Yale University
Department of Computer Science

Computing Arbitrary Symmetric Functions

Daniel A. Spielman

YALEU/DCS/TR-906
May 1992

Computing Arbitrary Symmetric Functions

Daniel A. Spielman*
Yale College

Abstract

We begin by surveying some results concerning the gate complexity of computing arbitrary boolean and symmetric functions in threshold circuits. Paturi and Saks [PS90] proved that $\Omega\left(n/\log^2 n\right)$ gates are necessary to compute parity by a depth-2 threshold circuit with bounded weights. By a theorem of Winder [Win63], the same bound holds for most symmetric functions without any restrictions on weights. Siu, Roychowdhury and Kailath [SRK91] present a construction that computes parity in depth $d+1$ using $O\left(dn^{1/d}\right)$ threshold gates. In contrast, it is implicit in the work of Lupanov [Lup71] that $2(\frac{n}{\log n})^{1/2}$ gates are necessary to compute most symmetric functions, even without bounds on weight or depth. Combining constructions of [SRK91] and [Lup71], we present circuits that achieve this bound for all symmetric functions. Comparing results of [Win63] and [SRK91], we prove that, for most sizes G, more functions can be computed by depth-3 threshold circuits of size $(\sqrt{2}+\epsilon)G$ than by unbounded-depth circuits of size G, for any $\epsilon > 0$.

It is implicit in [Win63] that $\frac{2^n}{n^2}(1-o(1))$ threshold gates are needed to compute arbitrary boolean functions in depth 2. Using a novel application of error-correcting codes, we show that arbitrary boolean functions can be computed by depth-2 threshold circuits with $3 \cdot 2^{n-1}/n$ gates. In the last section, we construct wire-efficient circuits that compute arbitrary symmetric functions. Our construction simplifies the construction of Beame, Brisson and Ladner [BBL90] and improves its constant factors.

1. Introduction

TC^0, the class of polynomial-size threshold circuits, has been the subject of much recent study [HMP+87, PS88, SRK91, BBL90, SB91]. One question of interest has

*Until September 1992, may be reached at 2037 Pine St., Philadelphia, PA 19103. After September 1992, may be reached at MIT Department of Applied Mathematics. Supported in part by NSF grant CCR-8958528 under an REU supplement.

been the complexity of computing symmetric functions [BBL90, SRK91]. A symmetric function of n variables is a function $f : \{0,1\}^n \to \{0,1\}$ such that the value of $f(x_1, \ldots, x_n)$ only depends on $\sum_{i=1}^{n} x_i$. If f is a symmetric function of n variables, we will denote by $\bar{f} : \{0, \ldots, n\} \to \{0,1\}$ the associated function such that $f(x_1, \ldots, x_n) = \bar{f}(\sum_{i=1}^{n} x_i)$. This paper is based on the intuition that efficiently computing f, a symmetric boolean function of n variables, is similar to efficiently computing $\bar{f}$, which can be considered an arbitrary boolean function of $\log n$ variables.

2. Notation

Throughout this paper, $x_1, \ldots, x_n$ will denote the boolean input variables to a circuit. We refer to the inputs as being at the "bottom" of a circuit and to the output gate as being at the "top".

By a threshold gate, we mean a gate that computes a predicate of the form

$$\sum_{i=1}^{n} w_i x_i \geq w_0,$$

where $x_1, \ldots, x_n$ are the input variables and $w_0, \ldots, w_n$ are integers specific to the gate, usually called the weights of the gate. We make no restrictions on the weights other than that they must be integers. We will use the name of a gate to refer to both the gate and to its output value. Which meaning is intended should be clear from context.

All logarithms in this paper are base 2.

3. A sum-to-binary converter

In order to treat arbitrary symmetric functions on n inputs as arbitrary boolean functions on $\log n$ inputs, we must be able to obtain the binary representation of the sum of the inputs. In this section, we explain work by [SRK91] that enables us to do this. The "sum-to-binary" converters that we obtain will be useful in both the gate efficient and wire efficient computation of arbitrary symmetric functions.

Definition 1. A function $f(x_1, \ldots, x_n)$ is *linearly-computable* from a threshold circuit C, if there exist gates $\mathbf{g}_1, \ldots, \mathbf{g}_m$ in C and weights $w_0, \ldots, w_m, v_1, \ldots, v_n$ such that

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{m} w_i \mathbf{g}_i + \sum_{i=1}^{n} v_i x_i - w_0.$$

When talking about linear computability, we will use the term "circuit" to refer to collections of gates, regardless of whether or not they have only one output gate. We

use this loose definition because we will want to talk about functions being linearly computable from sub-circuits of a circuit under construction. If a function is linearly computable from a sub-circuit of a circuit, then there is no need to have a gate representing that function: it can just be incorporated into the input of another threshold gate as the linear sum. An example of linear computability appears in the following well-known lemma:

Lemma 2 ([HMP+87]). *Let C be a circuit that contains gates $\mathbf{g}_1, \ldots, \mathbf{g}_n$ and let $f(x_1, \ldots, x_n)$ be a symmetric function. Moreover, assume that $\mathbf{g}_j = 1$ if and only if $\sum_{i=1}^{n} x_i \leq j$. Then $f(x_1, \ldots, x_n)$ is linearly computable from C.*

Proof: Let $w_0 = \bar{f}(0)$. For $1 \leq i \leq n$, let

$$w_i = \begin{cases} 0, & \text{if } \bar{f}(i) = \bar{f}(i-1); \\ 1, & \text{if } \bar{f}(i) \neq \bar{f}(i-1) \text{ and } \bar{f}(i) = 1; \\ -1, & \text{if } \bar{f}(i) \neq \bar{f}(i-1) \text{ and } \bar{f}(i) = 0. \end{cases}$$

Then, it is clear that $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \mathbf{g}_i w_i + w_0$. ∎

For completeness, we present a circuit from [SRK91] from which the bits of the binary representation of $\sum_{i=1}^{n} x_i$ are linearly-computable. For convenience, we will let $X = \sum_{i=1}^{n} x_i$.

Let d be a natural number and let e be the least integer such that $2^e > n^{1/d}$. We construct a family of circuits $\mathbf{B}_d$ where the circuit for inputs of length n has gates $\mathbf{g}_{j,k}$ for $1 \leq j \leq d$ and $1 \leq k \leq 2^e - 1$. Gate $\mathbf{g}_{j,k}$ computes the predicate

$$X - \sum_{j'=j+1}^{d} 2^{e(j'-1)} \left(\sum_{k'=1}^{2^e-1} \mathbf{g}_{j',k'} \right) \geq k 2^{e(j-1)}. \tag{1}$$

It is easy to see that these gates can be arranged into d levels, with the $\mathbf{g}_{d,k}$'s at the bottom and the $\mathbf{g}_{1,k}$'s at the top.

Theorem 3 ([SRK91]). *For every natural number d, $\mathbf{B}_d$ is a circuit of depth d with at most $2dn^{1/d}$ gates from which the binary representation of $\sum_{i=1}^{n} x_i$ is linearly computable.*

We provide a proof in Appendix A.

4. Unbounded Depth

Let $B(n)$ be the number of threshold gates needed to compute an arbitrary boolean function of n variables and let $S(n)$ be the number of threshold gates needed to compute an arbitrary symmetric function of n variables. Lupanov [Lup71, Lup73] proves:

Theorem 4 (Lupanov).

$$B(n) \sim 2\left(\frac{2^n}{n}\right)^{1/2}.$$

Corollary 5.

$$S(n) \sim 2\left(\frac{n}{\log n}\right)^{1/2}.$$

Proof: The lower bound is well known: one can consider any boolean function of n variables as a symmetric function of 2^n variables: just assign weight 2^{i-1} to input x_i. This map from inputs in $\{0,1\}^n$ to integers in $\{0,\ldots,2^n-1\}$ is clearly injective. So, if any symmetric function of 2^n variables can be computed by G gates, then any boolean function of n variables can be computed by G gates. This implies that $B(n) \leq S(2^n)$.

For the upper bound, use the sum-to-binary converter $\mathbf{B}_3$ from Section 3 to convert the sum of the n inputs to binary. This requires at most $6n^{1/3}$ gates. Since the sum is represented by at most $\log n$ binary bits, Theorem 4 says that we can compute any function of these bits using $B(\log n)$ gates. Thus, we see that $S(n) \leq B(\log n) + 6n^{1/3}$. ∎

To achieve the number of gates used in Theorem 4, Lupanov uses unbounded depth. However, his construction can be used to compute any boolean function in depth 4 using $O\left(\sqrt{2^n/n}\right)$ threshold gates. It follows that any symmetric function can be computed in depth 7 using $O\left(\sqrt{n/\log n}\right)$ threshold gates. It would be interesting to know whether this depth can be improved.

5. Unbounded depth vs. depth 3

The following theorems demonstrate that for for most sizes, G, more functions can be computed by depth-3 circuits of size $(\sqrt{2}+\epsilon)G$ than by unbounded-depth circuits of size G, for any $\epsilon > 0$.

Theorem 6 ([Win63]). *A threshold circuit of size G can realize one of at most $2^{\frac{G^2 n}{2}+Gn^2}$ different functions.*

Let $x_1,\ldots,x_n$ be the inputs to a circuit. Let $X = \sum_{i=1}^{n} 2^{i-1}x_i$. We see that functions mapping $(x_1,\ldots,x_n) \in \{0,1\}^n$ into $\{0,1\}$ are in one-to-one correspondence with functions mapping $X \in \{0,\ldots,2^n-1\}$ into $\{0,1\}$. Let $\bar{f} : \{0,\ldots,2^n-1\} \to \{0,1\}$. We say that $\bar{f}$ *changes value at i* if $\bar{f}(i) \neq \bar{f}(i+1)$.

Siu, Roychowdhury and Kailath prove:

Theorem 7 ([SRK91]). *If $\bar{f}(X)$ changes value at most m times as X goes from 0 to $2^n - 1$, then the corresponding function f can be computed by a depth-3 threshold circuit of size $2\lceil\sqrt{m}\rceil + O(1)$.*

For completeness, we provide a proof in Appendix B.

Corollary 8. *There are at least $\binom{2^n}{G^2(n)}$ distinct functions computable by depth-3 threshold circuits of size $2G(n) + O(1)$.*

Proof: By Theorem 7, we see that any function that alternates value at most $G^2(n)$ times can be computed by a depth-3 threshold circuit of size $2G(n) + O(1)$. There are at least $\binom{2^n}{G^2(n)}$ such functions. ∎

Comparing the values obtained by Corollary 8 and Theorem 6, we get that:

Theorem 9. *Let $\epsilon > 0$ and let $G(n)$ be a function such that $n = o(G(n))$ and $G(n) = 2^{o(n)}$. Then, for almost all n, there are more functions computable by depth-3 threshold circuits of size $(\sqrt{2} + \epsilon)G(n)$ than by unbounded-depth threshold circuits of size $G(n)$.*

Proof: We will prove the equivalent assertion that for almost all n, there are more functions computable by depth-3 threshold circuits of size $2G(n) + O(1)$ than by unbounded-depth threshold circuits of size $(\sqrt{2} - \epsilon)G(n)$. For convenience, we will use G to denote $G(n)$.

By Corollary 8, the number of functions computable by depth-3 threshold circuits of size $2G + O(1)$ is

$$\begin{aligned}
\binom{2^n}{G^2} &= \frac{(2^n)^{2^n}\sqrt{2^n}}{(2^n - G^2)^{2^n - G^2}(G^2)^{G^2}\sqrt{2^n - G^2}\sqrt{2\pi G^2}}\theta(1) \qquad \text{by Stirling's formula} \\
&\geq \left(1 + \frac{G^2}{2^n - G^2}\right)^{2^n - G^2}\left(\frac{2^n}{G^2}\right)^{G^2}\frac{1}{\sqrt{2\pi G^2}}\theta(1) \\
&\geq 2^{G^2\cdot(n - 2\log G + \log e) - \log G - O(1)}
\end{aligned}$$

We compare this with the number of functions computable by unbounded-depth threshold circuits of size $(\sqrt{2} - \epsilon)G(n)$ from Theorem 6:

$$2^{\frac{((\sqrt{2}-\epsilon)G)^2 n}{2} + (\sqrt{2}-\epsilon)Gn^2} = 2^{\delta G^2 n + (\sqrt{2}-\epsilon)Gn^2}$$

for some $\delta < 1$, which proves the theorem. ∎

6. Depth 2

Depth-2 threshold circuits require many more gates to compute most functions than do depth-3 threshold circuits. Paturi and Saks [PS90] proved that any depth-2 threshold circuit with polynomially bounded weights that computes parity requires $\Omega\left(n/\log^2 n\right)$ gates. In fact, most symmetric functions of n variables require this many gates, even without restrictions on the weights. The best bound that we know on the number of gates that a depth-2 threshold circuit with unbounded weights needs to compute parity is $\Omega\left(\sqrt{n}\right)$ from [PS90].

R.O. Winder [Win63] proved the following theorem:

Theorem 10 (Winder). *The class of depth-2 threshold circuits with G gates can compute at most $2^{G(n^2+n)+n^2}$ functions on n input variables.*

Corollary 11. *Most functions of n variables require $\frac{2^n}{n^2}(1-o(1))$ gates to be computed by a depth-2 threshold circuit.*

Proof: There are 2^{2^n} boolean functions of n variables. In order for every one of them to be computable by a threshold circuit of G gates, we must have that $2^{2^n} \leq 2^{G(n^2+n)+n^2}$, which implies that $G \geq (2^n - n^2)/(n^2+n)$. ∎

Corollary 12. *Most symmetric functions of n variables require $\frac{n}{\log^2 n}(1-o(1))$ gates to be computed by a depth-2 threshold circuit.*

Proof: The reduction is the same as that used in the lower bound of Theorem 5. ∎

The best upper bound that we know for computing symmetric functions by depth-2 threshold circuits is the obvious linear upper bound from [HMP+87]. Although we do not know of a construction matching Winder's lower bound for depth-2 circuits, we can come close to Winder's bound. This suggests that linear lower bounds on the number of threshold gates necessary to compute symmetric functions in depth-2 threshold circuits will not be achieved by examination of the arbitrary boolean case.

Theorem 13. *If $n = 2^d - 1$ for some integer d, then every boolean function of n variables can be computed by a depth-2 threshold circuit with at most $\frac{3}{2}2^n/n$ gates.*

Proof: For this construction, we let the inputs be points (represented as vectors) on the $\{1,-1\}^n$ hypercube.

Let f be a boolean function. From the theory of error correcting codes [Tho83], we know that an n-dimensional hypercube can be exactly covered by $2^n/(n+1)$ Hamming balls of radius 1, provided that $n = 2^d - 1$ for some integer d. Let the points $\left\{\vec{c}_1, \ldots, \vec{c}_{2^n/(n+1)}\right\}$ be the set of centers of such a set of Hamming balls, where

$\vec{c_j} = (c_{j,1}, \ldots, c_{j,n})$. The Hamming ball with center c_j is separated from the rest of the hypercube by the hyperplane $\sum_{i=1}^{n} x_i c_{j,i} \geq n - 2$. For all j, let $\mathbf{h}_j$ be the gate that computes the predicate

$$\sum_{i=1}^{n} c_{j,i} x_i \geq n - 2.$$

Let $\vec{e_i}$ denote the vector that is 1 in every place except the ith index, in which it is -1. If we use componentwise multiplication, then multiplication by $\vec{e_i}$ flips the ith bit of a vector. For $1 \leq j \leq 2^n/(n+1)$ and $1 \leq i \leq n$, let

$$w_{j,i} = \begin{cases} c_{j,i}, & \text{if } f(\vec{c_j}\vec{e_i}) = f(\vec{c_j}); \\ 2c_{j,i}, & \text{otherwise.} \end{cases}$$

Let $w_{j,0} = \sum_{i=1}^{n} |w_{j,i}| - 2$. Thus, if x differs from c_j in $k+1$ places, then $\sum_{i=1}^{n} w_{j,i} x_i \leq w_{j,0} - 2k$; if $x = e_i \vec{c_j}$ for some i and $f(x) = f(c_j)$, then $\sum_{i=1}^{n} w_{j,i} x_i = w_{j,0}$; if $x = e_i \vec{c_j}$ for some i and $f(x) \neq f(c_j)$, then $\sum_{i=1}^{n} w_{j,i} x_i = w_{j,0} - 2$. For $1 \leq j \leq 2^n/(n+1)$, let gate $\mathbf{g}_j$ compute the predicate

$$\sum_{i=1}^{n} w_{j,i} x_i \geq w_{j,0}.$$

Note that it will never be the case that $\mathbf{h}_i = 1$ and $\mathbf{h}_j = 1$ for $i \neq j$, and that $\mathbf{g}_i = 1$ implies that $\mathbf{h}_i = 1$ for all i. Let $r = \operatorname{sgn}\left(\sum_j f(\vec{c_j})\right)$. (We could choose r to be ± 1, but this choice is optimal.) We now note that f is linearly computable from the $\mathbf{g}_j$'s and the $\mathbf{h}_j$'s because $f(\vec{x}) = r$ if and only if

$$\sum_{\{j | f(\vec{c_j}) = r\}} \mathbf{g}_j + \sum_{\{j | f(\vec{c_j}) = -r\}} (\mathbf{h}_j - \mathbf{g}_j) = 1.$$

Observe that we chose r so that gate $\mathbf{h}_j$ need exist for only $1/2$ of the j's. ∎

As far as we know, this upper bound is new and the best known.

Recall that in the unbounded-depth case, the upper and lower bounds match. We conjecture that there is a better construction which achieves the upper bound of $O(2^n/n^2)$ gates suggested by the lower bound in Corollary 11.

7. Wire-efficient constructions

Instead of considering the number of gates, one can study the number of wires needed to compute a function with a threshold circuit. In this model, each wire is only allowed to have weight ± 1. Thus, to have weight w from an input, it is necessary to have w wires from that input. For example, we provide the following lemma concerning the sum-to-binary converters from Section 3:

Lemma 14. *Circuit $\mathbf{B}_d$ uses at most $4dn^{1+1/d}(1+o(1))$ wires. Moreover, any bit in the binary representation of the sum of the inputs to $\mathbf{B}_d$ may be linearly computed with at most $2n^{1/d}$ wires.*

Proof: Every time an input wire appears, it has weight 1. Since there are at most $2dn^{1/d}$ gates in the circuit, wires from the inputs can account for at most $2dn^{1+1/d}$ wires. Similarly, there are $2n^{1/d}$ threshold gates that have output wires of weight $n^{1-1/d}$; these can account for at most another $2dn^{1+1/d}$ wires. Since the weights from the wires on the remaining gates have a smaller exponent, their contribution is negligible. That at most $2n^{1/d}$ wires are necessary to linearly compute any of the bits of the binary representation follows from Lemma 2 and that only one row of gates need be examined to compute any one bit. ∎

Beame, Brisson and Ladner [BBL90] present a wire-efficient construction of threshold circuits that compute arbitrary symmetric functions. Their construction also uses the trick of computing the binary representation of the sum of the inputs. To save wires, they use a recursive construction that divides the inputs into several sections, computes the binary representation of the sum in each section, and then uses a binary adder to find the sum over all the sections. In this section, we present a simplified version of their construction which takes advantage of the sum-to-binary converters from Section 3. Independently, [SRK92b] have improved upon the results in [BBL90]; however, their results are weaker than those presented here.

In order to minimize both gates and wires used in our circuit, it will be necessary to have a construction for computing arbitrary boolean functions that is efficient in both wires and depth. We derive such a circuit below. We do not use Lupanov's circuit mentioned in Theorem 4 because we are not sure of the number of wires used in his construction. The theorem we use is originally due to Redkin [Red70]. We present a proof for completeness.

Theorem 15 (Redkin). *Any boolean function f of n variables can be computed by a depth-3 AND/OR circuit having $2(2^{n/2})+1$ gates if n is even and $\frac{3}{\sqrt{2}}2^{n/2}+1$ gates if n is odd.*

Proof: For $x, \sigma \in \{0,1\}$, we define

$$x^\sigma = \begin{cases} x, & \text{if } \sigma = 1; \\ 1-x, & \text{if } \sigma = 0. \end{cases}$$

Let $x_1, \ldots, x_n$ be the input variables. Let $m = \lceil n/2 \rceil$. By a disjunction of the variables $x_1, \ldots, x_m$ we mean a term of the form $\vee_{i=1}^m x_i^{\sigma_i}$, where $\sigma_i \in \{0,1\}$ for $1 \leq i \leq m$. The bottom level of our circuit will consist of gates computing all the disjunctions of $x_1, \ldots, x_m$. We will call $g_{(\sigma_1,\ldots,\sigma_m)}$ the gate computing the disjunction corresponding to $(\sigma_1, \ldots, \sigma_m) \in \{0,1\}^m$. Thus, the bottom level of the circuit requires 2^m gates.

Observe that any function $f : \{0,1\}^m \to \{0,1\}$ can be expressed in disjunctive normal form as

$$f(x_1, \ldots, x_m) = \bigwedge_{\{(\sigma_1,\ldots,\sigma_m)|f(\bar{\sigma}_1,\ldots,\bar{\sigma}_m)=0\}} \vee_{i=1}^{m} x_i^{\sigma_i}.$$

Accordingly, the second level of our circuit will consist of gates $h_{(\tau_{m+1},\tau_{m+2},\ldots,\tau_n)}$ where $(\tau_{m+1}, \ldots, \tau_n) \in \{0,1\}^{n-m}$. Gate $h_{(\tau_{m+1},\tau_{m+2},\ldots,\tau_n)}$ will compute the predicate

$$\bigwedge_{i=m+1}^{n} x_i^{\tau_i} \bigwedge_{\{(\sigma_1,\ldots,\sigma_m)|F(\bar{\sigma}_1,\ldots,\bar{\sigma}_m,\tau_{m+1},\ldots,\tau_n)=0\}} g_{(\sigma_1,\ldots,\sigma_m)}.$$

That is, gate $h_{(\tau_{m+1},\tau_{m+2},\ldots,\tau_n)}$ computes

$$\begin{cases} f(x_1, \ldots, x_n), & \text{if } (x_{m+1}, \ldots x_n) = (\tau_{m+1}, \ldots \tau_n); \\ 0, & \text{otherwise.} \end{cases}$$

The root of the circuit then computes the predicate

$$\bigvee_{(\tau_{m+1},\ldots,\tau_n)\in\{0,1\}^{n-m}} h_{(\tau_{m+1},\ldots,\tau_n)}$$

The second level uses only 2^{n-m} gates. Thus, the circuit uses $2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil} + 1$ gates. ∎

Corollary 16. *Any boolean function f of n variables can be computed by a depth-3 threshold circuit having $\frac{3}{\sqrt{2}}(2^{n/2})$ gates and $2^n(1 + o(1))$ wires.*

Proof: Each AND/OR gate can be computed as a threshold gate. The wire bound comes from counting the number of wires in the above construction. ∎

As the top level of our circuit, we will use this corollary to compute $\bar{f}$ with $\frac{3}{\sqrt{2}} n^{1/2}$ gates and $n(1 + o(1))$ wires.

8. A Binary Adder

Beame, Brisson and Ladner prove that

Theorem 17 ([BBL90]). *For all $d \geq 2$ and $n \geq 8^{2^d}$ for any n input symmetric function f there is a threshold circuit computing f with at most*

$$150\sqrt{1 + \frac{\log n}{2^d - 1}} \cdot n^{1+\frac{1}{2^d-1}} + O\left(\frac{n}{\log n}\right)$$

wires and depth bounded by $7d + 6$.

Their construction uses at least $n/(\log n)$ gates. Our construction will use their binary addition trick; however, we will use a system of addition that, instead of keeping two partial sums at each level, keeps many partial sums present up until the last level. This enables us to save on the number of wires and gates used.

Assume that we have $m-1$ numbers, $\vec{x}_1, \ldots, \vec{x}_{m-1}$, each of length n. Let $x_{i,1}, \ldots, x_{i,n}$ be the binary representation of $\vec{x}_i$ (i.e., $\vec{x}_i = \sum_{j=1}^{n} 2^{j-1} x_{i,j}$). For $1 \leq j \leq n$, let $\vec{s}_j$ be the column sums, $\vec{s}_j = \sum_{i=1}^{m-1} x_{i,j}$, and let $s_{j,1}, \ldots, s_{j,\log m}$ be the binary representation of $\vec{s}_j$. Observe that $\sum_{i=1}^{m-1} \vec{x}_i = \sum_{j=1}^{n} 2^{j-1} s_j$. Thus, we can reduce the problem of adding m numbers of n bits each to the problem of adding n staggered numbers of $\log m$ bits each:

$$\sum_{i=1}^{m-1} \vec{x}_i = \sum_{j=1}^{n} 2^{j-1} \left(\sum_{k=1}^{\log m} 2^{k-1} s_{j,k} \right).$$

We will call this process *folding* the $m-1$ numbers of length n *into* n staggered numbers of length $\log m$.

Proposition 18. *We can fold $m-1$ numbers of length n into n staggered numbers of length $\log m$ in depth 3 using at most $n2d(m-1)^{1/d}$ gates and $n4d(m-1)^{1+1/d}(1+o(1))$ wires.*

Proof: We use the circuits $\mathbf{B}_2$ from Section 3 to obtain the binary representation of the $\vec{s}_j$'s. Since we will actually want gates whose outputs are the $s_{j,k}$'s, this will require depth 3. By Theorem 3 and Lemma 14, folding the summands will require at most $2dn(m-1)^{1/d} + n \log m$ gates and $4dn(m-1)^{1+1/d}(1+o(1))$ wires. ∎

By performing this process recursively, we obtain:

Theorem 19. *Let $\epsilon > 0$ and let a be a positive integer. Let $\delta = 1/3 + \epsilon$. Then, there exists a circuit of depth $3a+5$ that uses $4n^{1-\frac{1}{2}\delta^a}(1+o(n^{-\frac{1}{2}\delta^a}))$ gates and $8n^{1+\frac{1}{2}\delta^a}+o(n)$ wires to compute the binary representation of the sum of the inputs $x_1, \ldots, x_n$.*

Proof: At the bottom level of the circuit, we break the inputs, $x_1, \ldots, x_n$, into $n^{1-\delta^a}$ groups of size n^{δ^a}. For each of these groups, we compute the binary representation of the sum of the inputs in the group using $\mathbf{B}_2$. We will want to put gates on top of the $\mathbf{B}_2$'s that explicitly provide the binary representation. Thus, by Theorem 3 and Lemma 14, each copy of $\mathbf{B}_2$ and the level on top uses $4n^{\frac{1}{2}\delta^a} + \delta^a \log n$ gates and $8n^{\frac{3}{2}\delta^a} + 2\delta^a(\log n)n^{\frac{1}{2}\delta^a}$ wires. This yields a total of $4n^{1-\frac{1}{2}\delta^a} + \delta^a(\log n)n^{1-\delta^a}$ gates and $8n^{1+\frac{1}{2}\delta^a} + 2\delta^a(\log n)n^{1-\frac{1}{2}\delta^a}$ wires in depth 3.

The goal of the rest of the circuit is to obtain the sum of the resulting $n^{1-\delta^a}$ subsums. We will do this in a stages. The first stage above these subsums will take groups of $n^{\delta^{a-1}(1-\delta)}$ subsums of length $\delta^a(\log n)$ and fold them into $\delta^a(\log n)$ staggered numbers of length $\delta^{a-1}(1-\delta)(\log n)+1$. We will perform the folding with circuits $\mathbf{B}_2$. By Proposition 18, this will use $4\delta^a(\log n)(n^{\delta^{a-1}(1-\delta)})^{1/2} + \delta^{a^2-a}(1-\delta)(\log^2 n)$

gates, $8\delta^a(\log n)\left(n^{\delta^{a-1}(1-\delta)}(1+o(1))\right)^{3/2}$ wires and depth 3. Since there are $n^{1-\delta^{a-1}}$ such groups, a total of at most

$$4\delta^a(\log n)(n^{1-\delta^{a-1}(1+\delta)/2}) + O\left(n^{1-\delta^{a-1}}\log^2 n\right)$$

gates and

$$8\delta^a(\log n)\left(n^{1-\delta^{a-1}(1-\frac{3}{2}(1-\delta))}(1+o(1))\right) = o(n)$$

wires are used at the first stage. (Recall that $(1-\delta) < 2/3$.)

The sum of the inputs in each group can be obtained as the sum of $\delta^a(\log n)$ staggered numbers of length $\delta^{a-1}(1-\delta)(\log n)+1$; however, we will overestimate and assume that each sum can be obtained as the sum of $\delta^a(\log n)$ numbers of length $\delta^{a-1}(\log n)$. We will keep in mind that their sum is at most $n^{\delta^{a-1}}$.

For the remaining $a-1$ stages, we proceed by induction. Assume that, at stage i, we want to compute the sum of at most $\delta^{a-i+2}(\log n)$ binary numbers of length $\delta^{a-i+1}(\log n)$, but whose sum is at most $n^{d^{a-i+1}}$. Then, to get to the $(i+1)$st stage, we take collections of $n^{\delta^{a-i}(1-\delta)}$ numbers to fold together. Since each number is a sum of $\delta^{a-i+2}(\log n)$ binary numbers, we are folding $\delta^{a-i+2}(\log n)n^{\delta^{a-i}(1-\delta)}$ numbers of length $\delta^{a-i+1}(\log n)$ into $\delta^{a-i+1}(\log n)$ staggered numbers of length at most $\delta^{a-i}(1-\delta)(\log n) + \log\log n$. We overestimate this as the sum of $\delta^{a-i+1}(\log n)$ binary numbers of length $\delta^{a-i}(\log n) + \log\log n$. To complete the induction, we recall that we were summing $n^{\delta^{a-i}(1-\delta)}$ terms of length at most $\delta^{a-i+1}(\log n)$; so, we can assume that the length of each of the $\delta^{a-i+1}(\log n)$ binary numbers is at most $\delta^{a-i}(\log n)$.

If we always use $\mathbf{B}_2$'s to do the folding, then for each group, we use depth 3 and at most

$$4\delta^{a-i+1}(\log n)\left(\delta^{a-i+2}(\log n)n^{\delta^{a-i}(1-\delta)}\right)^{1/2}$$

gates, and at most

$$8\delta^{a-i+1}(\log n)\left(\delta^{a-i+2}(\log n)n^{\delta^{a-i}(1-\delta)}\right)^{3/2}$$

wires.

Since there are at most $n^{1-\delta^{a-1}}$ such groups, we use depth 3 and a total of at most

$$4\delta^{a-i+1}(\log n)\left(\delta^{a-i+2}(\log n)\right)^{1/2} n^{1-\delta^{a-i}\frac{1+\delta}{2}}$$

gates, and at most

$$8\delta^{a-i+1}(\log n)\left(\delta^{a-i+2}(\log n)\right)^{3/2} n^{1-\delta^{a-i}(1-\frac{3}{2}(1-\delta))} = o(n)$$

wires.

After the ath stage, we are left with $\delta\log n$ binary numbers, each of which is less than n, and whose sum is the sum of the inputs $x_1,\ldots,x_n$.

We now create a depth-2 circuit that obtains their sum. Call the $\delta\log n$ binary numbers $r_1,\ldots,r_{\delta\log n}$. Let $r_j = \sum_{i=1}^{\log n} 2^{i-1}r_{j,i}$ be their binary representation. Let

$b_1, \ldots, b_{\log(n+1)}$ be the binary representation of $\sum_{i=1}^{n} x_n$. For any l, $1 \leq l \leq \log n$, b_l can be computed as

$$b_l = \left(\left(\sum_{j=1}^{\delta \log n} (\vec{r}_j \bmod 2^l) \right) \operatorname{div} 2^{l-1} \right) \bmod 2 = \left(\left(\sum_{j=1}^{\delta \log n} \sum_{i=1}^{l-1} 2^i r_{j,i} \right) \operatorname{div} 2^{l-1} \right) \bmod 2$$

Since $\sum_{j=1}^{\delta \log n} (\vec{r}_j \bmod 2^l) \leq 2^l \delta \log n$, b_l can be computed by a depth-2 circuit with at most $2\delta \log n + 1$ gates and $2^{l+1}(\delta \log n)^2 + (\delta \log n)$ wires.

Thus, to compute all the b_l's from the $\vec{r}_j$'s will require at most $(2\delta \log n + 1)\log(n+1)$ gates and $2^{l+2}(\delta \log n)^2 + (\log(n+1))(\delta \log n)$ wires. ∎

Note that instead of using the circuits $\mathbf{B_2}$ in the above construction, we could have used $\mathbf{B}_d$ for some $d > 2$. This would enable use to choose δ as small as $1/(d+1) + \epsilon$. By adjusting d and δ, it is possible to trade increases in depth for constant-factor reductions in the exponents of wires and gates.

Corollary 20. *Let $\delta > 1/3$ and let a be a positive integer. Let f be a symmetric function of n inputs. Then, there exists a circuit of depth $3a + 8$ that uses $4n^{1-\frac{1}{2}\delta^a}(1 + o(n^{-\frac{1}{2}\delta^a}))$ gates and $8n^{1+\frac{1}{2}\delta^a} + o(n)$ wires to compute f.*

Proof: Combine Theorem 19 with Corollary 16. ∎

Corollary 21. *Let $\delta > 1/3$. Let f be a symmetric function of n inputs. Then, there exists a circuit of depth $\frac{3}{\log(3-\epsilon)} \log\log n + 8$ that uses $O(n)$ gates and $O(n)$ wires to compute f.*

Proof: Let $a = \log_{(1/d)} \log n$ in the preceding corollary. ∎

9. Concluding remarks

It is interesting to note that Paturi and Saks's [PS90] lower bound for the number of threshold gates needed to compute parity in a depth-2 threshold circuit agrees with the bound obtained for most symmetric functions. Moreover, we do not know of any non-trivial symmetric function that changes value t times yet which can be computed by a depth-2 threshold circuit of size less than $t/2$. Any example of such a function, or any improvement of Paturi and Saks's lower bound would be very interesting.

To obtain threshold circuits that compute arbitrary symmetric functions with a linear number of wires, we needed to use a linear number of gates. We conjecture that there is a tradeoff between the number of gates and number of wires needed to compute arbitrary symmetric functions, and that it is not possible to optimize both simultaneously.

I would like to thank Ian Parberry and Nick Pippenger for pointers to many of the results used in this paper, as well as Richard Beigel, my advisor, for encouragement, support, proofreading and for prompting this line of research.

References

[BBL90] Paul Beame, Erik Brisson, and Richard Ladner. The complexity of computing symmetric functions using threshold circuits. Technical Report 90-01-01, University of Washington, Department of Computer Science and Engineering, 1990.

[HMP+87] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 99–110, 1987.

[Lup71] O. B. Lupanov. Circuits using threshold elements. *Soviet Physics Doklady*, 17(2):91–93, August 1971.

[Lup73] O. B. Lupanov. On the synthesis of networks of threshold elements. *Problemy Kibernetiki*, 26:109–140, 1973. (in Russian).

[PS88] Ian Parberry and Georg Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, (36):278–302, 1988.

[PS90] Ramamohan Paturi and Michael E. Saks. On threshold circuits for parity. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, 1990. 397-404.

[Red70] N. P. Redkin. Synthesis of threshold circuits for certain classes of boolean functions. *Cybernetics*, pages 540–544, May 1970.

[SB91] K. Y. Siu and J. Bruck. On the power of threshold circuits with small weights. *SIAM J. Discrete Math*, 4(3):423–435, August 1991.

[SRK91] Kai-Yeung Siu, Vwani Roychowdhury, and Thomas Kailath. Depth-size tradeoffs for neural computation. *IEEE Transactions on Computers*, 40(12):1402–1412, December 1991.

[SRK92a] Kai-Yeung Siu, Vwani Roychowdhury, and Thomas Kailath. Computing with almost optimal size threshold circuits. Manuscript, 1992.

[SRK92b] Kai-Yeung Siu, Vwani Roychowdhury, and Thomas Kailath. Toward massively parallel arithmetic computation. Manuscript, 1992.

[Tho83] Thomas M. Thompson. *From Error-Correcting Codes Through Sphere Packings to Simple Groups*, volume 11 of *The Carus Mathematical Monographs*. The Mathematical Association of America, 1983.

[Win63] R. O. Winder. Bounds on threshold gate realizability. *IEEE Transactions on Computers*, C-12:561–564, October 1963.

Appendix A:

In Section 3, we presented the following construction of circuits from [SRK91] that obtain the binary representation of a sum of inputs:

Let d be a natural number and let e be the least integer such that $2^e > n^{1/d}$. We construct a family of circuits $\mathbf{B}_d$ where the circuit for inputs of length n has gates $\mathrm{g}_{j,k}$ for $1 \leq j \leq d$ and $1 \leq k \leq 2^e - 1$. Gate $\mathbf{g}_{j,k}$ computes the predicate

$$X - \sum_{j'=j+1}^{d} 2^{e(j'-1)} \left(\sum_{k'=1}^{2^e-1} \mathrm{g}_{j',k'} \right) \geq k 2^{e(j-1)}. \tag{2}$$

We now provide a proof that these circuits actually do provide the binary representation of X.

Lemma 22. *For $1 \leq m \leq d$,*

a. $\sum_{k=1}^{2^e-1} \mathrm{g}_{m,k} = (X \bmod 2^{em}) \operatorname{div} 2^{e(m-1)}$ *(i.e., the mth digit base 2^e), and*

b. $X \bmod 2^{e(m-1)} = X - \sum_{j=m}^{d} 2^{e(j-1)} (\sum_{k=1}^{2^e-1} \mathrm{g}_{j,k})$.

Proof: We will prove the assertions together by induction, starting with $m = d$.

Base case. Since $2^e > n^{1/d}$, $X \bmod 2^{ed} = X$. Observing that $\mathbf{g}_{d,k}$ computes the predicate $X \geq k 2^{e(d-1)}$, it becomes clear that

$$\sum_{k=1}^{2^e-1} \mathrm{g}_{d,k} = X \operatorname{div} 2^{e(d-1)} = (X \bmod 2^{ed}) \operatorname{div} 2^{e(d-1)}. \tag{3}$$

This proves (a) for $m = d$. To prove part (b), we note that

$$\begin{aligned} X \bmod 2^{e(d-1)} &= X - 2^{e(d-1)} \left(X \operatorname{div} 2^{e(d-1)} \right) \\ &= X - 2^{e(d-1)} \textstyle\sum_{k=1}^{2^e-1} \mathrm{g}_{d,k}, \qquad \text{by (3).} \end{aligned}$$

Induction. Assume that (a) and (b) are true for all levels greater than m. Part (a) for level m follows from part (b) for level $m+1$, since gate $\mathbf{g}_{m,k}$ computes the predicate

$$\begin{aligned} X - \sum_{j'=m+1}^{d} 2^{e(j'-1)} \left(\sum_{k'=1}^{2^e-1} \mathrm{g}_{j',k'} \right) &\geq k 2^{e(m-1)} \qquad \text{by (2), which is} \\ X \bmod 2^{em} &\geq k 2^{e(m-1)} \qquad \text{by part } (b) \end{aligned}$$

So,

$$\sum_{k=1}^{2^e-1} \mathrm{g}_{m,k} = (X \bmod 2^{em}) \operatorname{div} 2^{e(m-1)}.$$

To prove part (b), we note that for $m \leq j \leq d$,

$$
\begin{aligned}
2^{e(j-1)} \sum_{k=1}^{2^e-1} \mathbf{g}_{j,k} &= 2^{e(j-1)} \left((X \bmod 2^{ej}) \operatorname{div} 2^{e(j-1)} \right) && \text{by part } (a) \\
&= \left(X \bmod 2^{ej} \right) - \left(\left(X \bmod 2^{ej} \right) \bmod 2^{e(j-1)} \right) \\
&= \left(X \bmod 2^{ej} \right) - \left(X \bmod 2^{e(j-1)} \right)
\end{aligned}
$$

Telescoping,

$$
\sum_{j=m}^{d} 2^{e(j-1)} \left(\sum_{k=1}^{2^e-1} \mathbf{g}_{j,k} \right) = X - \left(X \bmod 2^{e(m-1)} \right)
$$

which implies that

$$
X - \sum_{j=m}^{d} 2^{e(j-1)} \left(\sum_{k=1}^{2^e-1} \mathbf{g}_{j,k} \right) = X \bmod 2^{e(m-1)}
$$

∎

Proof: (of Theorem 3) It is clear that $\mathbf{B}_d$ is a circuit of depth d with at most $2dn^{1/d}$ gates. To show that the binary representation of X is linearly computable from $\mathbf{B}_d$, we let $b_0, \ldots, b_{\lceil \log n \rceil - 1}$ be the binary representation of X. (i.e., $X = \sum_{j=0}^{\lceil \log n \rceil - 1} b_j 2^j$.)

Then $b_j = (X \bmod 2^{j+1}) \operatorname{div} 2^j$, for $0 \leq j \leq \lceil \log n \rceil - 1$. Let e be as in Lemma 22 and let $m = j \operatorname{div} e$. Then

$$
\begin{aligned}
b_j &= (X \bmod 2^{j+1}) \operatorname{div} 2^j \\
&= \left(\left((X \bmod 2^{e(m+1)}) \bmod 2^{j+1} \right) \operatorname{div} 2^{em} \right) \operatorname{div} 2^{j-em} \qquad \text{since } em \leq j < e(m+1) \\
&= \left(\left((X \bmod 2^{e(m+1)}) \operatorname{div} 2^{em} \right) \bmod 2^{j+1-em} \right) \operatorname{div} 2^{j-em} \\
&= \left(\left(\sum_{k=1}^{2^e-1} \mathbf{g}_{(m+1),k} \right) \bmod 2^{j+1-em} \right) \operatorname{div} 2^{j-em}, \qquad \text{by Lemma 22.}
\end{aligned}
$$

Since the $\mathbf{g}_{(m+1),k}$'s have the property that $\left(\sum_{k=1}^{2^e-1} \mathbf{g}_{(m+1),k} \right) \leq t$ if and only if $\mathbf{g}_{(m+1),t} = 1$, and since $\left(\left(\sum_{k=1}^{2^e-1} \mathbf{g}_{(m+1),k} \right) \bmod 2^{j+1-em} \right) \operatorname{div} 2^{j-em}$ is a symmetric function of the $\mathbf{g}_{(m+1),k}$'s, we can apply Lemma 2 to see that b_j is linearly computable from $\mathbf{B}_d$. ∎

This construction is pretty tight. Using the techniques of [PS90], [SRK92a] prove that $\Omega\left(dn^{1/d} / \log^2 n \right)$ gates are necessary for a depth $d+1$ threshold circuit to compute parity.

Appendix B:

We provide a proof of Theorem 7 [SRK91].

Proof: Let $X = \sum_{i=1}^{n} 2^{i-1}x_i$. Let $\{u_1, \ldots, u_t\} = \left\{i : \bar{f}(i) = 0 \text{ and } \bar{f}(i+1) = 1\right\}$ and let $u_i < u_{i+1}, \forall i$. Similarly, let $\{v_1, \ldots, v_r\} = \left\{i : \bar{f}(i) = 1 \text{ and } \bar{f}(i+1) = 0\right\}$ and let $v_i < v_{i+1}, \forall i$. Note that $t \leq m/2$. Assume, without loss of generality, that $\bar{f}(0) = 0$, so that $u_1 < v_1$. Let $e = \lceil \sqrt{t} \rceil$. The gates on the bottom level of our circuit will be $\mathbf{g}_j$ which, for $1 \leq j \leq e$, compute the predicates

$$X \geq u_{ej}.$$

On the second level of our circuit, for $1 \leq k \leq m/2e$, we have gates $\mathbf{a}_k$ computing the predicates

$$X \geq u_k + \sum_{j=1}^{e} (u_{ej+k} - u_{e(j-1)+k})\mathbf{g}_j$$

and gates $\mathbf{b}_k$ computing the predicates

$$X \geq v_k + \sum_{j=1}^{e} (v_{ej+k} - v_{e(j-1)+k})\mathbf{g}_j.$$

These sums telescope so that if j is such that $u_{e(j-1)+1} \leq X \leq u_{ej}$, then $\mathbf{a}_k$ computes the predicate $\sum_{i=1}^{n} 2^{i-1}x_i \geq u_{e(j-1)+k}$ and $\mathbf{b}_k$ computes $\sum_{i=1}^{n} 2^{i-1}x_i \geq v_{e(j-1)+k}$. Thus, the top gate of the circuit,

$$\sum_{j=1}^{e} \mathbf{a}_j - \sum_{j=1}^{e} \mathbf{b}_j > 0$$

will compute $\bar{f}(X)$. ∎