bounded by $O(\log^2 m)$. Furthermore, each of these block sums $\tilde{s}_j = \sum_{i=1}^{\log m} \tilde{s}_{i,j}$ can be represented in 2 log m bits. Thus there is no overlapping in the binary representation of $\tilde{s}_j 2^j$ and $\tilde{s}_{j+2}2^{(j+2)\log m}$. Using similar arguments as in Lemma 4, we let

$$\tilde{s}_{\text{even}} = \sum_{j \text{ even}} \tilde{s}_j \cdot 2^{j \log m}$$

and

$$\tilde{s}_{\text{odd}} = \sum_{j \text{ odd}} \tilde{s}_j \cdot 2^{j \log m};$$

then the binary representation of $\tilde{s}_{\text{even}}$ is simply a concatenation of the bits in $\tilde{s}_0, \tilde{s}_2, \tilde{s}_4, \ldots$ and so forth. Similarly, a concatenation of the bits in $\tilde{s}_1, \tilde{s}_3, \tilde{s}_5, \ldots$ yields the binary representation of $\tilde{s}_{\text{odd}}$. Moreover, the original sum $\sum_{i=1}^{n} x_i = \tilde{s}_{\text{even}} + \tilde{s}_{\text{odd}}$. Since there are $O(\log n/\log\log m)$ $\tilde{s}_j$'s and each $\tilde{s}_j$ can be computed in parallel with a depth-$7d$ threshold circuit of size $O(f_d(\log^2 m))$, we only require a depth-$7d$ threshold circuit with $O(f_d(\log^2 m) \log n/\log\log m)$ size to reduce the sum to the two $O(\log n)$-bit integers $\tilde{s}_{\text{odd}}$ and $\tilde{s}_{\text{even}}$.

To compute the sum of the remaining two $O(\log n)$-bit integers, it only requires another constant depth threshold circuit of almost logarithmic size. Hence the total circuit size is dominated by the $O(2^{-d/2} n m^{1/(2^d-1)} \sqrt{\log m})$ circuitry in the first step and the circuit depth is $O(d \log n/\log m)$. ■

LEMMA 6. *Let $f(x_1, x_2, \ldots, x_n)$ be an arbitrary symmetric function. If the log n-bit sum $\sum_{i=1}^{n} x_i$ is given, then the function f can be computed using an AND–OR circuit of size $O(n \log n)$, depth $O(\log n/\log m)$ and fan-in bounded by m.*

Proof. Since a symmetric function depends only on the sum of its inputs $\sum_{i=1}^{n} x_i = \sum_{j=0}^{\log n - 1} s_j 2^j$, which is given, we can treat f as a function of log n inputs $s_0, s_1, \ldots, s_{\log n - 1}$. Express f as a logical sum-of-products form. Then the number of product terms is at most $O(2^{\log n}) = O(n)$. The first level of the circuit consists of at most n AND gates with fan-in $\leq \log n$ computing all the product terms if $m \geq \log n$. If $m < \log n$, we can replace each AND gate by an m-ary tree of AND gates with fan-in m. This will require at most $O(\log\log n/\log m)$ depth and $O(n \log n)$ circuitry. To compute the OR of these product terms, we can use an m-ary tree of OR gates with fan-in m to compute. This costs at most $2n$ circuitry and $O(\log n/\log m)$ depth. Thus the overall circuit size is $O(n \log n)$ and circuit depth is $O(\log n/\log m)$. ■

As a consequence of Lemma 5 and Lemma 6, we have the following result:

THEOREM 3. *Any symmetric function of n inputs can be computed using a threshold circuit of size bounded by $O(2^{-d/2} n m^{1/(2^d-1)} \sqrt{\log m})$ or $O(n \log n)$ if $m^{1/(2^d-1)} \sqrt{\log m} = o(\log n)$, depth $O(d \log n/\log m)$, and fan-in bounded by m for every fixed integer $d > 0$.*

Proof. Let $f(x_1, x_2, \ldots, x_n)$ be an arbitrary symmetric function. By Lemma 5, the log n-bit sum of the inputs $\sum_{i=1}^{n} x_i$ can be computed by a threshold circuit of size $O(2^{-d/2} n m^{1/(2^d-1)} \sqrt{\log m})$, depth $O(d \log n/\log m)$ and fan-in bounded by m. It follows from Lemma 6 that another depth $O(\log n/\log m)$ and size $O(n \log n)$ circuit with fan-in $\leq m$ is needed to compute the final value. ■

3.3. Fast Multiplication

Although multiplication of two n-bit integers requires any bounded depth AND–OR circuit with arbitrary fan-in to have exponential size, it was shown in [3] that many fundamental arithmetic operations including multiplication and sorting can be computed by bounded depth threshold circuits of *polynomial size* (i.e., size that grows as n^c for some $c > 0$). The depth of these polynomial size threshold circuits has been significantly reduced in [15]. Still, the circuit has very large fan-in and size $(O(n^4))$.

THEOREM 4. *The product of two n-bit integers can be computed using a threshold circuit of size $O(2^{-d/2} n^2 m^{1/(2^d-1)} \sqrt{\log m})$, depth $O(d \log n/\log m)$, and fan-in bounded by m for every fixed integer $d > 0$.*

Proof. Again for convenience, let $f_d(m) = 2^{-d/2} m^{1+1/(2^d-1)} \sqrt{\log m}$. Let the two input binary 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 binary integers $z_i = z_{i_{2n-1}} z_{i_{2n-2}} \ldots z_{i_0}$ of size $2n$ bits, for $i = 0, \ldots, n-1$, where

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

It is easy to see that the product of x and y is simply the sum of the z_i's. This step requires one level of AND–OR gates with fan-in 2 and $O(n^2)$ circuitry.

The problem has been reduced to the computation of a sum of n integers z_i of size $2n$ bits. We proceed as in Lemma 5. Divide the n integers into n/m groups, each having m integers. Apply the reduction procedure of Lemma 4 (with $N = 2n$ in this step) in parallel to each group of integers. After the reduction, there are log m integers remaining in each group. The reduction for each group requires a depth-$7d$ threshold circuit of size

$O(nf_d(m)) = O(2^{-d/2}nm^{1+1/(2^d-1)}\sqrt{\log m})$ by Lemma 4. Since there are n/m groups, there will be $n \log m/m$ remaining integers and the total circuitry required in this step is $O(nf_d(m) \cdot n/m) = O(2^{-d/2}n^2m^{1/(2^d-1)}\sqrt{\log m})$.

Now we apply the reduction procedure iteratively until we have reduced the multiple sum into a sum of $< m$ integers each with $O(n)$-bits. A simple induction shows that in general at the end of the kth step (for $k > 1$), we will have $O(n(\log m)^k/m^k)$ integers of size at most $(2n + k \log m)$ bits, and the circuitry required in the kth step is $O((2n + k \log m) \cdot n(\log m)^{k-1}/m^k \cdot f_d(m))$. So $k = O(\log n/\log m)$. Another application of Lemma 4 with N replaced by $O(n)$ reduces the sum to a sum of $< \log m$ integers each with $O(n)$-bits, using a depth-$7d$ threshold circuit of size $O(nf_d(m))$. A "block partitioning" similar argument to that in Lemma 5 further reduces this sum of $< \log m$ integers to a sum of two $O(n)$-bit integers with a depth-$7d$ threshold circuit of size $O(nf_d(\log^2 m))$. At the final step, the sum of the two integers can be computed with almost linear size and depth $O(\log n/\log m)$.

Thus the overall circuit size is dominated by the first step, which is $O(2^{-d/2}n^2m^{1/(2^d-1)}\sqrt{\log m})$. Since each reduction requires only $7d$ levels of threshold gates, the overall circuit depth is $O(d \log n/\log m)$. ■

3.4. Comparison with Other Multipliers

Since many of the well known multiplier designs (e.g., [18, 13]) are based on the bounded fan-in combinational circuit model, it is necessary that the circuit depth of these multipliers increase proportionally to $\log n$. On the other hand, our multiplier design is based on the unbounded (but restricted) fan-in circuit model. Thus for specific values of n, it is difficult to compare the merits of our design with the well known ones. Since our goal is to provide circuit designers with new theoretical insights for the potential advantages of threshold gates, we would still like to put some figures in perspective.

EXAMPLE 1. For carry–save addition, the best known result on the circuit depth based on a bounded fan-in circuit model was obtained in [13], where a circuit consisting of 3-bit full adders with depth $3.71 \log n + O(1)$ and size $O(n^{4.16})$ was derived for the multiple carry–save addition of n numbers. If we let $n = 128$, then the carry–save addition circuit in [13] has depth larger than 26. With the same value of n, the multiplier in [18] has depth of 40. In our circuit design based on threshold circuits with fan-in $m = 8$, we can derive a multiple carry–save addition circuit with gate count $O(n^2)$ and if $n = 128$, the depth of the circuit is 20.

4. CONCLUDING REMARKS

Our main contribution in this paper is the demonstration of the tradeoffs among the circuits size, maximum allowable fan-in, and the computation delay for multipliers. While previous results on unbounded fan-in circuits allow the fan-in to be arbitrarily large, we are able to derive depth–size tradeoff results of these models, with a simultaneous bound on the maximum allowable fan-in, for multipliers and for circuits computing symmetric functions. Moreover, our results appear to be the best known in the literature.

The potential speed-up in performing multiplication by increasing the fan-in of the AND–OR circuit is minimal. In fact, even if the fan-in is allowed to be arbitrarily large, the best possible computation delay in a polynomial-size AND–OR circuit for multiplication of 2 n-bit integers is $\Omega(\log n/\log \log n)$. However, with threshold circuits, we can design a multiplier that speeds up asymptotically with respect to the maximum allowable fan-in. In particular, we can construct a threshold circuit of size $O(2^{-d/2}n^2m^{1/(2^d-1)}\sqrt{\log m})$ and depth $O(d \log n/\log m)$, with fan-in at most m, for every fixed integer $d > 0$.

REFERENCES

1. Abu-Mostafa, Y. S., and Psaltis, D. Optical neural computers. *Sci. Amer.* **256,** 3 (1987), 88–95.
2. Beame, P., Brisson, E., and Ladner, R. The complexity of computing symmetric functions using threshold circuits, *Theoret. Comput. Sci.* **100,** 1 (June 1992), 253–265.
3. Chandra, A. K., Stockmeyer, L., and Vishkin, U. Constant depth reducibility. *SIAM J. Comput.* **13** (1984), 423–439.
4. Cooley, J. W., and Tukey, J. W. An algorithm for the machine computation of complex fourier series, *Math. Comp.* **19** (1965), 297–301.
5. Furst, M., Saxe, J. B., and Sipser, M. Parity, circuits and the polynomial-time hierarchy, *IEEE Symp. Found. Comput. Sci.* **22** (1981), 260–270.
6. Ho, I. T., and Chen, T. C. Multiple addition by residue threshold functions and their representation by array logic, *IEEE Trans. Comput.* **C-22** (1973), 1021–1024.
7. Khrapchenko, V. M. Asymptotic estimation of addition time of a parallel adder, *Probl. Kibernet.* **19** (1967), 107–122. English translation in *Systems Theory Res.* **19** (1970), 105–122.
8. Kuck, D. *The Structure of Computers and Computations.* Wiley, New York, 1978.
9. Minnick, R. Linear-input logic. *IEEE Trans. Electron. Comput.* **EC-10** (1961).
10. Muller, D. E., and Preparata, F. P. Bounds to complexities of networks for sorting and for switching, *J. Assoc. Comput. Mach.* **22** (1975), 195–201.
11. Muroga, S. The principle of majority decision logic elements and the complexity of their circuits. *Intl. Conf. on Information Processing.* Paris, June 1959.
12. Ofman, Yu. The algorithm complexity of discrete functions. *Sov. Phys. Dokl.* **7** (1963), 589–591.
13. Paterson, M., Pippenger, N., and Zwick, U. Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions. *Proc. 31st IEEE FOCS.* St. Louis, 1990, pp. 642–650.
14. Schonhage, A., and Strassen, V. Schnelle Multiplikation groβer Zahlen. *Computing* **7** (1971), 281–292.
15. Siu, K.-Y., and Bruck, J. Neural computation of arithmetic functions, *Proc. IEEE* **78,** 10 (Oct. 1990), 1669–1675 (Special Issue on Neural Networks).

15a. Siu, K.-Y., and Roychowdhury, V. P. On optimal depth threshold circuits for multiplication and related problems, *SIAM Journal on Discrete Mathematics* **7**, 2 (1994), 284–292.

16. Siu, K.-Y., Roychowdhury, V. P., and Kailath, T. Depth-size tradeoffs for neural computation. *IEEE Trans. Comput.* (Dec. 1991), 1402–1412 (Special Issue on Neural Networks).

17. Swartzlander, Earl E. (Ed.). *Computer Arithmetic*. IEEE Comput. Soc. Press, 1990.

18. Takagi, N., Yasuura, H., and Yajima, S. High-speed VLSI multiplication algorithm with a redundant binary addition tree. *IEEE Trans. Comput.* **C-34,** 9 (Sept. 1985), 789–796.

19. Wallace, C. S. A suggestion for a fast multiplier. *IEEE Trans. Electron. Comput.* **EC13** (1964), 14–17.

KAI-YEUNG (SUNNY) SIU received the B.Sc. degree (summa cum laude) in mathematics and computer science from New York University and the B.Eng. degree (summa cum laude) in electrical engineering from the Cooper Union, both in 1987. He pursued his graduate studies at Stanford University and received the M.Sc. degree and the Ph.D. degree in electrical engineering in 1988 and 1991, respectively. He is currently an assistant professor in the Department of Electrical and Computer Engineering of the University of California at Irvine. He was a recipient of an NSF Young Investigator Award in 1993. His research interests include artificial neural networks, parallel and distributed algorithms, and computational complexity theory.

Received March 13, 1992; revised January 18, 1994; accepted January 18, 1994

VWANI P. ROYCHOWDHURY received the B. Tech degree from the Indian Institute of Technology, Kanpur, and the Ph.D. degree from Stanford University, Stanford, in 1982 and 1989, respectively, both in electrical engineering. He is currently an assistant professor in the School of Electrical Engineering at Purdue University. From September 1989 to August 1991 he held a research associateship position in the Department of Electrical Engineering at Stanford University. His research interests include parallel algorithms and architectures, design and analysis of neural networks, special purpose computing arrays, VLSI design, and fault-tolerant computation.

THOMAS KAILATH received the S.M. degree in 1959 and the Sc.D. degree in 1961, both from the Massachusetts Institute of Technology. After a year at the Jet Propulsion Laboratories, Pasadena, he joined Stanford University as an associate professor of electrical engineering in 1963. He has been a professor since 1968, served as Director of Information Systems Laboratory from 1971 through 1980 and as Associate Chairman from 1981 to 1987, and currently holds the Hitachi America Professorship in Engineering. He is the author of *Linear Systems* (Prentice–Hall, 1980), and *Lectures on Wiener and Kalman Filtering* (Springer-Verlag, 1981). Dr. Kailath has held Guggenheim, Churchill, and Royal Society fellowships, among others, and received awards from the IEEE Information Theory Society, the IEEE Signal Processing Society, and the American Control Council. He served as President of the IEEE Information Theory Group in 1975, and holds an honorary doctorate from Linköping University, Sweden. He is a Fellow of the IEEE and of the Institute of Mathematical Statistics and is a member of the National Academy of Engineering.