The Graph of Multiplication is Equivalent to Counting

Samuel R. Buss*

Department of Mathematics

University of California, San Diego

November 12, 1991

Abstract

Counting is AC^0-reducible to the graph of multiplication. Hence the graph of multiplication is equivalent under AC^0 reductions to majority and to the function form of multiplication.

Keywords: constant depth circuits, threshhold circuits, circuit complexity, graph of multiplication, computational complexity.

Chandra, Stockmeyer and Vishkin [4] showed that MULTIPLICATION and BINARY-COUNT and four other problems are equivalent with respect to constant-depth circuit reducibility (see also Furst, Saxe and Sipser [5]). In this note we prove that the graph of multiplication is also equivalent to these six problems, by proving that BINARY-COUNT is AC^0-reducible to the graph of multiplication.

The problems we are most interested in are:

MULTIPLICATION:

Input: Two integers x, y in binary notation

Output: The product $x \cdot y$ in binary notation

MULTIPLICATION-GRAPH:

Input: Three integers x, y and z in binary notation

Output: Accept if and only if $x \cdot y = z$.

BINARY-COUNT:

Input: $x_1, \ldots, x_n \in \{0, 1\}$

Output: The binary representation of the number of x_i's equal to 1

*Supported in part by NSF Grant DMS-8902480. Email address: `sbuss@ucsd.edu`.

The prior constant-depth circuit reduction of BINARY-COUNT to MULTIPLICATION is based on a method of Furst-Saxe-Sipser. For inputs $x_1, \ldots, x_n$, construct the integers $\overline{x}$, $\overline{y}$ with binary representations $\overline{x} = (x_1\vec{0}x_2\vec{0}\cdots x_{n-1}\vec{0}x_n)_2$ and $\overline{y} = (1\vec{0}1\vec{0}\cdots 1\vec{0}1)_2$ where $\vec{0}$ denotes $\log(n)$ many 0's. In the middle of the binary representation of the product $\overline{x}\cdot\overline{y}$ will be a block of $\log(n)$ many bits containing the binary representation of the sum of the x_i's. However, the binary representation of $\overline{x}\cdot\overline{y}$ contains a lot of additional information; and it is easy to see that there is a constant depth circuit which, given $x_1, \ldots, x_n$ and given an integer z, determines if $z = \overline{x}\cdot\overline{y}$. Thus, the prior work left open the possibility that there are constant depth circuits for recognizing the graph of multiplication—this paper shows that such circuits do not exist.

Part of the reason we are interested in the non-existence of constant depth circuits for the graph of multiplication is that this implies the Δ_0^b-predicates of bounded arithmetic [2] are not in AC^0; this should be contrasted with the work of Mancivi on Δ_0^b-formulas in one free variable [7].

Recall the following theorem:

Theorem 1 *(Chandra-Stockmeyer-Vishkin [4])*

(a) *MULTIPLICATION* $\leq_{AC^0}$ *BINARY-COUNT*

(b) *BINARY-COUNT* $\leq_{AC^0}$ *MULTIPLICATION*

Obviously MULTIPLICATION-GRAPH $\leq_{AC^0}$ MULTIPLICATION. For the converse reduction we shall prove:

Theorem 2 *BINARY-COUNT* $\leq_{AC^0}$ *MULTIPLICATION-GRAPH.*

Recall that TC^0 is the set of predicates which can be computed by a family of unbounded fanin, constant depth circuits formed from NOT gates, AND gates and MAJORITY gates. In [4] it is also shown that MAJORITY is AC^0-equivalent to MULTIPLICATION, etc.

Corollary 3 *The decision problem MULTIPLICATION-GRAPH is* AC^0 *equivalent to MULTIPLICATION and to BINARY-COUNT. Unbounded fanin, constant depth circuits with* NOT *gates,* AND *gates and gates for the graph of multiplication capture exactly* TC^0*.*

Proof The proof of Theorem 2 involves a modification of the Furst-Saxe-Sipser method. Now we let

$$\overline{x} = (x_1 0^{k_1-1} x_2 0^{k_2-1} x_3 \cdots 0^{k_{n-1}-1} x_n)_2$$

and

$$\overline{y} = (10^{k_{n-1}-1}10^{k_{n-2}-1}1\cdots 0^{k_1-1}1)_2$$

where 0^k denotes k occurences of the symbol 0. As before the product $\overline{x}\cdot\overline{y}$ will contain a block of bits encoding the sum of the x_i's; however, the values of k_i will be chosen that the rest of the product is easily predictable and thus can not aid in determining the sum of the x_i's.

Let z be the (binary representation of the) product $\overline{x}\cdot\overline{y}$. The computation of z by the usual multiplication algorithm reduces to computing the summation of the values $\overline{x}\cdot 2^{m_j}$ where $m_j = \sum_{i=1}^{j} k_i$ for $0 \le j < n$. We shall choose values for k_i so that each column of this summation has at most a single "1" except, of course, that the middle column will contain all the bits $x_1, \ldots, x_n$. In addition, each k_j will be greater than $\log n$ and thus the sum of the x_i's in the middle column will not propogate any carries into a column containing a "1". Let $C = \sum_{i=0}^{n-1} k_i$, it follows that columns C through $C + \log n$ of z contain the binary representation of the number of x_i's equal to one. (Columns are numbered from right-to-left starting with 0.)

We define

$$k_j \;=\; n^2 + j.$$

Since x_j occurs in column $\sum_{i=j}^{n-1} k_i$ of $\overline{x}$ and since $\overline{y}$ has a "1" in precisely columns $\sum_{i=1}^{m-1} k_i$ for $1 \le m \le n$, the summation for z contains the bit x_j in the columns numbered

$$g(j,m) = \sum_{i=1}^{m-1} k_i + \sum_{i=j}^{n-1} k_i$$

for $1 \le m \le n$. When $j = m$, $g(j,m) = C$; so each x_j occurs in column C of the summation. To show that in all other columns there is at most a single non-zero bit, it suffices to show that if (j,m) and (j',m') are distinct pairs with $j \ne m$ then $g(j,m) \ne g(j',m')$. Without loss of generality, $m' < m \le j < j'$ and thus it suffices to show that

$$\sum_{i=m'}^{m-1} k_i \;\ne\; \sum_{i=j}^{j'-1} k_i$$

i.e., that

$$\sum_{i=m'}^{m-1} n^2 + i \;\ne\; \sum_{i=j}^{j'-1} n^2 + i.$$

But since $\sum_{i=1}^{n-1} i < n^2$, it must be that the two summations have the same number of terms; which is impossible.

We have shown that for $\overline{x}$ and $\overline{y}$ as above, the product z contains the number of nonzero x_i's in columns C through $C + \log n$; the other bits of z are zero except for that for all $j \neq m$, z has bit value x_j in column $g(j,m)$. Thus BINARY-COUNT can be reduced to MULTIPLICATION-GRAPH by in parallel trying every possible number of nonzero values of x_i's and using graph-of-multiplication gates to determine if the numbers are correct. $\Box$

Let LH denote the logtime hierarchy, i.e., the set of predicates accepted by logtime, constant alternation Turing machines. LH was originally defined by Sipser and is a uniform version of AC^0; see [1, 3] for more information. To conclude, we observe that the constant depth circuits reducing BINARY-COUNT to the graph of multiplication are LH-uniform. To see this, it suffices to note that since

$$g(j,m) = n^2(n+k-j) + \frac{k^2-k}{2} + \frac{n^2-n}{2} - \frac{j^2-j}{2}$$

$g(j,m)$ can be computed by constant depth, polynomial size, uniform circuits. This is because the multiplication function is in constant alternation, linear time (see Lipton [6] or Wilkie [8]) and because the computation of $g(j,m)$ involves addition and multiplication of numbers of length $O(\log n)$.

Acknowledgements

This construction is based, in part, on an idea due to P. Pudlák. The final form was constructed during conversations with E. Allender and H. Straubing.

References

[1] D. A. M. Barrington, N. Immerman, and H. Straubing, *On uniformity within NC^1*, J. Comput. System Sci., 41 (1990), pp. 274–306.

[2] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[3] ——, *The Boolean formula value problem is in ALOGTIME*, in Proceedings of the 19-th Annual ACM Symposium on Theory of Computing, May 1987, pp. 123–131.

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

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

[6] R. J. Lipton, *Model theoretic aspects of computational complexity*, in Proceedings of the 19th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1978, pp. 193–200.

[7] S.-G. Mancivi, *Sharply bounded predicates do not exhaust the polynomial time computable predicates, parts I&II.* Typeset manuscript, 1990.

[8] A. J. Wilkie, *Applications of complexity theory to* Σ_0*-definability problems in arithmetic*, in Model Theory of Algebra and Arithmetic, Lecture Notes in Mathematics #834, Springer-Verlag, 1979, pp. 363–369.