Threshold Circuits of Small Majority-Depth

Alexis Maciel*

Department of Computer Science, University of Maryland
College Park, MD 20742, U.S.A.

Denis Thérien†

School of Computer Science, McGill University
Montréal, Québec, H3A 2A7, Canada

August 9, 1996

*Email: alexis@cs.umd.edu. This work was done while this author was at the School of Computer Science, McGill University, and supported by NSERC and FCAR.

†Email: denis@cs.mcgill.ca. Supported by NSERC and FCAR.

Proposed running head:

Threshold Circuits of Small Majority-Depth

Send proofs to:

Alexis Maciel
Department of Computer Science
A.V. Williams Building
University of Maryland
College Park, MD 20742
U.S.A.
alexis@cs.umd.edu
(301) 405-2692

Abstract

Constant-depth polynomial-size threshold circuits are usually classified according to their total depth. For example, the best known threshold circuits for iterated multiplication and division have depth four and three, respectively. In this paper, the complexity of threshold circuits is investigated from a different point of view: explicit AND, OR gates are allowed in the circuits, and a threshold circuit is said to have majority-depth d if no path traverses more than d threshold gates. It is then shown that iterated multiplication can be computed by polynomial-size threshold circuits of total depth five but of majority-depth three. Circuits of depth four and majority-depth two are obtained for division and powering. These results rely on a careful implementation of iterated addition and Chinese remaindering. In addition, a simple symbolic calculus for composing circuit classes is developed: this notation allows for a concise and elegant presentation of the results.

List of symbols

AC^0, AND, LT_d, MAJ, MOD_m, NC^0, NOT, NP, OR, POL, SUM, SYM, TC^0, $\widehat{\mathrm{TC}}^0$	uppercase roman letters; the superscript is a "zero"
$\mathrm{dom}, \log, \max, \mathrm{mod}$	lowercase roman letters
$O(\ldots)$	uppercase italic "oh"
$\Delta_d, \Gamma, \Lambda, \Pi_d, \Sigma_d, \theta_t$	Greek letters
$\mathbf{N}, \mathbf{Z}, \mathbf{Z}_p$	uppercase roman bold; the subscript is lowercase italic

Footnotes

1. Note that the inputs to the gate can be repeated, negated, or constant. Therefore, the function computed by such a circuit is not necessarily the symmetric function labeling the gate. It may not even be symmetric.

2. Recall that we are representing Boolean values by 0 and 1, and not 1, -1.

3. The case $d = 1$ has to be handled separately. In that case, it is easy to see that Δ_1 is closed under complementation but not under union and intersection; Σ_1 is closed under union but not under complementation and intersection; and Π_1 is closed under intersection but not under complementation and union. As for AC_1^0, a simple argument shows that this class is closed under complementation but not under union and intersection.

4. Unless otherwise specified, we assume that numbers are given by their binary representation. Also, as is the case here, n will usually not be the exact length of the input, but something polynomially related to it.

5. This was not explicitly stated by Hofmeister *et al.* (1991), but a quick examination of their proof reveals that the output gates of the circuit are all OR gates.

6. In particular, Γ can be any of the complexity classes defined in this article. (See Section 2.4.)

(The footnotes also appear at the bottom of the appropriate pages.)

1 Introduction

The computing power of small-depth circuits has been extensively studied in the past fifteen years. In particular, several complexity classes defined by polynomial-size constant-depth circuits constructed with various types of unbounded fan-in symmetric gates (i.e., gates whose output value is determined by the sum of the inputs) have been considered. A primary example of such a class is AC^0, which is obtained by allowing AND and OR gates. The seminal result that parity is not computable in AC^0 was first obtained by Furst, Saxe and Sipser (1984) and Ajtai (1983): this lower bound was subsequently improved by Yao (1985) and Håstad (1986), and generalized by Razborov (1987), Barrington (1986) and Smolensky (1987). Furthermore, by combining work of Fagin *et al.* (1985) and Yao (1985), we now have a complete characterization of the symmetric functions realizable in AC^0; they are (essentially) the threshold functions

$$\theta_t(x_1, \ldots, x_n) = \begin{cases} 1 & \sum_{i=1}^n x_i \geq t \\ 0 & \text{else} \end{cases}$$

where the parameter t is polylogarithmic in n.

Another such class is TC^0 which consists of polynomial-size constant-depth circuits constructed with arbitrary threshold gates. It is well-known that the class remains the same if only majority gates are allowed. On the other hand, threshold circuits can compute any symmetric function, hence arbitrary symmetric gates can be used. By the previous remark, $\mathrm{AC}^0 \subset \mathrm{TC}^0$, but otherwise this class is poorly understood. Indeed, threshold circuits have been found to be surprisingly powerful. For example, Beame *et al.* (1986) gave algorithms to compute powering (given x, return $x^2, \ldots, x^n$), iterated multiplication (given $x_1, \ldots, x_n$, return the product $x_1 \cdots x_n$) and division (given x, y, return $\lfloor x/y \rfloor$) that are easily checked to be in TC^0, as was observed by Reif (1987), Hajnal *et al.* (1987) and Immerman and Landau (1989). In fact, it is not known if all of NP is contained in TC^0.

Usually, circuits in TC^0 are classified according to their total depth, but in our work, we will adopt a different perspective. Considering that arbitrary threshold gates are much more powerful than AND and OR gates, we propose a parametrization of TC^0 which allows free use of AC^0-subcircuits: we define $\widehat{\mathrm{TC}}^0$ to consist of constant-depth polynomial-size circuits constructed with NOT, AND, OR and threshold gates, and we declare that such a circuit is in $\widehat{\mathrm{TC}}^0_d$ provided that no path traverses more than d threshold gates. We say that circuits in $\widehat{\mathrm{TC}}^0_d$ have *majority-depth d*. Clearly $\widehat{\mathrm{TC}}^0 = \mathrm{TC}^0$ and $\widehat{\mathrm{TC}}^0_0 = \mathrm{AC}^0$.

For the example of iterated multiplication, a straightforward implementation of the algorithm of Beame *et al.* (1986) yields a circuit in $\widehat{\mathrm{TC}}^0_5$. The method of Immerman and Landau (1989) can be implemented as a $\widehat{\mathrm{TC}}^0_3$ circuit, with rather complicated AC^0 interconnections. This compares with the best known circuit, in terms of total depth, which is due to Siu and Roychowdhury (1994) and uses four levels of majority gates but no other. In contrast, we will produce a circuit in $\widehat{\mathrm{TC}}^0_3$ with only two levels of AND-OR gates. Thus we save one level of majority gates compared to the circuit of Siu and Roychowdhury (1994), at the expense of increasing the total depth by one. Similar results are obtained for powering and division.

Our paper is organized as follows: in Section 2, we summarize the necessary background and prove some results about various elementary classes of functions. In particular, we develop a simple symbolic calculus for composing bounded-depth circuit classes: this notation allows for a succinct and elegant presentation of our results.

The next section presents a slightly modified version of a known method to solve the iterated addition problem, which can be implemented with only one level of threshold gates (instead of two) at the cost of increasing the total depth by two: this algorithm will be central to the design of further algorithms. Then we give a general presentation of Chinese remaindering and how its implementation can be used to construct efficient threshold circuits for arithmetic problems.

We are then ready to describe our solutions to the problems of iterated multiplication, powering and division. The ideas presented earlier lead to circuits having one less level of threshold gates than the best known ones, at the cost of increasing the total depth by only one.

Finally, we briefly discuss the hierarchy $\widehat{\mathrm{TC}}^0_1 \subseteq \widehat{\mathrm{TC}}^0_2 \subseteq \widehat{\mathrm{TC}}^0_3 \subseteq \cdots$, its relationship to the usual depth hierarchy for TC^0, and the question of lower bounds.

2 Preliminaries

In this section, we provide basic definitions and some background on Boolean circuits. We also investigate exactly how TC^0 circuits can compute symmetric functions. In the process, we introduce some of the central ideas of this article and establish preliminary results that will be needed later.

2.1 Boolean functions, circuits and circuit complexity classes

In this article, *Boolean functions* are functions, either total or partial, from $\{0,1\}^*$ to $\{0,1\}^*$. In particular, the domain of a Boolean function may be any subset of $\{0,1\}^*$. Boolean functions will always have the property that they map inputs of the same length to outputs of the same length. This implies that every Boolean function can be written as a sequence $f_1, f_2, \ldots$, where f_k is a function from $\{0,1\}^k$ to $\{0,1\}^l$ and l is some function of k. Such a sequence is called a *family* of functions.

Examples of Boolean functions that will be considered in this article are the functions MOD_m and θ_t defined by $\mathrm{MOD}_m(x_1, \ldots, x_n) = 1$ iff m divides $\sum_{i=1}^n x_i$ and $\theta_t(x_1, \ldots, x_n) = 1$ iff $\sum_{i=1}^n x_i \geq t$. The parameters m and t are functions of n that can, of course, be constant. The θ_t are called *threshold* functions, and MOD_2 and $\theta_{n/2}$ are also called *parity* and *majority*.

A *Boolean circuit with n inputs and m outputs* naturally defines a total function from $\{0,1\}^n$ to $\{0,1\}^m$, if we assume that the output gates are numbered $1, 2, \ldots, m$. Note that input nodes can be labeled by variables, negated variables, or the constants 0 and 1. Since we will often consider partial functions, we say that f is *computed* by a circuit C if f is the total function defined by C or one of its restrictions. In other words, C computes f if C outputs the value of f for every input in the domain of f. As usual, to compute functions on inputs of arbitrary length, we consider *families* of circuits, that is, sequences $C_1, C_2, C_3, \ldots$, where C_n is a circuit with n inputs.

We will consider mainly classes of functions defined by constant-depth polynomial-size circuits with gates of unbounded fan-in, where the *size* of a circuit is defined to be the number of edges it contains.

The simplest such class is AC^0, where only NOT, AND and OR gates are allowed. AC^0 circuits cannot compute parity, MOD_m, for any constant m, and majority (Ajtai 1983, Furst *et al.* 1984, Håstad 1986, Yao 1985). Moreover, $\theta_t \in \mathrm{AC}^0$ if and only if $t \in (\log n)^{O(1)}$ (see Ajtai and Ben-Or 1984, Denenberg *et al.* 1986 or Fagin *et al.* 1985 for the upper bound; for the lower bound, combine a result of Fagin *et al.* 1985 with Håstad 1986 or Yao 1985).

By *adding* majority gates to AC^0 circuits, we get the class TC^0. It is not hard to show that TC^0 contains all the MOD_m functions, for m constant (Furst *et al.* 1984, Hajnal *et al.* 1987). In fact, TC^0 contains all the symmetric functions (Hajnal *et al.* 1987), i.e., functions whose value is determined by the sum of the inputs.

Both these classes are contained in the well-known circuit complexity class NC^1. In general, NC^i is the class of functions computed by circuits of polynomial size and depth $O((\log n)^i)$ with NOT, AND and OR gates of fan-in bounded by a constant. It is an important open problem to determine if the following inclusion is strict: $\mathrm{TC}^0 \subseteq \mathrm{NC}^1$.

NC^0 is the simplest class of constant-depth circuits. Circuits in this class are limited by the fact that each one of their outputs can only depend on a constant number of inputs. Nevertheless, NC^0 circuits will occur frequently in the constructions of the following sections.

For all of these constant-depth circuit classes, we define depth hierarchies by considering circuits of fixed constant-depth. Thus, NC^0_d, AC^0_d and TC^0_d are the classes of functions computed, respectively, by NC^0, AC^0 and TC^0 circuits of depth exactly d. In addition, Σ_d and Π_d are the classes of functions computed by AC^0_d circuits whose output gates consist entirely of OR and AND gates, respectively.

We will sometimes use the more descriptive notation AND, OR and MAJ for the classes Σ_1, Π_1 and TC^0_1. Also, AND_k and OR_k will denote AND and OR, but with gates of fan-in bounded by k. Let $\mathrm{AND}_{O(f)}$ and $\mathrm{OR}_{O(f)}$ denote $\bigcup_{c \geq 1} \mathrm{AND}_{cf}$ and $\bigcup_{c \geq 1} \mathrm{OR}_{cf}$.

Of course, $\mathrm{AC}^0 = \bigcup_{d \geq 1} \mathrm{AC}^0_d$ and similarly for the other classes. If f has output length one, then $f \in \mathrm{AC}^0_d$ if and only if $f \in \Sigma_d \cup \Pi_d$.

The NC^0 depth hierarchy is very simple. Each output gate of an NC^0 circuit computes a function on a constant number of inputs; we can express this function in disjunctive normal form and compute it in depth two and constant size. Therefore, $\mathrm{NC}^0 = \mathrm{NC}^0_2$. Also, $\mathrm{NC}^0_1 \subset \mathrm{NC}^0_2$ since $f(x_1, \ldots, x_n) = \mathrm{MOD}_2(x_1, x_2) \in \mathrm{NC}^0_2 - \mathrm{NC}^0_1$.

The AC^0 depth hierarchy, on the other hand, has been shown to be infinite, i.e., $\mathrm{AC}^0_d \subset \mathrm{AC}^0_{d+1}$ for all d (Sipser 1983, Håstad 1986). In contrast, all that is known about the TC^0 depth hierarchy is that $\mathrm{TC}^0_1 \subset \mathrm{TC}^0_2 \subset \mathrm{TC}^0_3$ (Hajnal *et al.* 1987). Note that since there are functions complete for NC^1 under sufficiently fine notions of reducibility, proving that the TC^0 depth hierarchy is infinite would imply that $\mathrm{TC}^0 \neq \mathrm{NC}^1$.

2.2 Threshold gates

A *threshold gate* is simply a gate labeled with a threshold function, and a threshold circuit, a circuit consisting entirely of threshold gates.

Since AND, OR and majority are threshold functions (θ_n, θ_1, $\theta_{n/2}$), it is clear that all of TC^0 can be computed by constant-depth polynomial-size threshold circuits. On the other hand, it is also true that single majority gates can simulate any threshold gate: simply add the appropriate number of constant inputs. This implies that TC^0_d can be equivalently defined defined in terms of depth-d polynomial-size threshold circuits.

We can generalize threshold functions by allowing integer weights $w_1, \ldots, w_n$ to be put on the terms of the sum and outputting 1 iff $\sum_{i=1}^n w_i x_i \geq t$. The weights become parameters in terms of which the function is defined, just like the threshold t. We call these functions *weighted threshold functions*.

A weighted threshold gate is a gate labeled with a weighted threshold function, and a weighted threshold circuit is a circuit consisting entirely of weighted threshold gates. The *weight* of a circuit is the maximum absolute value of any weight occurring in its gates. A family of weighted threshold circuits has *small weight* if its weight is bounded by some polynomial in n.

Small-weight threshold circuits provide another equivalent definition of TC^0. First, consider a single weighted threshold gate. By feeding several copies of the same input, we can reduce the weights to 1 or -1. Negative weights are then simulated by using negations. Second, given a depth-d small-weight polynomial-size threshold circuit, starting at level d, simulate threshold gates with majority gates and combine the NOT gates obtained with the threshold gates from previous levels. It can be verified that the resulting circuit has depth d and polynomial size.

To summarize, we have the following well-known result:

Proposition 2.1 *The class* TC^0_d *can be equivalently defined as depth-d polynomial-size* AND-OR-*majority circuits, majority circuits, threshold circuits, and small-weight threshold circuits.*

In particular, TC^0_d is closed under complementation.

General weighted threshold circuits, or large-weight threshold circuits, will be mentioned in Section 4.

2.3 Symmetric gates

A symmetric gate is a gate labeled with a symmetric function. This means that the output of a symmetric gate is determined by the sum of its inputs. This generalizes both threshold and MOD_m gates by allowing the sum of the inputs to be tested for any condition and not just "$\geq t$" or "$\equiv 0 \pmod{m}$".

Definition 2.2 *Let* SYM *denote the class of functions computed by depth-one polynomial-size circuits consisting of symmetric gates.*

We have already mentioned that TC^0 circuits can compute any symmetric function. In fact, depth-two TC^0 circuits are sufficient.

Theorem 2.3 (Hajnal et al. 1987) $\mathrm{SYM} \subseteq \mathrm{TC}^0_2$.

Proof Suppose that $f(x_1, \ldots, x_n) = h(\sum_{i=1}^n x_i)$, where $h : \mathbf{Z} \to \{0,1\}$, and suppose that $h^{-1}(1) \cap \{0, \ldots, n\} = \{v_1, \ldots, v_k\}$ with $0 \leq v_1 < \cdots < v_k \leq n$. For $j = 1, \ldots, k$, let

$$L_j(x) = 1 \text{ iff } \sum_{i=1}^n x_i \leq v_j$$

and

$$G_j(x) = 1 \text{ iff } \sum_{i=1}^n x_i \geq v_j,$$

where $x = x_1 \cdots x_n$. Then, $\sum_{j=1}^k (L_j(x) + G_j(x)) = k + 1$ if $f(x) = 1$, and $\sum_{j=1}^k (L_j(x) + G_j(x)) = k$ if $f(x) = 0$. Therefore, a threshold-$(k+1)$ gate with inputs $L_1, \ldots, L_k, G_1, \ldots, G_k$ will output $f(x)$. This shows that any symmetric function with n variables can be computed by a depth-two threshold circuit of size $2n^2$. The result follows. □

As before, we can generalize symmetric gates by assigning integer weights to the inputs. It is again possible to show that these generalized gates can be simulated by a single symmetric gate.

Proposition 2.4 *Let* $f(x) = h(\sum_{i=1}^m w_i x_i)$ *where* $w_1, \ldots, w_m \in \mathbf{Z}$ *and* $h : \mathbf{Z} \to \{0,1\}$. *Let* $w = \max |w_i|$. *Then* f *can be computed by a circuit consisting of a single symmetric gate of fan-in* mw.[1] *Moreover, if the weights are all nonnegative or all not positive, then no negations are needed in the circuit.*

In particular, SYM contains all functions of a weighted sum of the inputs, provided the weights are small, i.e., bounded by a polynomial in n.

We will often use symmetric gates as an intermediate step in constructing threshold circuits. In doing this, we will frequently make use of the closure properties of SYM. This will allow symmetric gates to be combined with certain types of gates into which they feed.

The class SYM is obviously closed under complementation since the negation of a symmetric gate can be clearly computed by a symmetric gate. However, SYM is also closed under finite intersections and unions. In fact, SYM is closed under finite Boolean functions. The following theorem slightly generalizes results of Hofmeister *et al.* (1991) and Beigel (1994).

[1] Note that the inputs to the gate can be repeated, negated, or constant. Therefore, the function computed by such a circuit is not necessarily the symmetric function labeling the gate. It may not even be symmetric.

Theorem 2.5 *Let $f(x) = h(S_1, \ldots, S_c)$ where $S_j = \sum_{i=1}^m w_{ji}x_i$, $w_{ji} \in \mathbf{Z}$, $\max|w_{ji}| = w$ and $h : \mathbf{Z}^c \to \{0,1\}$. Then f can be expressed as a function of a single weighted sum with integer weights bounded in absolute value by $(4mw)^c$.*

Proof We use a technique that Papadimitriou and Zachos (1983) used in the context of counting Turing machines. Let B be the smallest power of 2 greater than wm. Then $B > |S_j|$ for all j.

Let $S = \sum_{j=1}^c S_j(2B)^{j-1}$ and, for $v_1, \ldots, v_c \in \{-B+1, \ldots, B-1\}$, define

$$g\left(\sum_{j=1}^{c} v_j(2B)^{j-1}\right) = h(v_1, \ldots, v_c).$$

It is not hard to see that g is well-defined and that $g(S) = f(x)$. The weights of S are bounded in absolute value by $\sum_{j=1}^c w(2B)^{j-1} \leq 2w(2B)^{c-1} \leq (4mw)^c$. $\square$

Corollary 2.6 *Any function of c symmetric gates of fan-in m can be computed by a single symmetric gate of fan-in $(2m)^c$.*

In particular, if c is constant, then the fan-in of the resulting gate is polynomial in m.

2.4 Combining classes of functions

Circuits are often designed in two or more stages, each using a different type of gates. In order to refer to the class of functions computed by such circuits in a concise way, we will use extensively the following notation:

Definition 2.7 *If Γ_1 and Γ_2 are classes of Boolean functions, then let $\Gamma_1 \circ \Gamma_2$ denote the class of functions f of the form $f(x) = f_1(f_2(x))$, where $f_1 \in \Gamma_1$ and $f_2 \in \Gamma_2$.*

This definition is made exclusively in terms of the composition of functions; in particular, no reference is made to Boolean circuits. Nevertheless, if Γ_1 and Γ_2 are circuit complexity classes, then $\Gamma_1 \circ \Gamma_2$ is the class of functions that can be computed by circuits consisting of two stages: the first one computes a function in Γ_2, the second one, a function in Γ_1. More precisely, if $g \in \Gamma_1$ is computed by a family of circuits $D_1, D_2, \ldots$ and $h \in \Gamma_2$ is computed by $E_1, E_2, \ldots$, then $f(x) = g(h(x))$ is computed by $C_1, C_2, \ldots$ where C_n is $D_m(E_n)$, m being the output length of E_n.

For example, if $\Gamma_1 = \mathrm{AC}_2^0$ and $\Gamma_2 = \mathrm{TC}_3^0$, then every function in $\Gamma_1 \circ \Gamma_2$ can be computed by a family of circuits of depth six with majority gates on levels one, two and three, NOT gates on level four, and AND-OR gates on levels five and six. (The level of NOT gates is needed since the inputs of the AC_2^0 circuit may be negated.) It is easy to see that the total size of the circuit is polynomial in n. Moreover, since TC_3^0 is closed under complementation, NOT gates on level four can be combined with the first stage to give a polynomial-size depth-five circuit whose first three levels consist of majority gates and whose last two consist of AND-OR gates.

Conversely, if f is computed by a family of circuits in two stages, then f can be put in a certain class $\Gamma_1 \circ \Gamma_2$. However, some care must be taken. Consider the sequence of functions $g_1, g_2, \ldots$ where g_n is the function computed by the second stage of the nth circuit. If m, the output length of the first stage, is not an injective function of n, it might not be possible to define g such that all the g_n are restrictions of g to inputs of a certain length.

The obvious solution to this technical problem is to "pad" the output of the first stage so that m is monotone increasing. Then g can be defined and $f(x) = g(h(x))$, where h is not the function computed by the first stage of the family of circuits, but its padded version.

Another reason to pad the output of the first stage is so that the complexity of the second stage be measured relative to n and not relative to m, which might be much smaller.

To illustrate all this, suppose that f is computed by a family of depth-five polynomial-size circuits with a first stage consisting of three levels of majority gates and a second stage consisting of two levels of AND-OR gates. Notice that even if the second stage of the family of circuits does define correctly a function g, this function might be the parity function, if m is equal to $\log n$, for example. Since parity is not in AC^0, we would not be able to conclude that f is in $\mathrm{AC}^0_2 \circ \mathrm{TC}^0_3$. However, pad the output of the first stage so that m is increasing. Then $f(x) = g(h(x))$, where $g \in \mathrm{AC}^0_2$ since the input length of the first stage is now at least n, and $h \in \mathrm{TC}^0_3$ since such padding preserves the polynomial size of the first stage. Therefore, $f \in \mathrm{AC}^0_2 \circ \mathrm{TC}^0_3$.

In general, whenever the circuit class corresponding to the first stage is *closed under polynomial-size padding*, then we can assume that m is monotone increasing. All the classes considered in this article have this property and this "padding trick" will be used often.

2.5 Polynomials of constant degree

Polynomials are a common and useful way of representing Boolean functions. For example, many results on constant-depth circuits have been obtained by considering various kinds of polynomial representations of Boolean functions. (For more on this, see Beigel 1993.)

Simple examples of polynomial representations are $1-x_1 = \mathrm{NOT}(x_1)$, $x_1 \cdots x_n = \mathrm{AND}(x_1, \ldots, x_n)$ and $1-(1-x_1)\cdots(1-x_n) = \mathrm{OR}(x_1, \ldots, x_n)$.[2] We will be particularly interested in polynomials with small integer coefficients and constant degree.

Definition 2.8 *Let* POL *denote the class of Boolean functions whose output length is bounded by a polynomial in n and such that each output can be written as a polynomial in the input variables with integer coefficients bounded by a polynomial in n and degree bounded by a constant. Let* SUM *be the subclass of* POL *corresponding to polynomials of degree one.*

[2] Recall that we are representing Boolean values by 0 and 1, and not 1, -1.

For example, θ_t restricted to inputs with $t-1$ or t 1's can be written as $x_1 + \cdots + x_n - (t-1)$. It is easy to show that POL = SUM $\circ$ $\mathrm{AND}_{O(1)}$ = SUM $\circ$ $\mathrm{OR}_{O(1)}$ and that SUM $\subseteq$ MAJ and SUM $\subseteq$ MOD_m for any constant m.

Proposition 2.9 SYM $\subseteq$ SUM $\circ$ TC^0_1.

Proof Consider the proof of Theorem 2.3. Define H from $\{0,1\}^n$ to $\{0,1\}^{2k}$ by $H(x) = L_1(x)G_1(x)\cdots L_k(x)G_k(x)$. Then $f(x) = \sum_{i=1}^{2k} H_i(x) - k$, where $H_i(x)$ is the ith bit of $H(x)$. It is not hard to see that this implies the result. $\square$

This simple observation will be used frequently in this article since it will allow circuits with symmetric gates to be simulated efficiently by making use of the following result:

Proposition 2.10 SYM $\circ$ SUM $=$ SYM. *In addition,* MAJ $\circ$ SUM $=$ MAJ *and* $\mathrm{MOD}_m \circ \mathrm{SUM} = \mathrm{MOD}_m$.

The idea of the proof is to simply feed the inputs of the sums directly into the symmetric gate.

Corollary 2.11 SYM $\circ$ SYM $=$ SYM $\circ$ MAJ *and* MAJ $\circ$ SYM $=$ TC^0_2.

The classes POL and SUM are clearly closed under complementation since $1-f$ is the negation of f. However, POL is also closed under finite intersection and union since $\mathrm{AND}(f_1,\ldots,f_c) = f_1\cdots f_c$, $\mathrm{OR}(f_1,\ldots,f_c) = 1-(1-f_1)\cdots(1-f_c)$, and the product of c polynomials of degree d has degree cd. In addition, POL $\circ$ POL $=$ POL.

2.6 Using AND-OR gates

The approach taken in this article when investigating the complexity of functions in terms of threshold circuits is to minimize first the depth as measured by threshold gates only, and then the total depth. As a consequence, we will be interested in showing that the function computed by certain stages in our circuits belong to AC^0_d, for d as small as possible. However, we will often show that such functions can not only be computed by AC^0_d circuits, but by both AC^0_d circuits with only OR gates at the output, and AC^0_d circuits with only AND gates at the output.

Definition 2.12 *Let* $\Delta_d = \Sigma_d \cap \Pi_d$, *i.e., the class of functions computed by both* Σ_d *and* Π_d *circuits.*

The advantage of showing that $f \in \Delta_d$ instead of just $f \in \Sigma_d$, Π_d or AC^0_d comes from the fact that Δ_d has properties that Σ_d, Π_d and AC^0_d do not have. For example, let $d \geq 2$. It is easy to verify that Δ_d is closed under complementation, union and intersection, and that $\mathrm{NC}^0 \circ \Delta_d = \Delta_d$. However, Σ_d and Π_d are closed under union and intersection but not under complementation, since this would imply that $\Sigma_d = \Pi_d$ and that $\mathrm{AC}^0_{d+1} = \mathrm{AC}^0_d$. On the other hand, AC^0_d is closed under complementation but not under union and intersection, for the same reason.[3]

Another set of properties of Δ_d is given by the following proposition.

Proposition 2.13 *For every $k, d \geq 1$, we have that $\Pi_k \circ \Delta_d = \Pi_{k+d-1}$, $\Sigma_k \circ \Delta_d = \Sigma_{k+d-1}$, $\Delta_k \circ \Delta_d = \Delta_{k+d-1}$, and $\mathrm{AC}^0_k \circ \Delta_d = \mathrm{AC}^0_{k+d-1}$.*

Proof Suppose that k is even. Then, any Π_k circuit can be transformed so that level one contains only OR gates and so that all inputs feed into these level one OR gates. Whenever an output of a Δ_d function is a positive input of a Π_k circuit (i.e., x_i), compute it with a Σ_d circuit; whenever it is a negative input (i.e., $\overline{x}_i$), compute it with a Π_d circuit. By using $\mathrm{NOT}(\mathrm{AND}(x_1, \ldots, x_n)) = \mathrm{OR}(\overline{x}_1, \ldots, \overline{x}_n)$ and by combining together OR gates, we get a depth-$(k + d - 1)$ circuit. It can also be verified that the resulting circuit is of polynomial size. We therefore have a Π_{k+d-1} circuit.

The case k is odd is handled similarly so that $\Pi_k \circ \Delta_d \subseteq \Pi_{k+d-1}$. The reverse inclusion holds since $\Pi_{k+d-1} \subseteq \Pi_k \circ \mathrm{AC}^0_{d-1} \subseteq \Pi_k \circ \Delta_d$. The second and fourth equalities are proved similarly and the third follows from the first two. $\square$

Recall that Σ_d and Π_d contain all functions, both total and partial, that can be computed with Σ_d and Π_d circuits, respectively. Therefore, Δ_d may contain partial functions that cannot be extended to total functions in Δ_d. This is simply because the Σ_d and Π_d circuits computing a function in Δ_d only have to output the same value on the domain of the function. For example, it is easy to show that the only total functions in Δ_1 are families $f_1, f_2, \ldots$ such that f_n is either 0, 1, x_i or $\overline{x}_i$. However, Δ_1 contains useful partial functions that are not just restrictions of these simple functions. An example of such a function naturally occurs in the simulation of symmetric gates, as we will now see.

First, notice that $f \in (\Sigma_d \circ \Gamma) \cap (\Pi_d \circ \Gamma)$ does not trivially imply that $f \in \Delta_d \circ \Gamma$. This is because the Σ_d and the Π_d circuits do not necessarily compute the same function. However, say that a class of functions Γ is *closed under concatenation* if $f \in \Gamma$ and $g \in \Gamma$ implies that $fg \in \Gamma$. For example, all the complexity classes defined in this article are closed under concatenation. Then,

Proposition 2.14 *If Γ is closed under concatenation, then*

$$(\Sigma_d \circ \Gamma) \cap (\Pi_d \circ \Gamma) = \Delta_d \circ \Gamma.$$

Proof Suppose that $f = G_1 \circ H_1 = G_2 \circ H_2$ where $G_1 \in \Sigma_d$, $G_2 \in \Pi_d$ and $H_1, H_2 \in \Gamma$. Let $D = \{H_1(x)H_2(x) : x \in \{0,1\}^*\}$. For every x, given $H_1(x)H_2(x)$, $f(x)$ can be computed with either a Σ_d or a Π_d circuit. It is easy to define $G_3 \in \Delta_d$ with domain D such that $f = G_3 \circ H_3$, where $H_3 = H_1H_2 \in \Gamma$. $\square$

[3]The case $d = 1$ has to be handled separately. In that case, it is easy to see that Δ_1 is closed under complementation but not under union and intersection; Σ_1 is closed under union but not under complementation and intersection; and Π_1 is closed under intersection but not under complementation and union. As for AC^0_1, a simple argument shows that this class is closed under complementation but not under union and intersection.

Now consider the simulation of symmetric gates by threshold circuits. We saw earlier that symmetric functions can be computed by depth-two threshold circuits with threshold gates on both levels. However, only one level of threshold gates is necessary. In fact,

Proposition 2.15 SYM $\subseteq \Delta_1 \circ \mathrm{NC}_1^0 \circ \mathrm{MAJ}$.

Proof In the proof of Theorem 2.3, notice that

$$\begin{aligned} f(x) = h\left(\sum_{i=1}^{n} x_i\right) = 1 \quad &\text{iff} \quad \sum_{i=1}^{n} x_i = v_j, \text{ for some } j \\ &\text{iff} \quad L_j(x) = 1 \text{ and } G_j(x) = 1, \text{ for some } j. \end{aligned}$$

Therefore, $\mathrm{OR}_{j=1}^{k} \mathrm{AND}(L_j(x), G_j(x))$ computes $f(x)$. This implies that SYM $\subseteq \Sigma_1 \circ \mathrm{NC}_1^0 \circ \mathrm{MAJ}$.

Now consider the negations of the symmetric gates. These can also be computed by symmetric gates and thus with circuits similar to the above. By using the fact that $\mathrm{NOT}(\mathrm{OR}(x_1, \ldots, x_n)) = \mathrm{AND}(\overline{x}_1, \ldots, \overline{x}_n)$, we get a $\Pi_1 \circ \mathrm{NC}_1^0 \circ \mathrm{MAJ}$ circuit for the original symmetric gates. Therefore, SYM $\subseteq \Delta_1 \circ \mathrm{NC}_1^0 \circ \mathrm{MAJ}$. □

In certain situations, we may prefer that the NC_1^0 level be at the input since, by using closure properties, it may combine with the circuit computing the input. We can show that SYM $\subseteq \Delta_1 \circ \mathrm{MAJ} \circ \mathrm{NC}_1^0$, by using the following result:

Proposition 2.16 *Suppose $f(x) = 1$ iff $t_1 \leq \sum_{i=1}^{n} x_i \leq t_2$. Then f can be computed by a depth-two circuit with a majority gate of fan-in $6n^2$ at the output and* OR *gates of fan-in two at level one.*

Proof The result is trivial if $t_1 > n$ or $t_2 < 0$. In addition, without loss of generality, we can assume that $t_1 \geq 0$ and that $t_2 \leq n$. Now observe that

$$\begin{aligned} t_1 \leq \sum_{i=1}^{n} x_i \leq t_2 \quad &\text{iff} \quad \left(\sum_{i=1}^{n} x_i - t_1\right)\left(\sum_{i=1}^{n} x_i - t_2\right) \leq 0 \\ &\text{iff} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j + \sum_{i=1}^{n} -(t_1 + t_2)x_i + t_1 t_2 \leq 0 \\ &\text{iff} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{x_i x_j} + \sum_{i=1}^{n} (t_1 + t_2)x_i \geq n^2 + t_1 t_2. \end{aligned}$$

Therefore, $t_1 \leq \sum_{i=1}^{n} x_i \leq t_2$ can be determined by a weighted threshold gate whose inputs are the x_i and the $\overline{x_i x_j}$. As indicated in Section 2.2, such a threshold gate can be simulated by a majority gate. The result follows since $\overline{x_i x_j} = \mathrm{OR}(\overline{x}_i, \overline{x}_j)$. □

By applying this result to the circuits in the proof of Proposition 2.15 we get that

Proposition 2.17 SYM $\subseteq \Delta_1 \circ \mathrm{MAJ} \circ \mathrm{NC}_1^0$.

We end this section with an example that illustrates how the ideas of this section apply to NC^0 functions.

Example 2.18 In Section 2.1, we noted that the NC^0 depth hierarchy collapses to level two, i.e., $\mathrm{NC}^0 = \mathrm{NC}^0_2$. The argument was in terms of the disjunctive normal form and yielded an $\mathrm{OR} \circ \mathrm{AND}_{O(1)}$ circuit. In fact, such a circuit can be constructed so that, on any given input, at most one AND on the first level can output 1. This implies that $\mathrm{NC}^0 \subseteq \mathrm{POL}$. In addition, we can get an $\mathrm{AND} \circ \mathrm{OR}_{O(1)}$ circuit by using conjunctive normal form or, equivalently, via the negation of the function, as was done for symmetric gates in the proof of Proposition 2.15. Therefore,

$$\mathrm{NC}^0 \subseteq \mathrm{POL} \cap (\Delta_1 \circ \mathrm{NC}^0_1).$$

Finally, notice that the above inclusion is strict: consider, for example, the function f defined by $f(x_1 \cdots x_{3n}) = \mathrm{OR}(x_1 \cdots x_n)$ on domain $D = \{x \in \{0,1\}^{3n} : n \geq 1 \text{ and } \mathrm{OR}(x_1 \cdots x_n) = \mathrm{AND}(x_{n+1} \cdots x_{2n}) = \sum_{i=2n+1}^{3n} x_i - \lceil n/2 \rceil\}$. In contrast, it is not very difficult to show that, for every total function f, $f \in \mathrm{NC}^0$ iff $f \in \Delta_1 \circ \mathrm{NC}^0_1$. $\square$

3 Iterated Addition and Multiplication

In this section, we construct constant-depth polynomial-size threshold circuits for addition, iterated addition and multiplication. These circuits are all constructed with the objective of trying to minimize first the number of levels of threshold gates and then the total depth. We provide comparisons between our results and other results that try to minimize the total depth only. The iterated addition circuit will be an important building block of our iterated multiplication and division circuits of Sections 5.1 and 5.3.

3.1 Addition

A precise definition of the problem is

addition

input	Two n-bit numbers[4]
output	Their $(n+1)$-bit sum

It is well-known that addition can be computed by AC^0 circuits, and a straightforward implementation of the usual carry-look-ahead method gives an $\mathrm{NC}^0_1 \circ \mathrm{AC}^0_2 \circ \mathrm{NC}^0_1$ circuit. By being more careful, this can be improved to $\Sigma_2 \circ \mathrm{NC}^0_1$ (see Hofmeister *et al.* 1991 and Siu *et al.* 1991, for example). We show here that addition can be computed by both $\Sigma_2 \circ \mathrm{NC}^0_1$ and $\Pi_2 \circ \mathrm{NC}^0_1$ circuits.

[4] Unless otherwise specified, we assume that numbers are given by their binary representation. Also, as is the case here, n will usually not be the exact length of the input, but something polynomially related to it.

Theorem 3.1 *Addition is in* $\Delta_2 \circ \mathrm{NC}^0_1$.

Proof Suppose that x and y are the two input numbers, given in binary by $x_n \cdots x_1$ and $y_n \cdots y_1$, so that $x = \sum_{i=1}^n x_i 2^{i-1}$ and $y = \sum_{i=1}^n y_i 2^{i-1}$. Let z be their sum, with binary representation $z_{n+1} \cdots z_1$. Let C_i be the carry coming into position i from the right. Then $z_i = \mathrm{MOD}_2(C_i, x_i, y_i)$.

Consider position i in x and y. Let $G_i = \mathrm{AND}(x_i, y_i)$, $P_i = \mathrm{MOD}_2(x_i, y_i)$ and $A_i = \mathrm{AND}(\overline{x}_i, \overline{y}_i)$. For $i > j$, let $R_{ij} = \mathrm{AND}(P_{i-1}, \ldots, P_{j+1}, G_j)$. Then, $R_{ij} = 1$ means that position i receives a carry from position j. It is easy to see that $C_i = \mathrm{OR}(R_{i(i-1)}, \ldots, R_{i1})$. Now let $Q_{ij} = \mathrm{AND}(P_{i-1}, \ldots, P_{j+1}, A_j)$, for $i > j$. $Q_{ij} = 1$ means that position j prevents position i from receiving a carry. Therefore, $\overline{C}_i = \mathrm{OR}(Q_{i(i-1)}, \ldots, Q_{i1}, Q_{i0})$, where $A_0 = 1$.

This implies that C_i is in $\Delta_2 \circ \mathrm{NC}^0$. Therefore, z_i is in $\mathrm{NC}^0 \circ \Delta_2 \circ \mathrm{NC}^0$ which equals $\Delta_2 \circ \mathrm{NC}^0$, by the closure properties of Δ_2 (see Section 2.6). Finally, $\mathrm{NC}^0 \subseteq \Delta_1 \circ \mathrm{NC}^0_1$ implies that $\Delta_2 \circ \mathrm{NC}^0 \subseteq \Delta_2 \circ \mathrm{NC}^0_1$. $\square$

Notice that the improvement from depth four ($\mathrm{NC}^0_1 \circ \mathrm{AC}^0_2 \circ \mathrm{NC}^0_1$) to depth three ($\Delta_2 \circ \mathrm{NC}^0_1$) was made possible by observing that C_i can be computed in $\Delta_2 \circ \mathrm{NC}^0_1$ instead of just $\Sigma_2 \circ \mathrm{NC}^0_1$. This fact is already implicit in Hofmeister *et al.* (1991) and Siu *et al.* (1991).

Even though, for the purposes of this article, we are mainly interested in circuits of the type $\Delta_2 \circ \mathrm{NC}^0_1$, it is interesting to note that our circuits also show that addition can be computed using constant-degree polynomials.

Theorem 3.2 *Addition is in* $\mathrm{SUM} \circ \mathrm{AC}^0_1 \circ \mathrm{NC}^0_1$.

Proof Consider the circuit in the proof of the previous theorem. On any given input, at most one R_{ij} can be 1. Therefore, the OR that gives the value of C_i can be computed in SUM. The result follows by the closure properties of POL. $\square$

Notice that addition $\in \mathrm{SUM} \circ \mathrm{AC}^0_1 \circ \mathrm{NC}^0_1$ is not a weaker statement than addition $\in \mathrm{SUM} \circ \mathrm{AND} \circ \mathrm{NC}^0_1$, since $\mathrm{SUM} \circ \mathrm{AC}^0_1 = \mathrm{SUM} \circ \mathrm{AND} = \mathrm{SUM} \circ \mathrm{OR}$. These classes are also equal to $\mathrm{POL} \circ \mathrm{AC}^0_1$.

We therefore have two depth-three addition circuits: in the first case, an AC^0_3 circuit, in the second case, a TC^0_3 circuit with only one level of threshold gates which are used to implement the sums.

In terms of total depth, as threshold circuits, these circuits are not optimal; an optimal TC^0_2 circuit was obtained by Siu and Bruck (1991) (see also Alon and Bruck 1994). However, this circuit uses threshold gates on both levels; the $\Delta_2 \circ \mathrm{NC}^0_1$ circuit of Theorem 3.1 uses no threshold gates, at the extra cost of one level of gates of constant fan-in.

3.2 Iterated addition

iterated addition

input n n-bit numbers

output Their $(n + \lceil \log n \rceil)$-bit sum

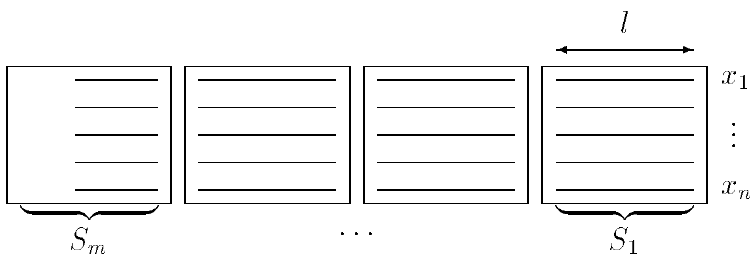
Chandra, Stockmeyer and Vishkin (1984) were the first to show that iterated addition was in TC^0, and a simple analysis of their result shows that iterated addition is in TC^0_{10}. Hofmeister, Hohberg and Köhling (1991) and Siu and Bruck (1991), independently, and using very different techniques, showed that iterated addition was in fact in TC^0_3. Siu and Roychowdhury (1994) were the first to obtain an optimal TC^0_2 circuit, by using results of Goldmann *et al.* (1992) on large-weight threshold gates (see also Goldmann and Karpinski 1993).

The depth-two circuit of Siu and Roychowdhury (1994) uses two levels of threshold gates, as does the depth-three circuit of Hofmeister *et al.* (1991).[5] On the other hand, the depth-three circuit of Siu and Bruck (1991) uses three levels of threshold gates. However, the circuit of Chandra *et al.* (1984) uses only one level of threshold gates, at the input. By using the results of Section 2.6 and Theorem 3.1, it is possible to implement their technique in $\Delta_4 \circ \mathrm{SYM}$. We show here that iterated addition can in fact be computed in $\Delta_2 \circ \mathrm{SYM}$.

Theorem 3.3 *Iterated addition is in* $\Delta_2 \circ \mathrm{SYM}$.

Proof The main idea, as in Hofmeister *et al.* (1991) and Siu and Bruck (1991), is to reduce the addition of n numbers to the addition of two numbers. Suppose that $x_1, \ldots, x_n$ are the input numbers, given in binary by $x_{in} \cdots x_{i1}$, $i = 1, \ldots, n$. Let z be their sum, with binary representation $z_{n+\lceil \log n \rceil} \cdots z_1$. Let $l = \lceil \log n \rceil$ and $m = \lceil n/l \rceil$.

Divide each x_i in m blocks of l bits and let S_k be the sum of the kth block of every x_i:

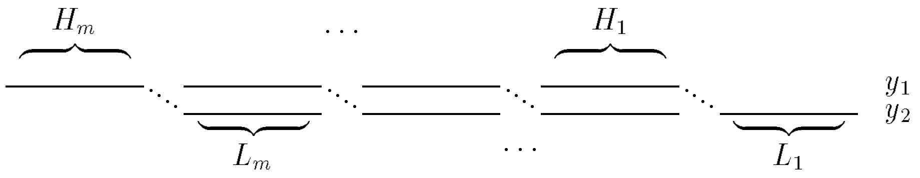


In other words, $S_k = \sum_{i=1}^{n} \sum_{j=1}^{l} x_{i((k-1)l+j)} 2^{j-1}$. It is clear that $z = \sum_{k=1}^{m} S_k 2^{(k-1)l}$.

The maximum value of any S_k is $n(2^l - 1) < 2^{l+\log n}$. Therefore, the binary representation of S_k has no more than $2l$ bits. Let L_k be the low-order half of S_k

[5] This was not explicitly stated by Hofmeister *et al.* (1991), but a quick examination of their proof reveals that the output gates of the circuit are all OR gates.

and H_k, its high-order half, i.e., $S_k = H_k 2^l + L_k$. Let y_1 and y_2 be the numbers obtained by concatenating the H_k and the L_k, respectively.



In other words, $y_1 = \sum_{k=1}^{m} H_k 2^{kl}$ and $y_2 = \sum_{k=1}^{m} L_k 2^{(k-1)l}$. Of course, $z = y_1 + y_2$.

Every bit of y_1 and y_2 can be expressed as some function of one of the S_k. Since every S_k is a sum of x_{ij} with weights bounded by $2^{l-1} \leq n$, every bit of y_1 and y_2 can be computed with a symmetric gate, by Proposition 2.4. Therefore, by Theorem 3.1, iterated addition is in $\Delta_2 \circ \mathrm{NC}_1^0 \circ \mathrm{SYM} \subseteq \Delta_2 \circ \mathrm{SYM}$. $\square$

The results of Sections 2.5 and 2.6 on the simulation of symmetric gates imply the following corollary.

Corollary 3.4 *Iterated addition is in all of the following classes:*

1. $\Delta_1 \circ \mathrm{TC}_2^0$
2. $\Delta_2 \circ \mathrm{NC}_1^0 \circ \mathrm{TC}_1^0$
3. $\Delta_2 \circ \mathrm{TC}_1^0 \circ \mathrm{NC}_1^0$

The $\Delta_1 \circ \mathrm{TC}_2^0$ circuit is a slight improvement over the $\Sigma_1 \circ \mathrm{TC}_2^0$ circuit of Hofmeister *et al.* (1991). Moreover, this circuit is not far from the optimal depth-two circuit, in the sense that the AC_1^0 function computed at the output is of a very restricted type. The second and third circuits have depth four, but have the advantage of using only one level threshold gates. Among circuits using only one level of threshold gates, these are the best that are known. It is the third circuit that will be the most useful in the following sections, since we will be able to combine the NC_1^0 level of gates with gates at the preceding levels.

Once again, it is interesting to note that iterated addition can be computed using constant-degree polynomials. This follows simply from using the $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{NC}_1^0$ addition circuit of Theorem 3.2.

Theorem 3.5 *Iterated addition is in* $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM}$.

Corollary 3.6 *Iterated addition is in all of the following classes:*

1. $\mathrm{SUM} \circ \mathrm{TC}_2^0$
2. $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{NC}_1^0 \circ \mathrm{TC}_1^0$
3. $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_1^0 \circ \mathrm{NC}_1^0$

The $\mathrm{SUM} \circ \mathrm{TC}_2^0$ circuit also improves on the TC_3^0 circuit of Siu and Bruck (1991), but in a way different than that of the $\Delta_1 \circ \mathrm{TC}_2^0$ circuit of Corollary 3.4. In fact, an intermediate result of Siu and Bruck (1991) is that iterated addition can be approximated as the sum of the outputs of a TC_2^0 circuit. Our $\mathrm{SUM} \circ \mathrm{TC}_2^0$ circuit computes iterated addition exactly and the underlying proof is much simpler.

3.3 Multiplication

multiplication

input	Two n-bit numbers
output	Their $(2n)$-bit product

It is well-known that multiplication can be easily reduced to an iterated addition.

Proposition 3.7 *If iterated addition is in* Γ, *then multiplication is in* $\Gamma \circ \mathrm{NC}_1^0$.

By combining this result with the iterated addition circuits of the previous section, we get several multiplication circuits. In particular, multiplication is in $\Delta_2 \circ \mathrm{TC}_1^0 \circ \mathrm{NC}_1^0$ and in $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_1^0 \circ \mathrm{NC}_1^0$. Note that the TC_2^0 addition circuit of Siu and Roychowdhury (1994) gives a $\mathrm{TC}_2^0 \circ \mathrm{NC}_1^0$ multiplication circuit. In terms of total depth, this is optimal, by the results of Hajnal *et al.* (1987).

Using techniques similar to those used for the problems of this section, it is possible to show that subtracting two n-bit numbers and comparing two n-bit numbers can be done in $\Delta_2 \circ \mathrm{NC}_1^0$ and in $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{NC}_1^0$, that adding $\log n$ n-bit numbers can be done in Δ_4 and in $\mathrm{SUM} \circ \mathrm{AC}_3^0$, and that multiplying a $(\log n)$-bit numbers with an n-bit number can be done in Δ_3 and in $\mathrm{SUM} \circ \mathrm{AC}_2^0$. In addition, weighted iterated addition, a variant of iterated addition in which n-bit integer weights are associated to the input numbers, can be computed in $\Delta_2 \circ \mathrm{SYM}$ and $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM}$. (For more details, see Maciel 1995.)

4 Chinese Remaindering

Chinese remaindering is the technique that was originally used by Beame *et al.* (1986) to show that iterated multiplication and division are in NC^1. It is this same technique that is used in obtaining all the small-depth threshold circuits for these two functions (Siu *et al.* 1993, Siu and Roychowdhury 1994), and it is this technique that will be used in this article. In this section, we present Chinese remaindering as a general tool for computing arbitrary integer functions using small-depth threshold circuits.

Let f be an arbitrary function from $\{0,1\}^*$ to $\mathbf{N}$. The strategy for computing f will be as follows. First, choose, for every n, m pairwise relatively prime numbers $q_1, \ldots, q_m$ such that $Q = \prod_{i=1}^m q_i > \max\{f(x) : x \in \mathrm{dom}(f) \cap \{0,1\}^n\}$. Then, given x,

A. compute $r_i = f(x) \bmod q_i$, for $i = 1, \ldots, m$

B. compute $f(x) \bmod Q$ from the residues $r_1, \ldots, r_m$

This last number is the correct value of $f(x)$ since $Q > f(x)$ implies that $f(x) \bmod Q = f(x)$.

In the above, Step A amounts to solving the following problem:

f **modulo** $q_1, \ldots, q_m$

input $x \in \mathrm{dom}(f) \subseteq \{0,1\}^*$

output The $\lceil \log q_i \rceil$-bit number $f(x) \bmod q_i$, for $i = 1, \ldots, m$

The complexity of computing f modulo $q_1, \ldots, q_m$ will of course depend on the particular function f under consideration, and on the choice of $q_1, \ldots, q_m$.

Step B, on the other hand, is independent of f. The computation specified there is possible because the residues $f(x) \bmod q_i$ determine the value of $f(x) \bmod Q$, by the Chinese Remainder Theorem. We will compute $f(x) \bmod Q$ by using the following lemma which is easily obtained from the usual constructive proof of the Chinese Remainder Theorem.

Lemma 4.1 *Let $q_1, \ldots, q_m$ be pairwise relatively prime numbers and let $Q = \prod_{i=1}^m q_i$. There are numbers $u_1, \ldots, u_m < Q$ such that, for every $y \in \mathbf{Z}$,*

$$y \equiv \sum_{i=1}^{m} u_i (y \bmod q_i) \pmod{Q}.$$

Therefore, $f(x) \bmod Q$ can be computed as $\sum_{i=1}^m u_i r_i \bmod Q$. Recalling that iterated addition can be computed in $\Delta_2 \circ \mathrm{SYM}$, we now prove the main result of this section.

Theorem 4.2 *Let $f : \{0,1\}^* \to \mathbf{N}$ and suppose that, for every n, $q_1, \ldots, q_m$ are pairwise relatively prime numbers such that*

a) $\max\{q_1, \ldots, q_m\} \in n^{O(1)}$

b) $Q = \prod_{i=1}^m q_i > \max\{f(x) : x \in \{0,1\}^n\}$

Let Γ be a class of functions closed under polynomial-size padding.[6] *Then, if f modulo $q_1, \ldots, q_m$ is in Γ, then f is in $\Delta_2 \circ \mathrm{SYM} \circ \Gamma$.*

Proof Let x be an arbitrary input of length n, let $r_i = f(x) \bmod q_i$, $i = 1, \ldots, m$, and let $z = \sum_{i=1}^m u_i r_i$, the u_i being given by Lemma 4.1. According to the above discussion, we simply have to compute $z \bmod Q$.

Clearly, $z \bmod Q = z - kQ$ for some $k \in \mathbf{N}$. Since $z < mQ \max\{q_i\}$, it must be that $k < m \max\{q_i\}$. The fact that the q_i are pairwise relatively prime implies that they are distinct. Therefore, $m \leq \max\{q_i\}$ and $k \leq n^c$, for some constant c. For $j = 0, \ldots, n^c$, let $z_j = z - jQ$. Then, $z_j = z_k = z \bmod Q$ if and only if $0 \leq z_j < Q$. Therefore, $z \bmod Q$ can be computed as follows:

[6] In particular, Γ can be any of the complexity classes defined in this article. (See Section 2.4.)

1. compute $z_j = (\sum_{i=1}^{n} u_i r_i) - jQ$, for $j = 0, \ldots, n^c$
2. output z_j iff $0 \leq z_j < Q$

Step 1 is a weighted iterated addition whose inputs are the r_i. Recall that the u_i are fixed numbers that depend only on Q. Since $Q \leq (\max\{q_i\})^m \in 2^{n^{O(1)}}$ and $u_i < Q$, there is $N \in n^{O(1)}$ such that 2^N is greater than both $n^c Q$ and the maximum possible value of z_j. Let

$$s_j = \left(\sum_{i=1}^{n} u_i r_i \right) + t_j + 1,$$

where t_j is the number obtained by complementing every bit of the N-bit binary representation of jQ. Since $t_j = (2^N - 1) - jQ$, we have that $s_j = 2^N + z_j$.

Therefore, representing s_j by $s_{j(N+1)} \cdots s_{j1}$, we have that $s_{j(N+1)} = 1$ iff $z_j \geq 0$ and, in that case, $s_{jN} \cdots s_{j1}$ is the binary representation of z_j. (In the case that $s_{j(N+1)} = 0$, which means that $z_j < 0$, $s_{jN} \cdots s_{j1}$ is not the binary representation of z_j. However, the value of z_j will be needed in Step 2 only when $z_j \geq 0$.)

The value of s_j can be computed by first distributing the r_i over the binary representation of the u_i. This gives a new sum whose terms are easily computed from the r_i without having to use any gates at all. Then, evaluate this sum in $\Delta_2 \circ \mathrm{SYM}$ by using the iterated addition circuit of Theorem 3.3. Let $\sigma_j = s_{j(N+1)}$.

For Step 2, notice that $z_j \geq 0$ if and only if $\sigma_j = 1$, and that $z_j < Q$ if and only if $z_{j+1} = z_j - Q < 0$ if and only if $\sigma_{j+1} = 0$. Therefore, $0 \leq z_j < Q$ if and only if $\sigma_j \wedge \overline{\sigma}_{j+1}$. Since $0 \leq z_j < Q$ can be true only for one value of j, we get that z_{ki}, the ith bit of z_k, is given by

$$z_{ki} = \mathrm{OR}_{j=0}^{n^c}(\sigma_j \wedge \overline{\sigma}_{j+1} \wedge s_{ji}).$$

This implies that z_{ki} can be computed in $\Sigma_1 \circ \mathrm{NC}_1^0$ from the σ_j and the s_{ji}. However,

$$\overline{z}_{ki} = \mathrm{OR}_{j=0}^{n^c}(\sigma_j \wedge \overline{\sigma}_{j+1} \wedge \overline{s}_{ji}),$$

which implies that z_{ki} can also be computed in $\Pi_1 \circ \mathrm{NC}_1^0$. Therefore, Step 2 can be computed in $\Delta_1 \circ \mathrm{NC}_1^0$.

Combining these two steps, we get that $z \bmod Q$ can be computed in $\Delta_1 \circ \mathrm{NC}_1^0 \circ \Delta_2 \circ \mathrm{SYM}$, which is the same as $\Delta_2 \circ \mathrm{SYM}$. □

Let us point out where the various hypotheses of this theorem were used. The fact that the q_i are pairwise relatively prime is to ensure that $f(x) \equiv z \pmod{Q}$, via the Chinese Remainder Theorem. Condition (a), requiring that the q_i be small, implies that m is small and gives the bound on the number of possible values for k. Finally, condition (b), requiring that Q be large, implies that $f(x) \bmod Q = f(x)$.

Recalling from Section 3.2 that iterated addition can also be computed in $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM}$, we show that f can be computed in $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM} \circ \Gamma$.

Theorem 4.3 *Let f, $q_1, \ldots, q_m$, and Γ be as in Theorem 4.2. If f modulo $q_1, \ldots, q_m$ is in Γ, then f is in* $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM} \circ \Gamma$.

Proof We basically show that the right z_j can be selected in POL. Recall that there is only one value of j for which $0 \le z_j < Q$. Therefore, the OR giving the value of z_{ki} can be computed in SUM, which implies that Step 2 can be computed in POL.

Therefore, f can be computed in $\mathrm{POL} \circ \mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM} \circ \Gamma$, which is equal to $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM} \circ \Gamma$, since $\mathrm{POL} \circ \mathrm{SUM} = \mathrm{POL}$ and $\mathrm{POL} \circ \mathrm{AC}_1^0 = \mathrm{SUM} \circ \mathrm{AC}_1^0$. □

Siu and Roychowdhury (1994) also used Chinese remaindering to compute iterated multiplication and division with small-depth threshold circuits. While our implementation of Chinese remaindering will lead to the best known iterated multiplication and division circuits in terms of majority-depth, their implementation leads to the best known circuits in terms of total depth. (In fact, their division circuit is optimal.)

From their work, we can extract a result similar to Theorems 4.2 and 4.3 and whose conclusion implies that $f \in \mathrm{TC}_2^0 \circ \Gamma$. A proof would go as follows. First, notice that the proof of Theorem 4.3 shows that $f \in \mathrm{POL} \circ \Lambda \circ \Gamma$, if f modulo $q_1, \ldots, q_m$ is in Γ, iterated addition is in Λ, and Λ and Γ are both closed under polynomial-size padding. Let LT_d denote the class of functions computed by general weighted threshold circuits of polynomial size and depth d. Siu and Roychowdhury (1994) showed that iterated addition is in $\mathrm{SUM} \circ \mathrm{LT}_1$. They then used a result of Goldmann *et al.* (1992) (see also Goldmann and Karpinski 1993 and Goldmann 1992) which states that every bit of an LT_1 function can be closely approximated by the sum of the outputs of a TC_1^0 circuit. This implies that every bit of iterated addition can be so approximated. Therefore, let Λ be the class of functions that can be approximated by the sum of the outputs of a TC_1^0 circuit. By observing that constant-degree polynomials of such approximations can also be approximated in the same way, we get that f can be approximated by a sum of the outputs of a $\mathrm{TC}_1^0 \circ \Gamma$ circuit. This implies that $f \in \mathrm{TC}_2^0 \circ \Gamma$.

It is a simple exercise to generalize the Chinese Remaindering technique to functions from $\{0,1\}^*$ to $\mathbf{Z}$. Condition (b) in Theorems 4.2 and 4.3 becomes $Q > \max\{2\,|f(x)| : x \in \mathrm{dom}(f) \cap \{0,1\}^n\}$. Details can be found in Maciel (1995).

5 Iterated Multiplication and Division

In this section, we use the Chinese remaindering technique of the previous section to obtain constant-depth polynomial-size threshold circuits for iterated multiplication, powering and division. We also provide, at the end of Section 5.1, a detailed comparison between our iterated multiplication circuits and the other small-depth iterated multiplication circuits that can be found in the literature.

5.1 Iterated multiplication

iterated multiplication

input n n-bit numbers

output Their (n^2)-bit product

Beame *et al.* (1986), using Chinese remaindering, were the first to show that iterated multiplication is in NC^1. Realizing that their algorithm could be implemented by TC^0 circuits (Reif 1987, Hajnal *et al.* 1987, Immerman and Landau 1989), researchers started looking for minimal-depth TC^0 circuits for iterated multiplication. In terms of total-depth, the best result obtained so far is that iterated multiplication is in TC^0_4 (Siu and Roychowdhury 1994).

However, four levels of threshold gates are not necessary. For example, the circuit of Immerman and Landau (1989) can be implemented using only three levels of threshold gates. Here, we show that iterated multiplication can be computed in $\Delta_2 \circ \mathrm{TC}^0_3$.

This will be proved using Theorem 4.2. As a consequence, we need, for every n, a sequence of pairwise relatively prime numbers satisfying the hypotheses of that theorem. For every n, consider the first n^2 prime numbers $p_1, \ldots, p_{n^2}$. Then, by the Prime Number Theorem, $\max\{p_1, \ldots, p_{n^2}\} = p_{n^2} \in O(n^2 \log n)$. Also, $Q = \prod_{i=1}^{n^2} p_i \geq 2^{n^2} > \prod_{i=1}^{n} x_i$, for every n-bit input numbers $x_1, \ldots, x_n$, so that conditions (a) and (b) of Theorem 4.2 are both satisfied.

We now need to compute iterated multiplication modulo $p_1, \ldots, p_{n^2}$ efficiently. This is done in the following lemma.

Lemma 5.1 *Iterated multiplication modulo the first n^2 prime numbers is in* $\mathrm{SYM} \circ \mathrm{SYM}$.

Before proving the lemma, let us show how it implies the desired result. By applying Theorem 4.2, we get that iterated multiplication is in $\Delta_2 \circ \mathrm{SYM} \circ \mathrm{SYM} \circ \mathrm{SYM}$. Simulating the third level of symmetric gates in $\Delta_1 \circ \mathrm{TC}^0_1 \circ \mathrm{NC}^0_1$, we get that

$$\begin{aligned}
&\text{iterated multiplication} \\
&\quad \in \Delta_2 \circ (\Delta_1 \circ \mathrm{TC}^0_1 \circ \mathrm{NC}^0_1) \circ \mathrm{SYM} \circ \mathrm{SYM} \\
&\quad = \Delta_2 \circ \mathrm{TC}^0_1 \circ \mathrm{SYM} \circ \mathrm{SYM}, \\
&\qquad\qquad \text{since } \Delta_2 \circ \Delta_1 = \Delta_2 \text{ and } \mathrm{NC}^0_1 \circ \mathrm{SYM} = \mathrm{SYM} \\
&\quad \subseteq \Delta_2 \circ \mathrm{TC}^0_1 \circ \mathrm{SUM} \circ \mathrm{TC}^0_1 \circ \mathrm{SUM} \circ \mathrm{TC}^0_1, \\
&\qquad\qquad \text{since } \mathrm{SYM} \subseteq \mathrm{SUM} \circ \mathrm{TC}^0_1 \\
&\quad = \Delta_2 \circ \mathrm{TC}^0_3, \qquad \text{since } \mathrm{TC}^0_1 \circ \mathrm{SUM} = \mathrm{TC}^0_1.
\end{aligned}$$

Proof (of the lemma) First notice that iterated multiplication modulo 2 is simply the AND of the low-order bits of the input numbers.

Now let $p \geq 3$ be a prime number and suppose that $x_1, \ldots, x_n$ are the input numbers, given in binary by $x_{in} \cdots x_{i1}$, $i = 1, \ldots, n$. Let $\mathbf{Z}_p$ denote the ring of integers modulo p. It is well-known that $\mathbf{Z}_p$ is in fact a field and that $\mathbf{Z}_p^* = \mathbf{Z}_p - \{0\}$, the multiplicative group of $\mathbf{Z}_p$, is cyclic. In particular, there is an element $g \in \mathbf{Z}_p$ such that $\mathbf{Z}_p^* \equiv \{1, g, g^2, \ldots, g^{p-2}\}$. For every i, let a_i be the unique number such that $0 \leq a_i \leq p-2$ and

$$x_i \equiv g^{a_i} \pmod{p}.$$

Then, $\prod_{i=1}^n x_i \equiv g^{a_1} \cdots g^{a_n} \equiv g^{a_1+\cdots+a_n} \pmod{p}$.

Therefore, $(\prod_{i=1}^n x_i) \bmod p$ can be computed as follows:

1. compute a_i, for $i = 1, \ldots, n$
2. compute $g^{a_1+\cdots+a_n} \bmod p$

Note that g and p are fixed and do not have to be computed.

In Step 1, the value of a_i is determined by $x_i \bmod p$. Since $x_i = \sum_{j=1}^n x_{ij} 2^{j-1}$, every bit of a_i can therefore be written as a function of $\sum_{j=1}^n (2^{j-1} \bmod p) x_{ij}$, which is a sum with weights bounded by p. Therefore, each bit of a_i can be computed by a symmetric gate of fan-in pn.

In Step 2, the number to be computed is a function of $a_1 + \cdots + a_n$, a sum of the bits of the a_i with weights bounded by p. Therefore, every bit of that number can also be computed by a symmetric gate of fan-in pn.

Since the largest prime $p_{n^2} \in O(n^2 \log n)$, all symmetric gates used in the above computations have fan-in $O(n^3 \log n)$ and so iterated multiplication modulo $p_1, \ldots, p_{n^2}$ is in SYM $\circ$ SYM. $\square$

Note that this proof uses both the fact that the p_i are small and the fact that they are prime.

We have therefore proved

Theorem 5.2 *Iterated multiplication is in* $\Delta_2 \circ \mathrm{TC}_3^0$.

We end this section with a review of the main results concerning the computation of iterated multiplication using small-depth Boolean circuits. All of these follow, more or less explicitly, the same pattern:

1. The computation of iterated multiplication modulo small primes
2. Chinese remaindering

Beame *et al.* (1986) were the first to compute iterated multiplication in this way; they implemented these two steps in NC^1. It was soon observed that this could even be done in TC^0 (Reif 1987, Hajnal *et al.* 1987, Immerman and Landau 1989).

In fact, only three levels of threshold gates are necessary. For example, Immerman and Landau (1989) implemented Chinese remaindering using the iterated addition circuit of Chandra *et al.* (1984). This requires only one level of threshold gates, the other two levels being used for the computation of iterated multiplication modulo small primes.

From there, progress was made possible first by the observation that iterated multiplication modulo small primes could be computed in $\mathrm{SYM} \circ \mathrm{SYM}$, a result that was obtained independently by Maciel and Thérien (1993) and Siu *et al.* (1993); second, by the design of iterated addition circuits more efficient than that of Chandra *et al.* (1984).

Siu *et al.* (1993) trying to minimize total depth, used the same block method that we used in Section 3.2 to reduce iterated addition to the addition of two numbers. Then, following Siu and Bruck (1991), which implies harmonic analysis results of Bruck (1990) and Bruck and Smolensky (1990), they approximated every bit of the sum of these two numbers as the sum of the outputs of a TC^0_1 circuit. The result was that every bit of iterated addition could be approximated as the sum of the outputs of a TC^0_2 circuit. Using the fact that the AND of such approximations can also be approximated in the same way, they obtained a TC^0_5 iterated multiplication circuit.

Maciel and Thérien (1993) introduced the idea of trying to minimize first the number of levels of threshold gates. Using the iterated addition circuit of Chandra *et al.* (1984), we obtained a TC^0 circuit with threshold gates on only the first three levels.

The results presented in this article improve on that result by reducing the total depth to five. This is done, as we have seen, by computing iterated addition in $\Delta_2 \circ \mathrm{SYM}$ using the block and the carry-look-ahead methods. The closure properties of Δ_2 then give a $\Delta_2 \circ \mathrm{TC}^0_3$ iterated multiplication circuit.

The best iterated multiplication circuit known so far, in terms of total depth, was obtained by Siu and Roychowdhury (1994). They improved the circuit of Siu *et al.* (1993) by using results of Goldmann *et al.* (1992) to approximate every bit of iterated addition as the sum of the outputs of a TC^0_1 circuit. This gave a TC^0_4 circuit.

By using the $\mathrm{SUM} \circ \mathrm{AC}^0_1 \circ \mathrm{SYM}$ iterated addition circuit of Section 3.2, we can prove a slightly stronger result than that of Siu *et al.* (1993) with a proof that requires neither harmonic analysis nor the definition of a notion of approximation.

Theorem 5.3 *Iterated multiplication is in* $\mathrm{SUM} \circ \mathrm{AC}^0_1 \circ \mathrm{TC}^0_3$.

Proof Simply use Lemma 5.1 together with Theorem 4.3. □

Table 1 gives a summary of the iterated multiplication circuits mentioned in this section. The third column lists the main ingredients used in the design of the circuits. Note that all of these circuits use Chinese remaindering and that the last four all use the fact that iterated multiplication modulo small primes is in $\mathrm{SYM} \circ \mathrm{SYM}$.

	Result	Main ingredients
Beame *et al.* (1986)	NC^1	iterated addition $\in \mathrm{NC}^1$ (e.g., Savage 1976)
Immerman and Landau (1989)	TC^0	iterated addition $\in \mathrm{TC}^0$ (Chandra *et al.* 1984)
Maciel and Thérien (1993)	$\mathrm{AC}^0 \circ \mathrm{TC}^0_3$	iterated addition $\in \mathrm{AC}^0 \circ \mathrm{TC}^0_1$ (Chandra *et al.* 1984)
Siu *et al.* (1993)	TC^0_5	blocks harmonic analysis
Siu and Roychowdhury (1994)	TC^0_4	large weights simulation of large weights (Goldmann *et al.* 1992)
This article	$\Delta_2 \circ \mathrm{TC}^0_3$ $\mathrm{SUM} \circ \mathrm{AC}^0_1 \circ \mathrm{TC}^0_3$	blocks carry-look-ahead

Table 1: Summary of iterated multiplication circuits

5.2 Powering

The following problem is often considered in conjunction with iterated multiplication.

powering

input An n-bit number x

output The kn-bit number x^k, for $k = 2, \ldots, n$

The computation of x^k can be viewed as a special case of iterated multiplication in which the first k numbers are equal to x and the other $n-k$, equal to one. As such, powering can be computed using an iterated multiplication circuit. However, we can take advantage of the special structure of the problem.

Lemma 5.4 *Powering modulo $p_1, \ldots, p_{n^2}$ is in* SYM.

Proof Suppose that the input number x is given in binary by $x_n \cdots x_1$. Let k and q be arbitrary. Since $x^k \equiv (x \bmod q)^k \pmod q$, the value of $x^k \bmod q$ is determined by $x \bmod q$. Since $x \equiv \sum_{j=1}^{n} (2^{j-1} \bmod q) x_j \pmod q$, we have that every bit of $x^k \bmod q$ is a function of a sum with weights bounded by q. Therefore, by Proposition 2.4, every bit of $x^k \bmod q$ can be computed by a symmetric gate of fan-in qn. The result now follows from the fact that $p_{n^2} \in n^{O(1)}$. $\square$

Theorem 5.5 *Powering is in* $\Delta_2 \circ \mathrm{TC}^0_2$.

Proof Consider Theorem 4.2 with $f(x) = x^k$, $k = 2, \ldots, n$. Recall that the $\Delta_2 \circ \mathrm{SYM}$ circuit that performs Chinese remaindering depends on $p_1, \ldots, p_{n^2}$, but not on f. Therefore, the same $\Delta_2 \circ \mathrm{SYM}$ circuit can be used to compute every x^k by giving it as input the residues $x^k \bmod p_1, \ldots, x^k \bmod p_{n^2}$. Together with the lemma, this implies that powering is in $\Delta_2 \circ \mathrm{SYM} \circ \mathrm{SYM}$, which is equal to $\Delta_2 \circ \mathrm{TC}_2^0$. $\square$

Similarly, we can show that

Theorem 5.6 *Powering is in* $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_2^0$.

In comparison, the results of Siu and Roychowdhury (1994) imply a TC_3^0 circuit for powering that uses threshold gates on all levels. In terms of total depth, their circuit is optimal (Hofmeister and Pudlák 1992).

5.3 Division

division

input Two n-bit numbers x and y such that $y > 0$

output The n-bit number $\lfloor x/y \rfloor$

The design of small-depth Boolean circuits for division parallels that of circuits for iterated multiplication and powering. As in the case of these other two functions, the first NC^1 division circuit was obtained by Beame *et al.* (1986) and the best TC^0 circuit, in terms of total depth, is due to Siu and Roychowdhury (1994). They showed that division is in TC_3^0 and this is optimal as shown by Hofmeister and Pudlák (1992).

The TC_3^0 circuit of Siu and Roychowdhury (1994) uses threshold gates on all levels. As was the case for iterated multiplication and powering, we show here that division can be computed using one less level of threshold gates. More precisely, we show that division is in $\Delta_2 \circ \mathrm{TC}_2^0$.

The idea behind our circuit is the same as in Beame *et al.* (1986) and Siu and Roychowdhury (1994). The quotient x/y is approximated using a power series for $1/y$ and this approximation is computed by Chinese remaindering. We will also make use of the following lemma.

Lemma 5.7 *Let x and y be two integers such that $|y| < 2^n$. If $x/y \le t \le x/y + 2^{-n}$, then $\lfloor t \rfloor = \lfloor x/y \rfloor$.*

Proof If $x/y \le t \le x/y + 2^{-n}$, then $\lfloor t \rfloor = \lfloor x/y \rfloor$, unless there is an integer s such that $x/y < s \le t$, which implies that there is an integer s different from x/y and such that $|s - x/y| \le 2^{-n}$.

Suppose that such an integer s exists. Since $|y| < 2^n$, we have that $|sy - x| < 1$. However, since s, x and y are all integers, this implies that $sy = x$, a contradiction. $\square$

Theorem 5.8 *Division is in* $\Delta_2 \circ \mathrm{TC}_2^0$.

Proof Suppose that x and y are the input numbers, given in binary by $x_n \cdots x_1$ and $y_n \cdots y_1$. Let $l \in \mathbf{N}$ be such that $2^{l-1} \leq y < 2^l$. In other words, l is the minimum number of bits needed to represent y in binary. Of course, $1 \leq l \leq n$. Let $u = 1 - y2^{-l}$. Then, $0 < u \leq 1/2$. Now consider the power series $\sum_{k=0}^{\infty} z^k$. It is well-known that this series converges to $1/(1-z)$ for $|z| < 1$. Therefore,

$$\sum_{k=0}^{\infty} u^k = \frac{1}{1-u} = \frac{2^l}{y}$$

and it is easy to verify that

$$\frac{1}{y} - 2^{-l} \sum_{k=0}^{N-1} u^k \leq 2^{-N}.$$

We can therefore approximate $1/y$ with arbitrary precision by taking N sufficiently large. Let $v = 2^{-l} \sum_{k=0}^{2n-1} u^k$. We get that $1/y - 2^{-2n} \leq v < 1/y$. Let $t = xv$. Then $x/y - 2^{-n} < t < x/y$, since $x < 2^n$, and $x/y < t + 2^{-n} < x/y + 2^{-n}$.

Consider $\lfloor t + 2^{-n} \rfloor$. According to the lemma, $\lfloor t + 2^{-n} \rfloor = \lfloor x/y \rfloor$. Division can therefore be computed as follows:

1. compute $F_j(x,y) = 2^{-n} + x2^{-j} \sum_{k=0}^{2n-1} u^k$,
   where $u = 1 - y2^{-j}$, for $j = 1, \ldots, n$
2. output $\lfloor F_j(x,y) \rfloor$ if $j = l$

The computation of the $F_j(x,y)$ will be done by Chinese remaindering. Let $G_j(x,y) = 2^{2n^2} F_j(x,y)$ so that $G_j(x,y)$ is an integer. Notice that $F_j(x,y)$ can be computed from the value of $G_j(x,y)$ without having to use any gates at all. Let $m = 5n^3$ and consider the first m prime numbers $p_1, \ldots, p_m$. It is easy to verify that the hypotheses of the integer version of Theorem 4.2 are satisfied (see the remark at the end of Section 4).

The rest is similar to the computation of powering in $\Delta_2 \circ \mathrm{TC}_2^0$ (see Lemma 5.4 and Theorem 5.5). Since

$$G_j(x,y) = 2^{2n^2-n} + x2^{n^2-j} \sum_{k=0}^{2n-1} (2^{n^2} - y2^{n^2-j})^k,$$

the value of $G_j(x,y) \bmod q$ is determined by $x \bmod q$ and $y \bmod q$. Therefore, every bit of $G_j(x,y) \bmod q$ is a function of two sums with weights bounded by q and thus can be computed by a symmetric gate of fan-in $(4qn)^2$, by combining Proposition 2.4 and Theorem 2.5. This implies that computing all the G_j modulo $p_1, \ldots, p_m$ can be done in SYM.

Theorem 4.2 now implies that each $G_j(x,y)$ can be computed in $\Delta_2 \circ \mathrm{SYM} \circ \mathrm{SYM}$. Since the $\Delta_2 \circ \mathrm{SYM}$ subcircuit depends only on $p_1, \ldots, p_m$, not on G_j, the same subcircuit can be used in the computation of every $G_j(x, y)$. Therefore, computing all the $G_j(x, y)$ can be done in $\Delta_2 \circ \mathrm{SYM} \circ \mathrm{SYM}$, which is equal to $\Delta_2 \circ \mathrm{TC}_2^0$.

To choose $F_l(x, y)$ among $F_1(x, y), \ldots, F_n(x, y)$, let

$$\lambda_j = \mathrm{AND}(\overline{y}_n, \ldots, \overline{y}_{j+1}, y_j).$$

Then, $\lambda_j = 1$ if and only if $2^j \leq y < 2^{j+1}$ which means that $j = l$. Therefore, $F_{li}(x, y)$, the ith bit of $F_l(x, y)$, can be computed as

$$F_{li}(x, y) = \mathrm{OR}_{j=1}^{n}(\lambda_j \wedge F_{ji}(x, y)).$$

Since we also have that

$$\overline{F}_{li}(x, y) = \mathrm{OR}_{j=1}^{n}(\lambda_j \wedge \overline{F}_{ji}(x, y)),$$

this means that F_l can be computed in $\Delta_1 \circ \mathrm{NC}_1^0$ from the λ_j and the F_j. Notice that the λ_j are computed in Π_1 which is trivially included in Δ_2. Therefore, F_l can be computed in $\Delta_1 \circ \mathrm{NC}_1^0 \circ \Delta_2 \circ \mathrm{TC}_2^0$, which is equal to $\Delta_2 \circ \mathrm{TC}_2^0$. To obtain $\lfloor x/y \rfloor$, simply truncate the value of $F_l(x, y)$. $\Box$

Theorem 5.9 *Division is in* $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_2^0$.

Proof Use Theorem 4.2 instead of Theorem 4.3 to compute all the $G_j(x, y)$ in $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{SYM} \circ \mathrm{SYM}$, which is the same as $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_2^0$. Now notice that there is only one value of j for which $\lambda_j = 1$. Therefore, the OR giving the value of F_{li} can be computed in SUM. This implies that F_l can be computed in POL given the λ_j and the F_j. Therefore, F_l can be computed in $\mathrm{POL} \circ \mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_2^0$, which is equal to $\mathrm{SUM} \circ \mathrm{AC}_1^0 \circ \mathrm{TC}_2^0$, since $\mathrm{POL} \circ \mathrm{AC}_1^0 = \mathrm{SUM} \circ \mathrm{AC}_1^0$. $\Box$

6 A Hierarchy for TC^0

A main theme of this article is that the main measure of complexity for TC^0 circuits is taken to be the number of levels of threshold gates and not the total depth of the circuit. This point of view naturally leads to a hierarchy of subclasses of TC^0.

Recall that a TC^0 circuit is a polynomial-size constant-depth circuit with NOT, AND, OR and majority gates of unbounded fan-in. Define the *majority-depth* of such a circuit to be the maximum number of majority gates on any path from an input node to an output gate. Circuits of the form $\mathrm{AC}_2^0 \circ \mathrm{TC}_3^0$, for example, have majority-depth three.

Definition 6.1 $\widehat{\mathrm{TC}}_d^0$ *is the class of functions computed by* TC^0 *circuits of majority-depth* d.

Clearly, $\widehat{\mathrm{TC}}_d^0 = (\mathrm{AC}^0 \circ \mathrm{MAJ})^d \circ \mathrm{AC}^0$, where Γ^d denotes $\Gamma \circ \Gamma \circ \cdots \circ \Gamma$, d times. For this reason, we say that $\widehat{\mathrm{TC}}_d^0$ circuits are TC^0 circuits with d levels of majority

	best majority-depth	best total depth
iterated multiplication	3 (IL 1989)	4 (SR 1994)
division	2 (MT 1993)	3 (SR 1994)
powering	2 (MT 1993)	3 (SR 1994)
multiplication	1 (CSV 1984)	3 (SR 1994)
iterated addition	1 (CSV 1984)	2 (SR 1994)
SYM	1	2 (HMPST 1987)
small-weight threshold functions	1	1
addition	0	2 (SB 1991)

Table 2: Some problems in the first three levels of the $\widehat{\mathrm{TC}}^0_d$ hierarchy

gates. It is obvious that $\mathrm{TC}^0_d \subseteq \widehat{\mathrm{TC}}^0_d$. On the other hand, it follows from the definition that $\widehat{\mathrm{TC}}^0_d \subseteq \mathrm{TC}^0$. Therefore, $\mathrm{TC}^0 = \bigcup_{d\geq 1} \widehat{\mathrm{TC}}^0_d$.

In this article, we have constructed TC^0 circuits for several Boolean functions with the objective of minimizing the majority-depth of the circuits. As a consequence, these results imply that the low levels of the $\widehat{\mathrm{TC}}^0_d$ hierarchy contain all of these functions. For example, since iterated multiplication is in $\mathrm{AC}^0_2 \circ \mathrm{TC}^0_3$, we have that this function is in $\widehat{\mathrm{TC}}^0_3$. Table 2 summarizes the results that can be obtained in this way. For each problem or class of problems, the best total-depth result is also included. The references given are for the first proof of the results.

There is a close relationship between the $\widehat{\mathrm{TC}}^0_d$ hierarchy and the usual depth hierarchy for TC^0. As mentioned earlier, it is obvious that $\mathrm{TC}^0_d \subseteq \widehat{\mathrm{TC}}^0_d$. However, it is not clear how large k has to be for $\widehat{\mathrm{TC}}^0_d \subseteq \mathrm{TC}^0_k$, or even if such a k exists. A partial solution comes from the simulation of AC^0 circuits by threshold circuits of fixed small depth. If Γ is a complexity class defined in terms of polynomial-size circuits, let $\mathrm{q}\Gamma$ denote its quasipolynomial-size version (i.e., size $n^{(\log n)^{O(1)}}$) and let $\mathrm{q}\Gamma^+$ denote the class $\mathrm{q}\Gamma \circ \mathrm{qAND}_{(\log n)^{O(1)}}$. For example, qSYM^+ denotes the class of functions computed by quasipolynomial-size depth-two circuits with symmetric gates at the output and AND gates of fan-in $(\log n)^{O(1)}$ at the input. Results of Beigel and Tarui (1994) imply that $\mathrm{MAJ} \circ \mathrm{AC}^0 \subseteq \mathrm{qSYM}^+$. Using this and the fact that $\mathrm{qAND}_{(\log n)^{O(1)}} \circ \mathrm{qSYM} \subseteq \mathrm{qSYM}^+$ (Corollary 2.6), it is possible to show that

$$\mathrm{q}\widehat{\mathrm{TC}}^0_d \subseteq \mathrm{qTC}^0_{d+2}{}^+.$$

Details can be found in Maciel (1995). One consequence of this result is that the design of circuits with small majority-depth can be used as a first step in constructing quasipolynomial-size circuits of small total depth.

Whenever faced with a hierarchy $\Gamma_1 \subseteq \Gamma_2 \subseteq \Gamma_3 \subseteq \cdots$ of complexity classes, it is natural to try to determine if the hierarchy is infinite or if it collapses, i.e., if

$\bigcup_{d\geq 1} \Gamma_d = \Gamma_k$, for some k. In the case of the AC^0 depth hierarchy, this question has been answered: the AC^0 depth hierarchy is infinite and the underlying lower bounds are exponential (Sipser 1983, Håstad 1986). In the case of the TC^0 depth hierarchy, the question remains open. It is known that $\mathrm{TC}^0_1 \subset \mathrm{TC}^0_2 \subset \mathrm{TC}^0_3$ (Hajnal *et al.* 1987), all with underlying exponential lower bounds, but it is not known if $\mathrm{TC}^0 = \mathrm{TC}^0_3$. Now consider this question in the context of the $\widehat{\mathrm{TC}}^0_d$ hierarchy. Unfortunately, no non-trivial lower bound is known even for $\widehat{\mathrm{TC}}^0_1$, the first level of this hierarchy.

A natural candidate for a function not in $\widehat{\mathrm{TC}}^0_1$ is the canonical depth-two majority function. Divide the input variables into $\sqrt{n}$ blocks of $\sqrt{n}$ variables. Let M_i denote the majority of the ith block and define $\mathrm{MAJ}^{(2)}(x_1, \ldots, x_n) = \mathrm{MAJ}(M_1, \ldots, M_{\sqrt{n}})$. Since this function is the typical function defined using two levels of majority gates, it is reasonable to conjecture that it cannot be computed with a single level of majority gates, even with the help of an arbitrary but constant number of levels of AND-OR gates. In other words, we conjecture that $\mathrm{MAJ}^{(2)} \notin \widehat{\mathrm{TC}}^0_1$. As evidence in favor of this conjecture, it can be shown, again by using $\mathrm{MAJ} \circ \mathrm{AC}^0 \subseteq \mathrm{qSYM}^+$ (Beigel and Tarui 1994), that $\mathrm{MAJ}^{(2)} \in \widehat{\mathrm{TC}}^0_1$ would imply that $\mathrm{TC}^0 \subseteq \mathrm{q}\widehat{\mathrm{TC}}^0_1$ and therefore that $\mathrm{TC}^0 \subseteq \mathrm{qTC}^0_4$.

In fact, the problem of showing a lower bound result for $\widehat{\mathrm{TC}}^0_1$ might prove to be a very difficult one. Let $\mathrm{ACC}^0[m]$ denote the class of functions computed by constant-depth polynomial-size circuits with AND, OR and MOD_m gates of unbounded fan-in. Let $\mathrm{ACC}^0 = \bigcup_{m\geq 2} \mathrm{ACC}^0[m]$. Beigel and Tarui (1994) showed that $\mathrm{ACC}^0 \subseteq \mathrm{qSYM}^+$. Since $\mathrm{qSYM}^+ \subseteq \mathrm{q}\widehat{\mathrm{TC}}^0_1$, a super-quasipolynomial lower bound for $\widehat{\mathrm{TC}}^0_1$ would immediately translate into a lower bound for ACC^0, solving a problem that has been open for almost ten years now.

An approach towards proving a lower bound result for $\widehat{\mathrm{TC}}^0_1$ is to first consider subclasses of $\widehat{\mathrm{TC}}^0_1$. One candidate is the class $\mathrm{MAJ} \circ \mathrm{AC}^0$. Zhang *et al.* (1993) have determined exactly which symmetric functions can be computed by such circuits; in particular, $\mathrm{MAJ} \circ \mathrm{AC}^0$ circuits cannot compute any of the MOD_m functions. Since these functions are all easily computed by $\widehat{\mathrm{TC}}^0_1$ circuits, the class $\mathrm{MAJ} \circ \mathrm{AC}^0$ therefore seems too weak to give much insight about $\widehat{\mathrm{TC}}^0_1$.

Another candidate is the class $\mathrm{AC}^0 \circ \mathrm{MAJ}$. In this class, we find all symmetric functions and iterated addition, but it is not known if $\mathrm{AC}^0 \circ \mathrm{MAJ}$ circuits can compute multiplication or inner product mod 2. Note that both these functions are in $\mathrm{AC}^0 \circ \mathrm{MAJ} \circ \mathrm{NC}^0_1$. In fact, no non-trivial lower bound is known for $\mathrm{AC}^0 \circ \mathrm{MAJ}$; it is not even known if $\mathrm{NP} \subseteq \mathrm{AC}^0 \circ \mathrm{MAJ}$. The function $\mathrm{MAJ}^{(2)}$ is again a candidate for a function not in $\mathrm{AC}^0 \circ \mathrm{MAJ}$: it can be shown that $\mathrm{MAJ}^{(2)} \in \mathrm{AC}^0 \circ \mathrm{MAJ}$ would imply that $\mathrm{TC}^0 \subseteq \mathrm{AC}^0 \circ \mathrm{MAJ}$ and $\mathrm{TC}^0 \subseteq \mathrm{qTC}^0_3$.

Acknowledgments

We thank Richard Beigel for useful comments and François Lemieux for many helpful discussions.

References

Ajtai, M. (1983), Σ_1^1 formulae on finite structures, *Ann. Pure Appl. Logic* **24**, 1–48.

Ajtai, M., and Ben-Or, M. (1984), A theorem on probabilistic constant depth computations, *in* "Proceedings, 16th ACM Symposium on Theory of Computing," pp. 471–474.

Alon, N., and Bruck, J. (1994), Explicit constructions of depth-2 majority circuits for comparison and addition, *SIAM J. Discrete Math.* **7**, 1–8.

Barrington, D.A. (1986), A note on a theorem of Razborov, Technical Report COINS TR 87–93, Dept. of Computer Science, University of Massachusetts.

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

Beigel, R. (1994), When do extra majority gates help? polylog(n) majority gates are equivalent to one, *Comput. Complexity* **4**, 314–324.

Beigel, R. (1993), The polynomial method in circuit complexity, *in* "Proceedings, 7th IEEE Structure in Complexity Theory Conference," pp. 82–95.

Beigel, R., and Tarui, J. (1994), On ACC, *Comput. Complexity* **4**, 350–366.

Bruck, J. (1990), Harmonic analysis of polynomial threshold functions., *SIAM J. Discrete Math.* **3**, 168–177.

Bruck, J., and Smolensky, R. (1990), Polynomial threshold functions, AC^0 functions and spectral norms, *in* "Proceedings, 31st IEEE Symposium on Foundations of Computer Science," pp. 632–641.

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

Denenberg, L., Gurevich, Y., and Shelah, S. (1986), Definability by Constant-Depth Polynomial-Size Circuits, *Inform. and Control* **70**, 216–240.

Fagin, R., Klawe, M., Pippenger, N.J., and Stockmeyer, L. (1985), Bounded-depth polynomial-size circuits for symmetric functions, *Theoret. Comput. Sci.* **36**, 239–250.

Furst, M., Saxe, J.B., and Sipser, M. (1984), Parity, circuits, and the polynomial-time hierarchy, *Math. Systems Theory* **17**, 13–27.

Goldmann, M. (1992), On Threshold Circuits and Related Models of Computation, Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden.

Goldmann, M., Håstad, J., and Razborov, A. (1992), Majority gates vs. general weighted threshold gates, *in* "Proceedings, 7th IEEE Structure in Complexity Theory Conference," pp. 2–13. To appear in *Comput. Complexity.*

Goldmann, M., and Karpinski, M. (1993), Simulating threshold circuits by majority circuits, *in* "Proceedings, 25th ACM Symposium on Theory of Computing," pp. 551–560.

Hajnal, A., Maass, W., Pudlák, P., Szegedy, M., and Turán, G. (1993), Threshold circuits of bounded depth, *J. Comput. System Sci.* **46**, 129–154.

Håstad, J. (1986), "Computational Limitations of Small Depth Circuits," MIT Press.

Hofmeister, T., Hohberg, W., and Köhling, S. (1991), Some notes on threshold circuits, and multiplication in depth 4, *Inform. Process. Lett.* **39**, 219–225.

Hofmeister, T., and Pudlák, P. (1992), A Proof that division is not in TC_2^0, Research report no. 447, Lehrstuhl Informatik II, Universität Dortmund, Germany.

Immerman, N., and Landau, S. (1989), The complexity of iterated multiplication, *in* "Proceedings, 4th IEEE Structure in Complexity Theory Conference," pp. 104–111.

Maciel, A. (1995), Threshold Circuits of Small Majority-Depth, Ph.D. Thesis, McGill University, Montréal, Québec, Canada.

Maciel, A., and Thérien, D. (1993), Threshold Circuits for Iterated Multiplication: Using AC^0 for Free, *in* "Proceedings, 10th Symposium on Theoretical Aspects of Computer Science," pp. 545–554, Lecture Notes in Computer Science, No. 665, Springer-Verlag, Berlin.

Papadimitriou, C.H., and Zachos, S.K. (1983), Two remarks on the power of counting, *in* "Proceedings, 6th GI Conference on Theoretical Computer Science," pp. 269–275, Lecture Notes in Computer Science, No. 145, Springer-Verlag, Berlin.

Razborov, A.A. (1987), Lower bounds on the size of bounded-depth networks over a complete basis with logical addition, *Mathematical Notes of the Academy of Sciences of the USSR* **41**, 333–338.

Reif, J.H. (1987), On threshold circuits and polynomial computation, *in* "Proceedings, 2nd IEEE Structure in Complexity Theory Conference," pp. 118–123.

Savage, J.E. (1976), "The Complexity of Computing," Wiley.

Sipser, M. (1983), Borel sets and circuit complexity, *in* "Proceedings, 15th ACM Symposium on Theory of Computing," pp. 61–69.

Siu, K.-Y., Bruck, J. (1991), On the power of threshold circuits with small weights, *SIAM J. Discrete Math.* **4**, 423–435.

Siu, K.-Y., Bruck, J., Kailath, T., and Hofmeister, T. (1993), Depth-efficient neural networks for division and related problems, *IEEE Trans. Inform. Theory* **39**, 946–956.

Siu, K.-Y., and Roychowdhury, V. (1994), On optimal depth threshold circuits for multiplication and related problems, *SIAM J. Discrete Math.* **7**, 284–292.

Siu, K.-Y., Roychowdhury, V., and Kailath. T. (1991), Depth-Size Tradeoffs for Neural Computation, *IEEE Trans. Comput.* **40**, 1402–1412.

Smolensky, R. (1987), Algebraic methods in the theory of lower bounds for Boolean circuit complexity, *in* "Proceedings, 19th ACM Symposium on Theory of Computing," pp. 77–82.

Yao, A.C.-C. (1985), Separating the polynomial-time hierarchy by oracles, *in* "Proceedings, 26th IEEE Symposium on Foundations of Computer Science," pp. 1–10.

Zhang, Z.-L., Barrington, D.A.M., Tarui, J.: Computing symmetric functions with AND/OR circuits and a single MAJORITY gate. *in* "Proceedings, 10th Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, No. 665, Springer-Verlag, Berlin.