INFORMATION AND COMPUTATION **131**, 81–93 (1996)
ARTICLE NO. 0093

Multiple Product Modulo Arbitrary Numbers

Claudia Bertram-Kretzberg and Thomas Hofmeister

Lehrstuhl Informatik II, Universität Dortmund, D-44221 Dortmund, Germany
E-mail: {bertram,hofmeist}@Ls2.informatik.uni-dortmund.de

Let n binary numbers of length n be given. The Boolean function "Multiple Product" MP_n asks for (some binary representation of) the value of their product. It has been shown (K.-Y. Siu and V. Roychowdhury, On optimal depth threshold circuits for multiplication and related problems, *SIAM J. Discrete Math.* **7**, 285–292 (1994)) that this function can be computed in polynomial-size threshold circuits of depth 4. For many other arithmetic functions, circuits of depth 3 are known. They are mostly based on the fact that the value of the considered function modulo some prime numbers p can be computed easily in threshold circuits of depth 2. In this paper, we investigate the complexity of computing MP_n modulo m by depth-2 threshold circuits. It turns out that for all but a few integers m, exponential size is required. In particular, it is shown that for $m \in \{2, 4, 8\}$, polynomial-size circuits exist, for $m \in \{3, 6, 12, 24\}$, the question remains open and in all other cases, exponential-size circuits are required. The result still holds if we allow m to grow with n.

1. INTRODUCTION

In the last few years, threshold circuits of constant depth have been studied intensively. Although a threshold gate is a rather simple device which can only decide whether the number of 1's in its input is above some threshold, it seems rather difficult to prove any superpolynomial lower bound even for circuits with depth bounded by 3. The first exponential lower bound for threshold circuits of depth 2 is by [1] for the "Inner Product modulo 2," defined by $IP_n: \{0,1\}^{2n} \to \{0,1\}$, $IP_n(x_1, y_1, ..., x_n, y_n) := x_1y_1 \oplus \cdots \oplus x_ny_n$. Providing "projection reductions" from the inner product made it possible to prove that many (more interesting) functions could not be computed in polynomial size and depth 2.

Further techniques for depth 2 were developed, but could not be extended to depth 3. The lack of negative results was then complemented by a series of results which proved threshold circuits to be surprisingly powerful. If we abbreviate by TC^0_k all Boolean functions which can be computed in threshold circuits of polynomial size and depth k, then the following complex Boolean functions are now known to be contained in TC^0_3: Sorting of n binary numbers which have length n

0890-5401/96 $18.00

each; multiplication of two binary numbers of length n; computing the nth power of an input number; computing an approximation of the division of two binary numbers of length n. If we want to add n numbers of length n, we even get away with TC_2^0. Relevant (survey) articles in this area which also provide lower bounds are, e.g., [2–4, 9–12].

One of the few exceptions among arithmetic functions where we know of a small-depth (actually, depth 4, see [11]) threshold circuit, but of no TC_3^0 circuit, is the "multiple product" which computes the binary representation of $\prod_{i=1}^{n} z_i$ for n binary input numbers z_i.

The technique which was used in realizing complex functions such as division consisted of computing the result modulo small prime numbers and then reconstructing the result via Chinese remaindering.

A notion which also turned out to be rather useful is that of 1-approximability. For a formal definition, we refer the reader to, e.g., [4]. Informally, a 1-approximable function can be computed in TC_2^0-circuits which have some special property. Namely, on any input, the number of ones which are fed into the output gate is restricted to some small range. This means that the output gate has a weak task and can be omitted if there are other gates underneath.[1]

The set of 1-approximable functions is a proper subclass of TC_2^0.

Though it may seem a random decision to consider "multiple product," it should be seen that this function is close to the boundary of what we know. It seems natural to investigate whether the decomposition via Chinese remaindering can be successful when applied to the multiple product.

The first results in this direction were obtained by Krause [5] who has shown that for all $O(\log n)$-bit numbers m which have a prime factor larger than 3, the problem of computing the multiple product modulo m is not 1-approximable. The proof showed that if the multiple product modulo m was 1-approximable, then one could construct from this a small probabilistic communication protocol which is known to not exist. This was a first hint that the usual approach via Chinese remaindering might not be useful for the multiple product.

In this paper, we improve upon this result in two respects. First, we show that the above negative statement can be strengthened to TC_2^0 instead of 1-approximability. Our proof is rather simple in that it uses projection reductions only. Second, we are able to extend the statement to numbers m which consist of $c \cdot n$ bits (for some constant c). (Note that if m is even larger than 2^{cn}, then computing the product of just two numbers of length approximately $cn/2$ modulo m means computing the product exactly. In [1], a reduction is given which shows that such a multiplication of two numbers cannot be computed in TC_2^0. Hence, considering moduli m which have at most linear length is not really a restriction.)

We then extend our investigations to numbers which have not been tackled in [5]. In particular, we are able to classify for exactly which numbers of the form $m := 2^i$ the multiple product modulo m can be computed in TC_2^0.

Considering powers of 2 is perhaps the most natural case since it corresponds to computing the actual bits in the output of the multiple product. In this respect, we

[1] Note that we draw circuits with the output at the bottom and the inputs at the top.

are able to design TC^0_2-circuits which compute the 3 least significant bits of the multiple product. The way the circuits are designed also reveals that those 3 bits are actually 1-approximable. We are also able to show that higher order bits are not computable in TC^0_2. Finally, the negative results are extended to all moduli m which are divisible by 16 or 9.

2. DEFINITIONS AND BASIC PROPERTIES

Let us recall some basic number theoretic notions. For a natural number m, let $\mathbb{Z}_m$ be the residue class modulo m, and let $\mathbb{Z}^*_m$ denote the multiplicative group modulo m. For $a \in \mathbb{Z}^*_m$, we denote—slightly abusing notation—by $1/a$ the multiplicative inverse of a. The modulus will be clear from the context.

Let $\mathrm{ord}_m(a)$ denote the order of a, i.e., the smallest $i \geqslant 1$ such that $a^i \equiv 1 \bmod m$. An element $a \in \mathbb{Z}^*_m$ is called a "primitive root" modulo m if $\mathrm{ord}_m(a) = |\mathbb{Z}^*_m|$.[2] Throughout this paper, we will assume that $z_1, ..., z_n$ are binary input numbers of length n each. If the number of factors is not equal to their length, then we implicitly pad with dummy inputs. We identify the z_i with the natural numbers they represent and denote by $MP^{(m)}_n(z_1, ..., z_n)$ the binary representation of the value $\prod_{i=1}^n z_i \bmod m$. A *projection reduction* from a Boolean function $f(x_1, ..., x_n)$ to a function $g(y_1, ..., y_t)$ is a mapping $p: \{1, ..., t\} \to \{0, 1, x_1, ..., x_n, \overline{x_1}, ..., \overline{x_n}\}$ such that $f(x_1, ..., x_n) = g(p(1), ..., p(t))$. A projection reduction is called *polynomial* if t is bounded by a polynomial in n. As an example, consider the projection reduction sketched in Fig. 1. Multiplying, modulo 16, the three given binary numbers (corresponding to the rows) means computing the four least significant bits of $(4y+1)(10x+1)(12y+2x+1)$. It turns out that in the result, bit 3 is equal to the AND of x and y. I.e., Fig. 1 describes a projection reduction from the AND-function to the Multiple-Product modulo 16.

Projection reductions are "depth-preserving", i.e., if there is a polynomial projection reduction from f to g, and f needs exponential size in threshold circuits of depth 2, then so does g. The same holds if we have a projection reduction from $\bar{f}$ (the complement of f) to g.

In this paper, we provide polynomial projection reductions from IP_n to $MP^{(m)}$ showing that computing multiple product modulo m is difficult in depth-2 threshold circuits.

As in Fig. 1, projection reductions to the multiple product can be described as the product of some linear terms. This motivates the following definition where we consider linear terms which contain at most two variables.

DEFINITION 1. Let $m \geqslant 2$ be an integer. Call a polynomial of the form $ax + by + c$, where $a, b, c \in \mathbb{Z}_m$, a *linear combination*. A polynomial $f(x, y)$ which is the product of linear combinations is called a *PLC*.

In the rest of this paper, we will construct PLCs which have certain properties. We want to use them to construct projection reductions, so we have to transform

[2] By a theorem of Gauss, it is known that a primitive root exists if and only if $m = 1, 2, 4, p^\alpha, 2p^\alpha$, where p is an odd prime and $\alpha \geqslant 1$. (See, e.g., [8]).

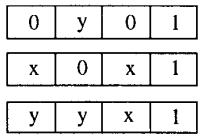

FIGURE 1

them into rows. Unfortunately, for even moduli m, this is not always possible. For example, the linear combination $2x+2y+1$ cannot be turned into a row modulo 16, since the values of all bit positions larger than 3 are equivalent to 0 modulo 16, so we have the bit with value 2 only once at our disposal. Nevertheless, we need it twice to represent $2x$ and $2y$. (Using negations of variables also will not help.) This motivates the following definition:

DEFINITION 2. (a) Three binary numbers a^*, b^*, and c^* are said to be *collision-free* if for every i, at most one of the bits a_i^*, b_i^*, or c_i^* is 1.

(b) A linear combination $ax+by+c$ *can be represented modulo* m if there are collision-free numbers a^*, b^*, and c^* such that $a \equiv a^*$, $b \equiv b^*$, and $c \equiv c^*$, modulo m. The maximum length of a^*, b^*, and c^* is called the *representation size*.

(c) A PLC *can be represented modulo* m if all of its linear combinations can be represented modulo m.

If a, b, and c are collision-free, then we can transform a PLC $ax+by+c$ into a row by putting the variable x into all positions where $a_i = 1$, the variable y into all positions where $b_i = 1$, etc.

For odd moduli, small representations of the linear combinations are easy to obtain, as the following lemma reveals.

LEMMA 3. *If m is odd, then any linear combination $ax+by+c$ can be represented modulo m, using $O(\log m)$ bits.*

Proof. Let $j := \lceil \log m \rceil$. Since m is odd, the inverses of powers of 2 exist. We binary encode the numbers a, $b/(2^j)$, and $c/(2^{2j})$ with j bits each. We then plug the encoding of these numbers into the bit positions $0, ..., j-1, j, ..., 2j-1, 2j, ..., 3j-1$, respectively. The representation size is $O(\log m)$. ∎

We now show how we can deal with the cases where the modulus m is even, assuming that we can handle the powers of 2:

LEMMA 4. *If the linear combination $ax+by+c$ can be represented modulo 2^k, then it can be represented modulo $m := 2^k \cdot r$, for all odd r, with $O(\log m)$ bits.*

Proof. We take the representation of $ax+by+c$ modulo 2^k (which occupies the least significant k bits only). Let a^*, b^*, and c^* be the corresponding collision-free numbers. We then represent

$$\frac{(a-a^*)}{2^k}x + \frac{(b-b^*)}{2^k}y + \frac{(c-c^*)}{2^k} \text{ modulo } r,$$

using the collision-free numbers a', b', and c'. This is possible by Lemma 3. It is easy to see that the three numbers $2^k a' + a^*$, $2^k b' + b^*$, and $2^k c' + c^*$ are collision-free and that $2^k a' + a^* \equiv a \bmod 2^k$ and $2^k a' + a^* \equiv a \bmod r$, which by the Chinese Remainder Theorem means that $2^k a' + a^* \equiv a \bmod(2^k \cdot r)$. Analogous statements hold, of course, for b and c. The representation size is $k + O(\log r) = O(\log m)$. ▮

We want to use the PLCs to simulate the behaviour of the inner product function. If we feed the variable pairs one by one into the inner product function, then it changes its output each time a pair has the value (1, 1). We want to achieve similar behaviour with PLCs, hence the following definition is motivated:

DEFINITION 5. We call a PLC f a *2-PLC modulo m* if it has the following property: $f(0, 0) \equiv f(0, 1) \equiv f(1, 0) \equiv 1 \bmod m$, and $\mathrm{ord}_m(f(1, 1)) = 2$.

Analogously, a PLC f is called a *3-PLC* if $f(1, 1)$ has order 3 instead of 2. (Note that 3-PLCs will be useful in dealing with the Inner Product modulo 3 instead of the Inner Product modulo 2.)

For example, $f(x, y) := (3x + 3y + 1)^2$ is a 2-PLC modulo 5 since $f(0, 0) = 1$, $f(0, 1) = f(1, 0) = 16$ are equivalent to 1 modulo 5 and $f(1, 1) = 49$ is equivalent to -1 modulo 5, which is an element of order 2.

We have reduced the problem of finding a projection reduction to the problem of finding a 2-PLC f modulo m which is representable (modulo m). The reason is the following: For every pair of variables (x_i, y_i), we take the PLC $f(x_i, y_i)$ and consider the linear terms as rows of the multiple product.

The rows which correspond to a variable pair $(x_i, y_i) \neq (1, 1)$ only contribute a factor of 1 modulo m. Let t be the number of variable pairs $(x_i, y_i) = (1, 1)$. The output of $MP^{(m)}$ is $f(1, 1)^t$. Since the order of $f(1, 1)$ is 2, this is equal to $f(1, 1)$ if t is odd and 1 if t is even. As a consequence, there is one bit in the output of $MP^{(m)}$ which is 1 iff t is odd. Hence, this bit is identical to $IP_n(x_1, y_1, ..., x_n, y_n)$.

The 2-PLCs we construct consist of 3 linear combinations which are representable with at most $c' \log m$ bits. This has the following consequence: Given n rows consisting of n bits each, we can store the PLCs of $n/3$ variables in those rows if $c' \log m \leqslant n$; hence for all $m \leqslant 2^{cn}$, the PLCs can be used to give a polynomial projection reduction from $IP_{n/3}$ to $MP_n^{(m)}$. This then yields that $MP_n^{(m)}$ needs exponential size in threshold circuits of depth 2.

Again, we note that as sketched in Section 1, the condition $m \leqslant 2^{cn}$ is not really a restriction.

We can now concentrate our attention on finding appropriate 2-PLCs.

3. NUMBERS CONTAINING A PRIME FACTOR LARGER THAN 3

In this section, we give 2-PLCs for every number m which contains a prime larger than 3 in its prime factorization.

LEMMA 6. *The following PLC f is a 2-PLC modulo m for every* $m \geqslant 5$ *which is neither divisible by* 2 *nor* 3: $(-2x + 1 - y/3) \cdot (-2x + 3y + 1) \cdot (1 - (5/8)\, y)$.

Proof. Since m is not divisible by 2 or 3, the existence of $\frac{1}{8}$ and $\frac{1}{3}$ is guaranteed. We have $f(x, 0) = (-2x+1)^2 = 1$ for $x \in \{0, 1\}$. Furthermore, $f(0, 1) = (1-\frac{1}{3}) \cdot 4 \cdot (\frac{3}{8}) \equiv 1 \bmod m$, and $f(1, 1) = (-1-\frac{1}{3}) \cdot 2 \cdot (\frac{3}{8}) \equiv -1 \bmod m$. Since $\mathrm{ord}_m(-1) = 2$, we have shown that f is a 2-PLC modulo m. ∎

Lemma 6 leaves open the question what we can show for numbers which are divisible by 2 or 3. The next lemma provides us with a method of obtaining 2-PLCs and 3-PLCs for numbers m which contain at least one other prime besides 2 and 3 in their factorization.

LEMMA 7. *Let $m_1, m_2 \geqslant 2$ be relatively prime. Assume that f is a 2-PLC modulo m_1. Let $f = f_1 \cdots f_k$ be the factorization of f into its linear combinations. There are two numbers r_1 and r_2 which depend only on m_1 and m_2 such that $f' := \prod_{i=1}^{k} (r_1 \cdot f_i + r_2)$ is a 2-PLC modulo $(m_1 \cdot m_2)$.*

The same holds with 2-PLC replaced both times by 3-PLC.

Proof. By the Chinese Remainder Theorem, we can choose two numbers r_1 and r_2 such that $r_1 \equiv 1 \bmod m_1$, $r_1 \equiv 0 \bmod m_2$, and $r_2 \equiv 0 \bmod m_1$, $r_2 \equiv 1 \bmod m_2$. This yields that $f'(x, y) \equiv f(x, y) \bmod m_1$ and $f'(x, y) \equiv 1 \bmod m_2$. We apply $\mathrm{ord}_{m_1 \cdot m_2}(a) = \mathrm{lcm}(\mathrm{ord}_{m_1}(a), \mathrm{ord}_{m_2}(a))$ to $a := f'(x, y)$ and find that since $\mathrm{ord}_{m_2}(a) = 1$, $f'(x, y)$ has the same order modulo $(m_1 \cdot m_2)$ as $f(x, y)$ has modulo m_1. Hence, f' is a 2-PLC modulo $(m_1 \cdot m_2)$ if f is a 2-PLC modulo m_1. The same arguments can be applied in the case of 3-PLCs. ∎

It should be noted that we need not know a 2-PLC modulo m_2 in the above lemma; hence it can also be applied if, e.g., $m_2 = 3$.

We are allowed to apply Lemma 7 to construct 2-PLCs in the case when one of m_1, m_2 is even. But, in order to get a projection reduction from these 2-PLCs, we also have to ensure that the PLCs can be represented. We now show that 2-PLCs constructed according to Lemma 7 have this property if we apply it carefully.

Assume therefore that we apply Lemma 7 with $m_1 = 2^i$, m_2 odd, and a PLC f modulo m_1 which is representable modulo m_1. A linear combination $ax + by + c$ within this PLC is turned by the technique of Lemma 7 into a linear combination which is equivalent to $ax + by + c$ modulo m_1. Lemma 4 shows that all linear combinations in the PLC f' can be represented modulo $m_1 m_2$.

The other case is when we apply Lemma 7 with m_1 odd and $m_2 = 2^i$. In that case, every linear combination within f' is equivalent to 1 modulo m_2 which is surely representable modulo m_2. Applying Lemma 4 again yields that f' is representable modulo $m_1 m_2$. Altogether, we have proved the following theorem:

THEOREM 8. *For every $m \geqslant 5$ which contains a prime factor larger than 3, one can construct a 2-PLC modulo m which is also representable modulo m with only $O(\log m)$ many bits. As a consequence, there is a constant c such that if $m \leqslant 2^{cn}$ has a prime factor larger than 3, then $MP_n^{(m)}$ needs exponential size when computed in threshold circuits of depth 2.*

Proof. We apply Lemma 6 to the largest number which divides m and which is not divisible by 2 or 3, and then use Lemma 7 in case m is divisible by 2 or 3. The

fact that the second statement in the theorem is a consequence of the first was already discussed in Section 2. ▮

EXAMPLE 1. Let $m_1 = 5^2$ and $m_2 = 2^2$. Modulo m_1, we have $1/3 = 17$, $1/8 = 22$. Lemma 6 yields the 2-PLC $f := (-2x + 1 - 17y)(-2x + 3y + 1)(1 - 10y)$ modulo 25. Applying Lemma 7 (with $r_1 = 76, r_2 = 25$), we find $(48x + 8y + 1)$ $(48x + 28y + 1)$ $(40y + 1)$, which is the desired (representable) 2-PLC modulo 100.

The only moduli m which remain to be investigated are of the form $2^i 3^j$. The next two sections are devoted to numbers of this form.

4. POWERS OF 2

Computing $MP_n^{(2^i)}$ corresponds to determining the bits 0, ..., $i-1$ of MP_n. This means that $MP_n^{(2^{i+1})}$ contains $MP_n^{(2^i)}$ as a subproblem. Thus, from a statement such as "$MP_n^{(2^i)}$ cannot be computed in TC_2^0," it follows immediately that $MP_n^{(2^{i+1})}$ cannot be computed in TC_2^0. Nevertheless, this would not tell us anything about whether we could compute bit i in TC_2^0 or not.

Hence, to make our statements as strong as possible, we consider in this section the functions bit_i^n which are 1 iff bit number i in MP_n is 1. (To simplify notation, we will assume n to be fixed and suppress it in the notation.)

In the following two subsections, we will prove the following theorem:

THEOREM 9.

$$bit_i \begin{cases} \text{can be computed in } TC_2^0, & \text{if } 0 \leqslant i \leqslant 2 \\ \text{can not be computed in } TC_2^0, & \text{if } 3 \leqslant i \leqslant n^2 - 4n + 3. \end{cases}$$

4.1. *Bits* 0, 1, *and* 2

Since we are dealing with the three least significant bits, we need only know the three least significant bits of the input numbers z_j, hence we can assume that $z_j = (z_{j,2}, z_{j,1}, z_{j,0})$.

The idea behind the computation is simple: If we want to compute $bit_i (i \geqslant 1)$, and one of the input numbers z_j is even, then we can replace z_j by $z_j/2$ and compute bit_{i-1} of the resulting product.

On the other hand, if we know that all input numbers are odd, then this knowledge can also be exploited appropriately. The following definitions will thus be useful.

DEFINITION 10.

$$test_i(z_1, ..., z_n) := \begin{cases} z_{1,0} \wedge \cdots \wedge z_{i-1,0} \wedge \overline{z_{i,0}}, & \text{if } 1 \leqslant i \leqslant n \\ z_{1,0} \wedge \cdots \wedge z_{n,0}, & \text{if } i = n+1. \end{cases}$$

$$shift_i(z_1, ..., z_n) := (z_1, ..., z_{i-1}, z_i/2, z_{i+1}, ..., z_n)$$

$$\text{for } 1 \leqslant i \leqslant n \quad \text{and} \quad z_i \quad \text{even.}$$

$test_i (1 \leqslant i \leqslant n)$ is 1 if z_i is the first even number and $test_{n+1}$ is 1 if all input numbers are odd. Note that exactly one of these test functions computes a 1. Note also that the division by 2 in the definition of $shift_i$ is equivalent to a simple renumbering of the bits of z_i.

LEMMA 11. *$bit_0(z_1, ..., z_n) = test_{n+1}(z_1, ..., z_n)$. For $k \in \{1, 2\}$, we have*

$$bit_k(z_1, ..., z_n) = test_{n+1}(z_1, ..., z_n) \wedge (z_{1,k} \oplus \cdots \oplus z_{n,k})$$

$$\vee \bigvee_{j=1}^{n} test_j(z_1, ..., z_n) \wedge bit_{k-1}(shift_j(z_1, ..., z_n)).$$

Proof. For bit_0, the statement of the lemma is trivial since the product can only be odd if all factors are odd. For bit_k, $k \in \{1, 2\}$, we obtain the correctness by observing the following: Either, one of the input numbers z_j is even (which is tested by $test_j$), then the corresponding term in the "big OR" computes the correct value (see the remarks before the lemma).

Now consider the remaining case when all input numbers z_j are odd. It is straightforward to verify that if we multiply $z_i = (z_{i,2}, z_{i,1}, 1)$ by $z_{i'} = (z_{i',2}, z_{i',1}, 1)$, then the three least significant bits in the product are $(z_{i,2} \oplus z_{i',2}, z_{i,1} \oplus z_{i',1}, 1)$. Inductively, we then obtain $bit_k(z_1, ..., z_n) = z_{1,k} \oplus \cdots \oplus z_{n,k}$ for $k \in \{1, 2\}$. This completes the proof. ∎

It remains to show that the formulas in Lemma 11 can be realized in threshold circuits of depth 2 and polynomial size.

LEMMA 12. *If $i \leqslant 2$, then bit_i can be computed in TC^0_2.*

Proof. We first show that the formulas in Lemma 11 can be computed in circuits of the form shown in Fig. 2, with polynomially many gates.

In words: The output OR-gate gets some AND-gates as inputs. The AND-gates get either only literals as inputs or literals plus exactly one parity-gate which also only gets literals as inputs. A literal is a variable or the negation of a variable. The formulas for bit_0 and bit_1 lead directly to circuits of the above form. In order to obtain a circuit for bit_k, $k = 2$, we observe that every term $test_j(z_1, ..., z_n) \wedge bit_{k-1}(\cdots)$ can be computed by simply feeding the literals occurring in the test function into the AND-gates of the circuit for bit_{k-1}. The "big-OR" can be melted together with the OR-gate of the circuit for bit_{k-1}.

We now sketch how such a circuit can be transformed into a TC^0_2-circuit. Consider an AND-gate which has a parity function and literals as input. Using a known trick called "wire-encoding" (see, e.g., [4]), it can be computed by a "symmetric" gate. Let us sketch this briefly: Let the literals entering the parity gate be $v_1, ..., v_t$

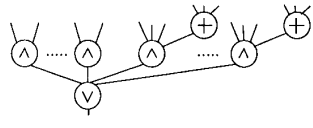

FIGURE 2

and let the other literals be $w_1, ..., w_T$. Then the output of the AND-gate only depends on the value of $(T+1)\sum_{i=1}^{t} v_i + \sum_{i=1}^{T} w_i$. Such "symmetric" functions are easy to realize in depth-2 threshold circuits; furthermore, they are 1-approximable.

This means that the output OR-gate can be seen as a gate which gets 1-approximable functions as input. Hence, the whole circuit can be simulated in polynomial-size, depth-2 threshold circuits.

(We only remark that due to the choice of our test functions, the above circuit additionally shows that bit_0, bit_1 and bit_2 are not only in TC_2^0, but also 1-approximable. Namely, at most one input of the OR-gate is 1. Therefore, the output can be regarded as exactly representable by 1-approximable functions. As a consequence, the function computed by the whole circuit is also 1-approximable.)

4.2. *Bits* 3, 4, 5, ...

LEMMA 13. *Let* $i \geqslant 4$ *and* $a := 2^{i-2}$. *The following PLC* f *is a* 2*-PLC modulo* 2^i *which is also representable modulo* 2^i:

$$f(x, y) = (ay+1) \cdot (3ay + (a-2)\,x + 1) \cdot ((3a-2)\,x + 1).$$

f *also has the property that* $f(1, 1) \equiv (2^{i-1}+1) \bmod 2^i$.

Proof. For $i \geqslant 4$, it holds that $a^2 = 2^{2i-4} \equiv 0 \bmod 2^i$. We conclude $f(0, y) = (ay+1) \cdot (3ay+1) = 3a^2y^2 + 4ay + 1 \equiv 1 \bmod 2^i$ and $f(1, 0) = (a-1) \cdot (3a-1) = 3a^2 - 4a + 1 \equiv 1 \bmod 2^i$. And, $f(1, 1) = (a+1) \cdot (4a-1) \cdot (3a-1) \equiv -(3a^2 + 2a - 1) \equiv 1 - 2a \equiv 2^i + 1 - 2^{i-1} \equiv 2^{i-1} + 1 \equiv 2a + 1 \bmod 2^i$. Since $(2a+1)^2 = 4a^2 + 4a + 1 \equiv 1 \bmod 2^i$, we have that $f(1, 1)$ is an element of order 2. Hence, f is a 2-PLC modulo 2^i.

Every linear combination within f is representable modulo 2^i: For the first and third linear combination, this is obvious since a and $3a-2$ are even numbers, for the second linear combination it suffices to see that $3a = 2a + a = 2^{i-1} + 2^{i-2}$, and $a - 2 = 2^{i-3} + \cdots + 2^1$. ∎

Applying Lemma 13 with $i = 4$, we obtain a 2-PLC modulo 16. Fig. 1 shows the projection reduction which corresponds to this 2-PLC.

Looking at the way we construct projection reductions from PLCs, we find that bit 3 in the multiplication of n binary numbers of length 4 is equal to $IP_{n/3}$; hence it is not in TC_2^0.

In order to generalize this result to the other bits, we simply observe that by multiplication of one of the n input numbers in the reduction with 2, the hard bit in the multiple product moves "one to the left." Padding all input numbers to length n in such a way, we obtain that all bits from position 3 to position $3 + (n-4) \cdot n = n^2 - 4n + 3$ cannot be computed in TC_2^0. One could even extend this result to higher order bits by observing that already a reduction from $IP_{\log^2 n}$ would prove noncontainment in TC_2^0, but we leave these extensions to the reader.

The following should be noted: There is a reduction in [1] which shows that the multiplication of two numbers cannot be computed in TC_2^0. More precisely, it is approximately the middle bit in the multiplication of 2 numbers of length $O(n \log n)$

which is equal to the Inner Product modulo 2. This reduction can of course be used to give exponential lower bounds if i grows with n. The strength of our reduction is that it already works for very small i.

5. POWERS OF 3

We start by showing that the situation for computing the multiple product modulo 3 is different. Namely, we show that there is no projection reduction from the Inner Product modulo 2 to $MP_n^{(3)}$. This means that the reduction technique which is behind Theorem 8 has to fail when applied to $MP^{(3)}$.

THEOREM 14. *For all n, the following holds*:

There is no projection reduction from IP_5 *to* $MP_n^{(3)}$.

There is no projection reduction from $\overline{IP_5}$ *to* $MP_n^{(3)}$.

Proof. In order to get a contradiction, let us assume that we can find a projection reduction. Again, we visualize the factors as rows where in the bit positions we have variables or constants. (Negations of variables can easily be simulated since $-1 = 2 \bmod 3$.) Let $V := \{x_1, y_1, ..., x_5, y_5\}$ be the set of binary variables. We recall that a row corresponds to an expression of the form $a_1x_1 + b_1y_1 + \cdots + a_5x_5 + b_5y_5 + c$, where all a_i, b_i, and c are taken from the set $\{0, 1, 2\}$.

We consider first in which situations we can force the value of a row to 0 (modulo 3) by some variable assignment. Let the row be given by the above expression. If $c = 0$, we can set all variables to 0 and force the value of the row to 0. If $c = 1$, and there is one variable v where the coefficient of v is 2, then we set $v := 1$, and all other variables to 0. If $c = 1$, and there are at least two variables v_1 and v_2 which have the coefficient 1, then we set $v_1 := v_2 := 1$, and all other variables to 0. The case $c = 2$ can be treated analogously. Thus, the only rows which cannot be forced to 0 are of the form 1, 2, $(v + 1)$, or $2(v + 1)$ for some variable v.

We now return to the projection reduction. Assume that no row in this projection reduction can be forced to 0 by some variable assignment. By the above arguments, the projection reduction then corresponds to a PLC f in which every linear combination is of the form 1, 2, $(v + 1)$, or $2(v + 1)$ for some variable v.

f has to depend essentially on all variables since IP_5 depends essentially on all variables. Furthermore, we have $(v + 1)^2 \equiv 1 \bmod 3$ for $v \in \{0, 1\}$. This means that for some $i \in \{0, 1\}$, f is equivalent to $2^i(x_1 + 1)(y_1 + 1) \cdots (x_5 + 1)(y_5 + 1)$. Then we have $f(0, 0, 0, ..., 0) = f(1, 1, 0, ..., 0) \bmod 3$, but $IP_5(0, 0, 0, ..., 0) \neq IP_5(1, 1, 0, ..., 0)$. Thus, f cannot correspond to a projection reduction which yields a contradiction. This contradiction was caused by the assumption that there is no row in the projection reduction which can be forced to 0.

We now investigate a row which can be forced to 0 and the corresponding variable assignment more closely. Let the value of IP_5 on this assignment be s. The value of the multiple product modulo 3 on this assignment is of course 0.

For every pair (x_i, y_i), it holds that changing it either to $(0, 0)$ or to $(1, 1)$ will change the output of IP_5. Hence, the value of the row under consideration also needs to change since otherwise the multiple product would remain 0.

For all pairs (x_i, y_i), we mark by which amount the value of the row will change. Let us call this value d_i. We have just seen that $d_i \in \{1, 2\}$.

We have 5 variable pairs; hence, by the pigeonhole principle, there are at least three d_i which are equal. Let us assume w.l.o.g. that $d_1 = d_2 = d_3$.

By changing the assignments of these pairs, the output of IP_5 will change 3 times and then is equal to $s \oplus 1$. On the other hand, the value of the row will be changed by an amount of $3d_1 \equiv 0 \bmod 3$; hence the multiple product is still zero. This is a contradiction. ▮

Three things should be noted: First, the proof of Theorem 14 shows more than that there are no 2-PLCs modulo 3 since in general there might be projection reductions which work in a different manner. This is why we have to deal with zero rows in the above proof. Second, it can easily be seen that computing whether the multiple product is divisible by 3 is in TC_2^0. Nevertheless, this does not tell us anything about how the bits in the representation can be computed. Third, it is easy to see that 2-PLCs modulo 3^i are also 2-PLCs modulo 3; hence, there are none.

Altogether, we have an indication that powers of 3 have to be treated differently. One way out is to consider the "inner product modulo 3," IP^*, defined by $IP^* = x_1 y_1 + \cdots + x_n y_n \bmod 3$. In order to turn IP^* into a Boolean function $f^{(IP3)}$, we have to find some appropriate encoding. We choose $f^{(IP3)} = 1 \Leftrightarrow IP^* = 0$. It has been shown in [7] that if we have a threshold circuit of depth 2 which gets binary coded values from $\{0, 1, 2\}$ as input and which has to compute some binary encoding of IP^* (over $\mathbb{Z}_3$), then this circuit needs exponential size. This result can be used to show that the Boolean function $f^{(IP3)}$ cannot be computed in TC_2^0 [6].

This suggests that one should try to find 3-PLCs modulo 3^i. First, let us show how we can exploit 3-PLCs to obtain projection reductions:

LEMMA 15. *Given a 3-PLC f modulo 3^i, we can construct a polynomial projection reduction from either $f^{(IP3)}$ or $\overline{f^{(IP3)}}$ to $MP^{(3^i)}$.*

Proof. We construct the reduction similar to the remarks after Definition 5. Depending on the value $f(1, 1)$, we possibly add an extra row. Whether it is a reduction from $f^{(IP3)}$ or from $\overline{f^{(IP3)}}$ also depends on the value $f(1, 1)$.

Let $a = f(1, 1)$ be the element of order 3. As the output of $MP^{(3^i)}$, we have either $a^0 = 1$, $a^1 = a$, or a^2, depending on whether the number of $(1, 1)$-pairs in the input is equivalent to 0, 1, or 2 modulo 3. In the binary representations of 1, a, and a^2, there is one bit position j which is not the same for all three numbers. Assume that the jth bit of a^k, $k \in \{0, 1, 2\}$, is 1 and that the jth bits of the other two numbers are 0. (The other case is similar; we then consider $\overline{f^{(IP3)}}$.)

We add an extra row which contains a factor a^k. The output of $MP^{(3^i)}$ is then a^k iff the number of pairs $(1, 1)$ is divisible by 3.

Hence, the jth bit in the output of $MP^{(3^i)}$ is equal to $f^{(IP3)}$. ▮

Let us now present the 3-PLCs that we have found. We need the following number theoretic proposition (see, e.g, [8, p. 102]).

PROPOSITION 16. *If p is an odd prime and g is a primitive root modulo p^2, then g is a primitive root modulo p^α for $\alpha = 3, 4, 5, \ldots$. In particular, 2 is a primitive root modulo 3^i for all $i \geqslant 2$.*

LEMMA 17. *For $i \geqslant 2$, let $w = 2^{2 \cdot 3^{i-2}}$, $b = (w+1)/2 - 1$, $a = 2/(w+1) - 1$. Then the PLC $f := (-2x+1) \cdot (-2x + ay + 1) \cdot (by + 1)$ is a 3-PLC modulo 3^i.*

Proof. We first have to ensure that the inverses modulo 3^i used in the definition of the parameters do exist. For $\frac{1}{2}$, this is clear. w is a power of 4, hence $w \equiv 1 \bmod 3$, and $w + 1 \equiv 2 \bmod 3$ has an inverse. Now, we evaluate f: $f(x, 0) = (-2x+1)^2 = 1$, $f(0, 1) = (a+1)(b+1) \equiv 1 \bmod 3^i$, and $f(1, 1) = (1-a)(b+1) \equiv w \bmod 3^i$.

Two is a primitive root modulo 3^i (see Proposition 16), hence it has order $2 \cdot 3^{i-1}$. Thus, $f(1, 1) \equiv w$ has order 3. ∎

For example, $(-2x+1) \cdot (-2x+3y+1) \cdot (6y+1)$ is a 3-PLC modulo 9.

Lemma 17 together with Lemma 15 shows that there is a bit in the binary representation of $MP^{(3^i)}$, $i \geqslant 2$ fixed, which cannot be computed in TC^0_2.

Lemma 7 can also be used to obtain 3-PLCs for numbers m which are of the form $3^i \cdot r$, $i \geqslant 2$.

This yields that the only moduli that we were not able to classify are of the form $3 \cdot 2^i$, $i \leqslant 3$, since to all other numbers of the form $2^i 3^j$, one of the PLC construction methods can be applied.

6. FINAL REMARKS

We were able to classify, for all positive integers $m \notin \{3, 6, 12, 24\}$, whether the multiple product modulo m can be computed in polynomial-size threshold circuits of depth 2. For $m = 24$, we have the strange situation that we are able to provide 2-PLCs modulo 24, but, we have not found a 2-PLC which is also representable.

One example of such a PLC is $(-2x + 12y + 1)(-2x + 18y + 1)(6y + 1)$.

ACKNOWLEDGMENTS

Our thanks go to Petr Savický for suggesting to us an improvement in Theorem 14, to Hanno Lefmann for pointing us to Proposition 16, and to Matthias Krause for stimulating discussions, in particular for suggesting the investigation of 3-PLCs instead of 2-PLCs for computing modulo 9. We also thank the anonymous referees for their useful comments.

Received July 28, 1995; final manuscript received October 9, 1996

REFERENCES

1. Hajnal, A., Maass, W., Pudlák, P., Szegedy, M., and Turán, G. (1993), Threshold circuits of bounded depth, *J. Comput. System Sci.* **46**, 129–154.
2. Goldmann, M., Håstad, J., and Razborov, A. (1994), Majority gates vs. general weighted threshold gates, *Comput. Complexity* **2**, 277–300.
3. Goldmann M., and Karpinski, M. (1993), Simulating threshold circuits by majority circuits, *in* "Proceedings, 25th STOC," pp. 551–560.

4. Hofmeister, T. (1994), Depth-efficient threshold circuits for arithmetic functions, *in* "Theoretical Advances in Neural Computation and Learning" (V. Roychowdhury, K-Y. Siu, and A. Orlitsky, Eds.), Chap. 2, Kluwer Academic, Dordrecht/Norwell, MA.
5. Krause, M. (1995), On realizing iterated multiplication by small depth threshold circuits, *in* "Proceedings of 12th STACS," pp. 83–94.
6. Krause, M., personal communication.
7. Krause, M., and Waack, S. (1995), Variation ranks of communication matrices and lower bounds for depth two circuits having symmetric gates with unbounded fan-in, *Math. Systems Theory* **28**, 553–564, see also "Proceedings of 32nd FOCS, 1991," pp. 777–782.
8. Niven, I., Zuckerman, H. S., and Montgomery, H. L. (1991), "An Introduction to the Theory of Numbers," 5th ed., Wiley, New York.
9. Razborov, A. (1992), On small depth threshold circuits, *in* "Proceedings, 3rd Scandinavian Workshop on Algorithm Theory," Lecture Notes in Computer Science, Vol. 621, pp. 42–52, Springer-Verlag, Berlin/New York.
10. Siu, K.-Y., Bruck, J., Kailath, T., and Hofmeister, T. (1993), Depth efficient neural networks for division and related problems, *IEEE Trans. Inform. Theory*, 946–956.
11. Siu, K.-Y., and Roychowdhury, V. (1994), On optimal depth threshold circuits for multiplication and related problems, *SIAM J. Discrete Math.* **7**, 285–292.
12. Wegener, I. (1993), Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions, *Inform. Process. Lett.* **46**, 85–87.

Printed in Belgium