Multiple Product modulo arbitrary numbers

Claudia Bertram and Thomas Hofmeister
Lehrstuhl Informatik II
Universität Dortmund
D-44221 Dortmund, Germany
Email: { bertram, hofmeist } @ls2.informatik.uni-dortmund.de

Abstract

Let n binary numbers $z_1, \ldots, z_n$ 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 in [SR],[SBKH] that this function can be computed in polynomial-size threshold circuits of depth 4. For a lot of 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.
In this paper, we show that the difficulty in constructing smaller depth circuits for MP_n stems from the fact that for all numbers m which are divisible by a prime larger than 3, computing MP_n modulo m already cannot be computed in depth 2 and polynomial size. This result still holds if we allow m to grow exponentially in n ($m < 2^{cn}$, for some constant c). This improves upon recent results in [K1].
We also investigate moduli which are of the form $2^i 3^j$. Especially, we show that there are polynomial-size threshold circuits for computing MP_n modulo $2, 4$, and 8, but that there are no such circuits for computing modulo m if m is divisible by 16. We also sketch that moduli which are divisible by 9 lead to exponential size.

1 Introduction

In the last few years, threshold circuits of constant depth have been intensively studied. Although a threshold gate is a rather simple device which can only decide whether the number of 1s in its input is above some threshold, it turned out that it seems rather difficult to prove any non-polynomial lower bound even for circuits with depth bounded by 3. The first exponential lower bound for circuits of depth 2 is by [HMPST]. The result proved in that paper is the now well-known fact that the "Inner Product modulo 2", defined by $IP_n : \{0,1\}^{2n} \to \{0,1\}$,

$$IP_n(x_1, y_1, \ldots, x_n, y_n) := x_1 y_1 \oplus \cdots \oplus x_n y_n$$

needs exponentially many gates in threshold circuits of depth 2. Different 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 each; multiplication of two binary numbers of length n; computing the $n-$th 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^0_2 ([GK, GHR]). For a survey or for lower bounds, see e.g. [H, R, W].
One of the few exceptions of arithmetic functions where we know of a small-depth (actually, depth 4, see [SR]) threshold circuit, but of no TC^0_3 circuit is the "multiple product".

The technique which was used when realizing complex functions like 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. We will not give a formal definition of this since we make no essential use of it. Informally, a 1-approximable function can be computed in TC^0_2-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. The set of 1-approximable functions is a proper subclass of TC^0_2.
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 why the decomposition via Chinese Remaindering fails when applied to the multiple product.
First results obtained in this direction were given in Krause [K1] 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 of this was based on communication complexity arguments. This indicates already why Chinese Remaindering cannot be successful.
In this paper (section 3), we improve upon this result in two respects. First, we show that the above negative statement can be strengthened to TC^0_2 instead of 1-approximability. Secondly, we are able to extend the statement to numbers m which consist of $c \cdot n$ bits (for some constant c). The proof that we give is also rather simple.

We then extend our investigations to numbers which have not been tackled in [K1]. In particular, we are able to classify exactly for which numbers of the form $m := 2^i$ the multiple product modulo m can be computed in TC^0_2.
It should be noted that considering prime 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 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 then able to show that higher order bits are not computable in TC^0_2.
We finally extend the negative results 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 number m, let Z_m be the residue class modulo m, and Z^*_m denote the multiplicative group modulo m. For an $a \in Z^*_m$, we denote – slightly abusing notation – by $\frac{1}{a}$ 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 $i \geq 1$ such that $a^i \equiv 1 \bmod m$. An element $a \in Z^*_m$ is called a "primitive root" modulo m if $\mathrm{ord}_m(a) = |Z^*_m|$.
Throughout this paper, we will assume that $z_1, \ldots, z_n$ 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 that they represent and denote by $MP^{(m)}_n(z_1, \ldots, z_n)$ the binary representation of the value $z_1 \cdots z_n \bmod m$.

A *projection reduction* from a function $f(x_1, \ldots, x_n)$ to a function $g(y_1, \ldots, y_t)$ is a mapping

$$p : \{1, \ldots, t\} \to \{0, 1, x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$$

such that $f(x_1, \ldots, x_n) = g(p(1), \ldots, p(t))$. This means that we can set variables of g to constants or identify them with variables of f in such a way that we obtain the function f. A projection reduction is called *polynomial* if t is bounded by a polynomial in n.

We will visualize projection reductions to "Multiple Product" as is sketched in figure

1. Input numbers are regarded as rows.

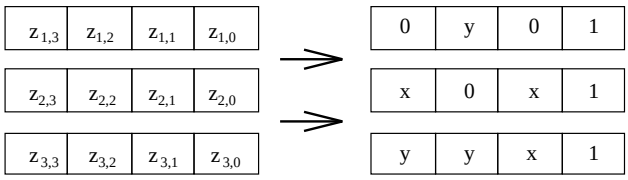

Figure 1

If we manage to show that there is a polynomial projection reduction from IP_n to some function f, this has the consequence that f also needs exponential size when computed in threshold circuits of depth 2.
It is easy to see that for the multiple product, projection reductions 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 2.1 *Let $m \geq 2$ be an integer. Call a polynomial of the form $ax + by + c$, where $a, b, c \in Z_m$, a "linear combination". A polynomial $f(x, y)$ which is the product of linear combinations is called a PLC.*

In the sequel, we will construct PLCs which have certain properties. We want to use them to construct projection reductions, so we have to transform them into rows. Unfortunately, for even moduli m, this is not always possible. For example, the PLC $2x + 2y + 1$ 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 won't help.)
This motivates the following definition:

Definition 2.2 *A linear combination $ax + by + c$ can be "represented modulo m" if a, b, and c possess binary representations modulo m*

$$a = \sum_{i=1}^{N} a_i 2^i, \quad b = \sum_{i=1}^{N} b_i 2^i, \quad c = \sum_{i=1}^{N} c_i 2^i$$

such that for all i the bits a_i, b_i, c_i "do not collide", i.e., for all i, at most one of the bits a_i, b_i, c_i is 1. The number of bits used in the representation is N. A PLC "can be represented modulo m" if all of its linear combinations can be represented.

If the bits do not collide, then we can transform a PLC $ax + by + c$ into a row by putting an x into all positions where $a_i = 1$, y into all positions where $b_i = 1$ etc.
The only moduli causing trouble are even, as the following lemma shows. It also reveals small representations of the linear combinations.

Lemma 2.3 *If m is odd, then any linear combination can be represented modulo m, using $O(\log m)$ bits.*

Proof: Let $ax + by + c$ be the linear combination we are considering. We only need to show that a, b, c can be represented in such a way that their bits "do not collide". Choose $j := \lfloor \log m \rfloor + 1$. 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 $0, \ldots, j-1$, $j, \ldots, 2j-1$, $2j, \ldots, 3j-1$, respectively. This yields a representation of the claimed size. •

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

Proof(Sketch:) Split a into $a = a' + a''$ such that $a' = a \bmod 2^j$. We split b and c similarly. Using the method from lemma 2.3, we represent a'', b'', c'' modulo r in such a way that we can plug them into bit positions which are larger than j. We get a representation of a'', b'', c'' modulo m, since we haven't changed their value modulo 2^j. Then, we plug a', b', and c' into the first j bit positions which can be done since we assumed that $ax + by + c$ could be represented modulo 2^j. The number of bits used in the representation is $j + O(\log r) = O(\log m)$. •

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 a similar behaviour with PLCs, hence the following definition is motivated:

Definition 2.5 *We call a PLC f a "2-PLC modulo m" if it has the following property:*

$$f(0,0) \equiv f(0,1) \equiv f(1,0) \equiv 1 \bmod m,$$

$$\text{and } ord_m(f(1,1)) = 2$$

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 when 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 2-PLCs modulo m which are representable. The reason is the following: For every pair of variables (x_i, y_i), we take a 2-PLC $f(x_i, y_i)$ modulo m and consider the linear terms as rows of the multiple product. This can be done since they were assumed to be representable.
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 order of $f(1,1)$ is 2, hence the output of $MP^{(m)}$ is:

$$MP^{(m)} = f(1,1)^t = \begin{cases} f(1,1), & \text{if } t \text{ is odd} \\ 1, & \text{if } t \text{ is even} \end{cases}$$

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 $IP(x_1, y_1, \ldots, x_n, y_n)$. We will construct the PLCs for every variable pair (x,y) and every modulus m such that they consist of 3 linear combinations and such that they 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 $c' \log m \leq n$, hence for all $m \leq 2^{cn}$, 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.

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 3.1 *For every number $m \geq 5$ which is neither divisible by 2 nor 3, the following PLC f is a 2-PLC modulo m:*

$$(-2x + 1 - \frac{y}{3}) \cdot (-2x + 3y + 1) \cdot (1 - \frac{5}{8}y)$$

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 $\text{ord}_m(-1) = 2$, we have shown that f is a 2-PLC modulo m. •

Lemma 3.1 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 to obtain 2-PLCs for numbers m which contain at least one other prime except 2 and 3 in their factorization.

Lemma 3.2 *Let $m_1, m_2 \geq 2$ be two numbers which are 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 only depend on m_1 and m_2 such that the following PLC f' is a 2-PLC modulo $m_1 \cdot m_2$.*

$$f' := f_1' \cdots f_k', \quad \text{where } f_i' := (r_1 \cdot f_i + r_2)$$

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 $f_i'(x,y)$ is equivalent to $f_i(x,y)$ modulo m_1 and equivalent to 1 modulo m_2. Consequently,

$$f'(x,y) \equiv \begin{cases} f(x,y) & \bmod m_1 \\ 1 & \bmod m_2 \end{cases}$$

A simple number-theoretic property states that

$$\mathrm{ord}_{m_1 \cdot m_2}(a) = \mathrm{lcm}(\mathrm{ord}_{m_1}(a), \mathrm{ord}_{m_2}(a))$$

We apply this to $a := f'(x,y)$ to find that since $\mathrm{ord}_{m_2}(a) = 1$, $f'(x,y)$ has the same order modulo $m_1 m_2$ as $f(x,y)$ has modulo m_1. Hence, f' is a 2-PLC modulo $m_1 m_2$. •

It should be noted that we do not need to know a 2-PLC modulo m_2 in the above lemma, hence it can also be applied if e.g. $m_2 = 3$.
Note further that the lemma can also be applied if we are searching for 3-PLCs instead of 2-PLCs.
We are allowed to apply lemma 3.2 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 3.2 have this property if we apply it carefully.
Assume therefore that we apply lemma 3.2 with $m_1 = 2^i$ 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 3.2 into a linear combination which is equivalent to $ax + by + c$ modulo m_1. Lemma 2.4 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 3.2 with $m_2 = 2^i$. In that case, every linear combination within f' is equivalent to 1 modulo m_2 which is surely representable modulo m_2. Applying lemma 2.4 again yields that f' is representable modulo $m_1 m_2$.

Altogether, we have proved the following theorem:

Theorem 3.3 *For every $m \geq 5$ 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 $m \leq 2^{cn}$ 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 3.1 to the largest number which divides m and which is not divisible by 2 or 3, and then use lemma 3.2 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. •

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.
Let us finish this section with an example:

Example 3.4 *Assume that we are looking for a 2-PLC modulo $300 = 2^2 \cdot 3 \cdot 5^2$. We apply lemma 3.1 with $m = 5^2$ where we have that $1/3 = 17$, $1/8 = 22$. This yields a 2-PLC modulo 25:*

$$(-2x + 1 - 17y)\,(-2x + 3y + 1)\,(1 - 10y)$$

We then apply lemma 3.2 to $m_1 = 25$ and $m_2 = 3$. We choose $r_1 = 51$, $r_2 = 25$ and get a 2-PLC modulo 75:

$$(48x - 42y + 1)(48x + 3y + 1)(15y + 1)$$

Applying lemma 3.2 again to $m_1 = 75, m_2 = 4$, (with $r_1 = 76$, $r_2 = 225$) we get the PLC

$$(48x + 108y + 1)(48x + 228y + 1)(240y + 1)$$

which is the desired 2-PLC modulo 300. This PLC is representable modulo 2^2 since it is equivalent to $1 \cdot 1 \cdot 1$ modulo 2^2. Hence, it is representable modulo 300.

4 Powers of 2

Computing $MP_n^{(2^i)}$ corresponds to splitting off the bits $0, \ldots, i-1$ of MP_n. This means that $MP_n^{(2^{i+1})}$ contains $MP_n^{(2^i)}$ as a subproblem. Thus, from a statement like "$MP_n^{(2^i)}$ cannot be computed in TC_2^0", it would follow 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 $h_n^{(2^i)}$ which are 1 iff bit number $i-1$ in MP_n is 1.
We start by showing that for $i \geq 4$, $h_n^{(2^i)}$ cannot be computed in TC_2^0 by giving appropriate 2-PLCs. We then show that for $i < 4$, $h_n^{(2^i)}$ can in fact be computed in TC_2^0.

Lemma 4.1 *Let $i \geq 4$ and $a := 2^{i-2}$. Then the following PLC f is a 2-PLC modulo 2^i which is also representable modulo 2^i.*

$$(ay+1)\cdot(3ay+(a-2)x+1)\cdot((3a-2)x+1)$$

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

Proof: For $i \geq 4$, it holds that $a^2 = 2^{2i-4} \equiv 0 \mod 2^i$. We conclude $f(0,y) = (ay+1)\cdot(3ay+1) = 3a^2y^2+4ay+1 \equiv 1 \mod 2^i$ and $f(1,0) = (a-1)\cdot(3a-1) = 3a^2-4a+1 \equiv 1 \mod 2^i$. And, $f(1,1) = (a+1)\cdot(4a-1)\cdot(3a-1) \equiv -(3a^2+2a-1) \equiv 1-2a \equiv 2^i+1-2^{i-1} \equiv 2^{i-1}+1 \equiv 2a+1 \mod 2^i$. Since $(2a+1)^2 = 4a^2+4a+1 \equiv 1 \mod 2^i$, 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 $a-2 = 2^{i-3}+\cdots+2^1$. •

Figure 1 in section 2 shows the projection reduction which corresponds to the 2-PLC modulo 16 constructed according to lemma 4.1. Looking at the way we construct projection reductions from PLCs, we find that because of $f(1,1) \equiv (2^{i-1}+1) \mod 2^i$, it is bit number $i-1$ which in the projection reduction is identical to the Inner Product modulo 2. Hence, $h_n^{(2^i)}$ cannot be computed in TC_2^0 for $i \geq 4$.
The size of the representation of the PLC from lemma 4.1 is $i-1$, hence if i is allowed to grow with n, we this time get: As long as $4 \leq i \leq n$, there is a projection reduction from $IP_{n/3}$ to $h_n^{(2^i)}$ which proves that all bits up to position $n-1$ are not computable in TC_2^0. The following should be noted: There is a reduction in [HMPST] which shows that the multiplication of two numbers cannot be computed in TC_2^0. That reduction can be used to show stronger results if i is growing in some way with n. The strength of our reduction is that it already works for very small i.

For $i \leq 3$, the situation is different. We are able to show that for those i, $h_n^{(2^i)}$ can be computed in TC_2^0.
We first prove that $h_n^{(2)}$, $h_n^{(4)}$ and $h_n^{(8)}$ can be computed in circuits of the following form with polynomially many gates:

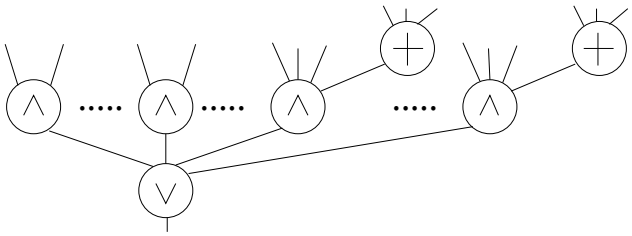

Figure 2

In words: On the output level is an OR-gate which gets some AND-gates as inputs. The AND-gates either only get 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 following proposition will be quite useful:

Proposition 4.2 *If the number x is even, then $x \mod 2^k = 2(\frac{x}{2} \mod 2^{k-1})$.*

The proposition leads to the following nice property: Assume that we want to compute $h^{(2^k)}(z_1, \ldots, z_n)$. If one of the input numbers, say z_i, is even, then the value of $h^{(2^k)}$

on this input is identical to the value of $h^{(2^{k-1})}$ on input $z_1, \ldots, z_{i-1}, z_i/2, z_{i+1}, \ldots, z_n$
To realize our idea, we define the following test functions $t_1, \ldots, t_{n+1}$ which check whether some input numbers z_i are even.

$$t_i := \begin{cases} z_{1,0} \wedge \cdots \wedge z_{i-1,0} \wedge \overline{z_{i,0}} & \text{if } i \leq n, \\ z_{1,0} \wedge \cdots \wedge z_{n,0} & \text{if } i = n+1 \end{cases}$$

Especially, (for $1 \leq i \leq n$), $t_i = 1 \Leftrightarrow z_i$ is the first even number. t_{n+1} is 1 if all z_i are odd. It should be noted that by definition, exactly one function t_i computes the value 1, the others compute the value 0.

$h_n^{(2)}$ is 1 iff all input numbers are odd, hence it is identical to t_{n+1} and can be computed in a circuit of the above form (using some dummy gates).
We now try to compute $h^{(4)}$ using $h^{(2)}$. Since $h^{(4)}$ only depends on the 2 least significant bits of the z_i, we can assume that all input numbers are smaller than 4.
Define the following functions for $1 \leq i \leq n$:

$$G_i := h^{(2)}(z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_n)$$

where z_i' is obtained from z_i by shifting the bits one position to the right and ignoring the least significant bit.
If z_i is even, then by the above remarks, G_i is identical to $h^{(4)}(z_1, \ldots, z_n)$.
Suppose that every z_i is odd. Consequently, all input numbers are either 1 or 3. The product of these numbers is either 1 or 3 modulo 4. Since $3^2 = 1 \bmod 4$, we find that $h^{(4)}$ is 1 if there is an odd number of 3's in the input. This can easily be tested by $G_{n+1} := z_{1,1} \oplus \cdots \oplus z_{n,1}$.
Putting things together, we get the following formula:

$$h^{(4)} = (t_1 \wedge G_1) \vee (t_2 \wedge G_2) \vee \ldots \vee (t_{n+1} \wedge G_{n+1})$$

For each G_i $(1 \leq i \leq n)$, we have already designed circuits of the form described in figure 2. The expression $(t_i \wedge G_i)$ can be realized by putting the literals which occur in t_i into the AND-gates of the circuit for G_i. The OR-gate can be melted together with the OR-gate of the G_i's. $G_{n+1} \wedge t_{n+1}$ contributes the AND-gate which gets the parity as input.

We proceed in a similar fashion with $h^{(8)}$. Again, we can assume that the input numbers are all smaller than 8. We will use the following lemma in which $a_i := \#\{j | z_j = i\}$ counts how many inputs have the value i.

Lemma 4.3 *If $z_1 \cdots z_n \bmod 8 \in \{1,3,5,7\}$, then $z_1 \cdots z_n \bmod 8$ is in $\{5,7\}$ if and only if $a_5 + a_7$ is odd.*

The proof is immediate from the graphical representation of the multiplication table modulo 8 in figure 3.

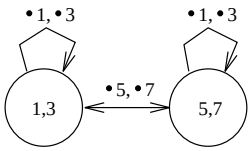

Figure 3

To construct a circuit for $h^{(8)}$, we proceed as follows.
If one of the input numbers is divisible by 2, we can compute, as we did for $h^{(4)}$, the function $t_i \wedge G_i'$, where G_i' is obtained from G_i by replacing $h^{(2)}$ by $h^{(4)}$.

We now have to consider the case where all input numbers are odd. This is the case if all input numbers are from $\{1,3,5,7\}$. Lemma 4.3 tells us that we only have to count how many input numbers are 5 or 7. As all numbers are odd, we can identify the 5's and 7's by bit number 2, namely: $z_i \in \{5,7\} \Leftrightarrow z_{i,2} = 1$. Consequently, $a_5 + a_7$ is odd if and only if

$$G_{n+1} := z_{1,2} \oplus \cdots \oplus z_{n,2} = 1.$$

We get a term $t_{n+1} \wedge G_{n+1}$. Analogously to $h^{(4)}$, we can now compute $h^{(8)}$ in a circuit of the form described in figure 2.

We now sketch how this circuit can be transformed into a TC_2^0-circuit. Consider an AND-gate which has a parity function and literals as input. Using a known trick called "wire-encoding", it can be computed by a "symmetric" gate. Let us sketch this shortly: Let the literals entering the parity gate be $v_1, \ldots, v_t$ and the other literals be $w_1, \ldots, w_T$. Then the output of the AND-gate only depends on the value of $(T+1)v_1 + \cdots + (T+1)v_t + w_1 + \cdots + w_T$. 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 and hence the whole circuit can be computed in polynomial-size, depth-2 threshold circuits.
(As a remark, it should be noted that due to the choice of our test functions, the above circuit also shows that $h^{(2)}, h^{(4)}$, and $h^{(8)}$ are 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. It can be shown (see e.g. [H]) that as a consequence, the function computed by the whole circuit is also 1-approximable.)

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 $MP_n^{(3)}$. More precisely, we will show that there is no n such that there is a projection reduction from IP_5 to $MP_n^{(3)}$. In other words: The reduction technique which is behind theorem 3.3 has to fail when applied to $MP^{(3)}$.

Theorem 5.1 *For all n, there is no projection reduction from 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 $V := \{x_1, y_1, \ldots, x_5, y_5\}$ 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. We obtain that 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, we know that 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. Hence, for every v, there has to be a term $(v + 1)$ or $2(v + 1)$ in f.
Since $x^2 \equiv 1 \bmod 3$ for all x, we know that $f = 2^i(x_1 + 1)(y_1 + 1)\cdots(x_n + 1)(y_n + 1)$ for some $i \in \{0, 1\}$. Then we have $f(0, 0, 0, \ldots, 0) = f(1, 1, 0, \ldots, 0) \bmod 3$, but the Inner Product modulo 2 yields 0 on the first, and 1 on the second input. 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, hence it is then 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. •

We remark that by a finer analysis, one could improve upon the constant 5 in theorem 5.1. Three things should be noted: First, the proof of Theorem 5.1 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 the decision whether the multiple product is divisible by 3 can be computed in TC^0_2. 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. This indicates that we have to treat powers of 3 differently.
One way out is to consider the "inner product modulo 3", IP^*, defined by $IP^* = x_1y_1 + \cdots + x_ny_n \bmod 3$.
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 [KW] 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 the inner product modulo 3 (over 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^0_2 ([K2]).
This suggests that one should try to find 3-PLCs modulo 3^i. We omit the details why the existence of a 3-PLC modulo 3^i guarantees that there is a projection reduction from $f^{(IP3)}$ to $MP^{(3^i)}$ and report instead the 3-PLCs that we have found.
We need the following number theoretic lemma, (see e.g. [S], p.61)

Proposition 5.2 *If p is an odd prime, then the set of primitive roots modulo p^i (for $i \geq 2$) consists of all elements x which are primitive roots modulo p and which fulfill $x^{p-1} \not\equiv 1 \bmod p^2$.*

One easy to check consequence of this proposition is that 2 is a primitive root modulo 3^i for all $i \geq 2$.

Lemma 5.3 *Let $i \geq 2$ and define parameters a, b, w by*

$$w = 2^{2 \cdot 3^{i-2}}, \quad b = \frac{w+1}{2} - 1, \quad a = \frac{2}{w+1} - 1$$

Then the following PLC f is a 3-PLC modulo 3^i:

$$f := (-2x+1)(-2x+ay+1)(by+1)$$

Proof: We first have to ensure that the inverses modulo 3^i used in the definition of the parameters do exist. For $1/2$, 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 $f(1,1) = (1-a)(b+1) \equiv w \bmod 3^i$. The order of $f(1,1)$ is 3, by the following argument: 2 is a primitive root modulo 3^i, hence it has order $2 \cdot 3^{i-1}$. Thus, w has order 3. •

Example 5.4 *Applying lemma 5.3 to $i = 2$, we get $w = 4, b = 6, a = 3$ to find that*

$$(-2x+1)(-2x+3y+1)(6y+1)$$

is a 3-PLC modulo 9.

This shows that there is some bit in the binary representation of $MP^{(3^i)}, i \geq 2$ fixed, which cannot be computed in TC^0_2.
We have noted earlier that the technique from lemma 3.2 can also be used to obtain 3-PLCs for numbers m which are of the form $3^i \cdot r$, $i \geq 2$.

This yields that the only moduli that we were not able to classify are of the form $3 \cdot 2^i$, with $i \leq 3$ 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 fixed numbers $m \notin \{3, 6, 12, 24\}$, whether computing 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, nevertheless, we have not found a 2-PLC which is also representable. One example of such a PLC is $(-2x + 12y + 1)(-2x + 18y + 1)(6y + 1)$.

7 Acknowledgments

Our thanks go to Petr Savický for suggesting to us an improvement in theorem 5.1, to Hanno Lefmann for pointing us to Proposition 5.2, and to Matthias Krause for stimulating discussions, in particular for suggesting the investigation of 3-PLCs instead of 2-PLCs for computing modulo 9.

References

[HMPST] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, G. Turán, *Threshold circuits of bounded depth,* Proceedings of 28th FOCS, 1987, 99-110.

[GHR] M. Goldmann, J. Håstad, A. Razborov, *Majority gates vs. general weighted threshold gates,* Proceedings of 7th Annual Structure in Complexity Theory Conference (1992), pp. 2-13.

[GK] M. Goldmann, M. Karpinski, *Simulating threshold circuits by majority circuits,* Proc. 25th STOC, 1993, p. 551-560.

[H] T. Hofmeister, *Depth-efficient threshold circuits for arithmetic functions,* Chapter 2 in: Theoretical Advances in Neural Computation and Learning, V. Roychowdhury, K-Y. Siu, and A. Orlitsky (eds.), Kluwer Academic Publishers, ISBN 0-7923-9478-X

[K1] M. Krause, *On realizing iterated multiplication by small depth threshold circuits,* to appear in: Proceedings of 12th STACS (1995).

[K2] M. Krause, pers. comm.

[KW] M. Krause, S. Waack, *Variation ranks of communication matrices and lower bounds for depth two circuits having symmetric gates with unbounded fan-in,* Proc. of 32nd FOCS, 1991, 777-782.

[R] A. Razborov, *On small depth threshold circuits,* In Proc. 3rd Scandinavian Workshop on Algorithm Theory, 42-52, LNCS 621, 1992.

[S] A. Scholz,B. Schoeneberg, *Einführung in die Zahlentheorie,* Sammlung Göschen, Band 5131, Walter de Gruyter, 1973

[SBKH] K.-Y. Siu, J. Bruck, T. Kailath, T. Hofmeister, *Depth efficient neural networks for division and related problems,* IEEE Transactions on Information Theory, May 1993, p. 946-956.

[SR] K.-Y. Siu, V. Roychowdhury, *On optimal depth threshold circuits for multiplication and related problems,* SIAM Journal on Discrete Mathematics 7 (1994), p. 285-292.

[W] I. Wegener, *Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions,* Information Processing Letters 46 (1993), p. 85-87.