

Hilbert’s thirteenth problem and circuit complexity

Kristoffer Arnsfelt Hansen¹, Oded Lachish², and Peter Bro Miltersen¹

¹ Department of Computer Science, Aarhus University,
`{arnsfelt,bromille}@cs.au.dk`

² Department of Computer Science, University of Warwick,
`O.Lachish@warwick.ac.uk`

Abstract. We study the following question, communicated to us by Miklós Ajtai: Can all explicit (e.g., polynomial time computable) functions $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ be computed by word circuits of constant size? Here, a word circuit is an acyclic circuit where each wire holds a word (i.e., an element of $\{0, 1\}^w$) and each gate G computes some binary operation $g_G : (\{0, 1\}^w)^2 \rightarrow \{0, 1\}^w$, defined for all word lengths w . As our main result, we present an explicit function so that its w ’th slice for any $w \geq 8$ cannot be computed by word circuits with at most 4 gates. Also, we formally relate Ajtai’s question to open problems concerning ACC^0 circuits.

1 Introduction

1.1 Statement and background of problem considered

A *word* is a bit string of length w , where w is a parameter called the *word length*. We define a *word circuit* to be an acyclic circuit where each wire holds a word and each gate G computes some binary operation $g_G : (\{0, 1\}^w)^2 \rightarrow \{0, 1\}^w$, defined for all word lengths w . A word circuit with k input wires computes a function $g : (\{0, 1\}^w)^k \rightarrow \{0, 1\}^w$ in the natural way. Consider some ternary operation on words $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ defined for all word length $w = 1, 2, 3, \dots$. We say that f is *decomposable* if f can be computed by a constant size word circuit. Otherwise, f is called non-decomposable. For instance, identifying $\{0, 1\}^w$ with $\{0, 1, 2, \dots, 2^w - 1\}$, the function $f(x, y, z) = x + y + z \bmod 2^w$ is decomposable, as $f(x, y, z) = g(g(x, y), z)$, where $g(x, y) = x + y \bmod 2^w$. It is much harder to give examples of natural functions that are provably non-decomposable. However, a simple counting argument (for details, see Section 2) shows that functions $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ exist so that any word circuit computing the slice of the function corresponding to word length w has size at least roughly 2^w . In particular, such an f is non-decomposable. Standard arguments translate this non-constructive existence result into a function computable in EXPSPACE (in fact, the second level of the exponential hierarchy) with this property.

The central complexity theoretic question we investigate is the following:

Main Question: *Are all polynomial time computable functions $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ decomposable?*

This question was communicated to us by Miklós Ajtai (personal communication). We believe that the question should be resolved in the negative. That is, we conjecture the existence of ternary polynomial time computable functions that can not be expressed using a constant number of binary functions. However, the present paper leaves the question unresolved. On the other hand, we prove some weaker statements and present reductions suggesting that the question might be difficult to resolve. Before presenting these, we explain how the question and the study of word circuits in general are motivated.

Hilbert’s thirteenth problem: Our main question can be viewed as a discrete analogue of Hilbert’s 13th problem. In fact, according to personal communication with Miklós Ajtai, a version of the question (where “computable in polynomial time” was replaced with the less tangible “natural”) has circulated in the combinatorics community with this motivation. We are, however, not aware of any previous published work on the problem. We briefly recall Hilbert’s 13th problem to make the correspondence explicit. Hilbert asked if a solution to the 7th degree equation $x^7 + ax^3 + bx^2 + cx + 1 = 0$ as a function of a, b, c can be expressed using a finite number of continuous two-variable functions, and conjectured that the question should be answered negatively. A natural more general question is: *Can all continuous functions of three real variables be expressed using a finite number of continuous two-variable functions?* Very surprisingly, this more general question was resolved in the positive (thus refuting Hilbert’s conjecture) by Kolmogorov and Arnold in 1957 [1, 4]. The Kolmogorov-Arnold representation theorem states that any continuous 3-variable function f in three variables can be written $f(x_1, x_2, x_3) = \sum_{i=0}^6 g_i(\sum_{j=1}^3 \phi_{ij}(x_j))$, where g_i, ϕ_{ij} are continuous one-variable functions. That is, the only two-variable function needed is “+”. On the other hand, variants of the question where “continuous” are replaced by “analytic” or “smooth” have been resolved in the negative (see, e.g., [10]). We ask the analogous question in a finitary computational setting with the continuity condition on f being replaced by a polynomial time computability condition. Arguably, one would obtain a more faithful discrete computational analogue by also insisting that the *gates* in the decomposition compute efficiently computable functions. This variant may be easier to answer in the negative. However, we conjecture that even the version with gates computing arbitrary functions should be resolved in the negative.

Network coding: Our main question is arguably one of the simplest questions one can ask about the word circuit model introduced above. The study of word circuits in general is motivated by their relationship to *network coding, word RAMs*. In particular, Adler *et al.* considers the *transposition* problem $t : (\{0, 1\}^w)^w \rightarrow (\{0, 1\}^w)^w$ where $t(M) = M^T$ when the input and output are interpreted as matrices. They conjecture that any “oblivious I/O machine” computing t requires $\Omega(w \log w)$ I/O’s and show that this conjecture is implied by the central “undirected k-pairs conjecture” of network coding. It is easy to see that their conjecture is *exactly* equivalent to conjecturing that any word circuit

computing t has size $\Omega(w \log w)$. It seems likely (though we admit to having no formal argument for this) that one should first understand apparently very simple questions about word circuits such as our main question concerning 3-input functions, before one can make progress on questions such as the undirected k -pairs conjecture.

1.2 Our Results

We show (in Section 2) that there exists a function $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ that requires word circuits of size $(1 - o(1))2^w$. We match this with an upper bound: All functions $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ can be computed by a word circuit of size $(2 + o(1))2^w$. The lower bound is a standard non-constructive counting argument, i.e., the lower bound holds for a random function. The most important thing to notice is that the upper and lower bounds are off by a factor of roughly two. In contrast, for the case of Boolean circuits, the circuit size of the worst case Boolean function is known up to a low order term [6]. We consider it an interesting open problem to get similar tight bounds for word circuits.

As our main technical result, we show (in Section 3) that there exists an explicit (polynomial time computable) function $F : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ so that for all $w \geq 8$, the w 'th slice of F cannot be computed by a word circuit of size 4. While this lower bound is not astronomical, it is non-trivial; note that many interesting ternary functions have word circuits of size 2! Also, since the lower bound is valid even for small word lengths, it can be meaningfully interpreted in realistic settings as a lower bound on the number of computational instructions needed to compute F by any CPU limited to binary computational instructions. We have not been able to prove a better lower bound for any explicit function. Thus, a natural open question left by this work is: *Can all polynomial time computable ternary functions be computed by word circuits of size 5?* Obviously, we believe that the answer is no. Also, we are optimistic that this answer can be obtained using current techniques.

Finally, we show (in Section 4) that resolving our main question in the negative (i.e., proving non-constant lower bounds on word circuit size for ternary functions) is likely to be somewhat hard, by relating it to open problems in *Boolean* circuit complexity. More precisely, while non-linear bounds on the number of *wires* of ACC^0 circuits are known for explicit functions [5], no such lower bounds are currently known for the number of *gates* for explicit functions, even multi-output functions. We show that if a Boolean function $f : \{0, 1\}^{3w} \rightarrow \{0, 1\}^w$ has an ACC^0 circuit of size (i.e., number of gates) $O(w)$, then f , viewed as a function mapping three words to one word, is decomposable. Thus, resolving our main question in the negative would lead to new ACC^0 lower bounds. As the version of the question where “polynomial time computable” is replaced with “natural” is a natural question from the perspective of “pure” (i.e., non-computational) combinatorics³, we find it interesting that one can successfully

³ We can argue that it is indeed so in two different ways. Empirically: It has circulated in the combinatorics community. Philosophically: It is concerned with the existence

gauge its difficulty by referring to the current state of the art of Boolean circuit lower bounds.

1.3 Related research

The present paper is about bounding the number of *gates* in a *fan-in 2* circuit with gates computing arbitrary functions on a non-Boolean domain. A sequence of papers [8, 9, 2] have studied a somewhat related model: *Unbounded* fan-in, bounded-depth circuits with gates computing arbitrary functions on a *fixed* non-Boolean domain. There, non-linear lower bounds on the number of *wires* required for computing certain natural multi-output functions were given, with the constant in the big- Ω depending on the size of the domain.

2 The complexity of the hardest ternary function

Proposition 1. *There exists a function $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ so that no word circuit of size smaller than $(1 - o(1))2^w$ computes f .*

Proof. We phrase the standard counting argument using Kolmogorov complexity lingo for readability: A word circuit of size s can be described using $s(2 \log_2(s + 3) + w2^{2w})$ bits (the first term accounts for describing the wiring of inputs to the gates, the second for describing the binary function of the gate). A random ternary function has no description shorter than $w2^{3w}$ bits. Thus, if s is an upper bound on the word circuit size of all functions, we have

$$s(2 \log_2(s + 3) + w2^{2w}) \geq w2^{3w}$$

or $s \geq (1 - o(1))2^w$.

This lower bound is matched within a constant factor by the following upper bound.

Theorem 1. *Every function $f : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ is computed by a word circuit of size at most $(2 + o(1))2^w$.*

Proof. Let the names of the three input variables be x, y, z . Partition the set $\{0, 1\}^w \times \{0, 1\}^w$ into $m = \lceil \frac{2^{2w}}{2^w - 1} \rceil$ sets P_1, \dots, P_m , each of size at most $2^w - 1$. For all i , fix injections $\pi_i : P_i \rightarrow \{0, 1\}^w \setminus \{0^w\}$, and define gates g_1, \dots, g_m and h_1, \dots, h_m as follows, letting g_0 denote x .

$$h_i(y, z) = \begin{cases} \pi_i(y, z) & \text{if } (y, z) \in P_i \\ 0^w & \text{if } (y, z) \notin P_i \end{cases}$$

of *one* circuit (or formula) rather than a family of such. Compared to a single formula, families of circuits or formulas satisfying a certain size restriction (say, polynomial or linear size) appear to be much less natural objects unless one specifically adopts a computational perspective.

$$g_i(g_{i-1}, h_i) = \begin{cases} f(g_{i-1}, \pi_i^{-1}(h_i)) & \text{if } h_i \neq 0^w \\ g_{i-1} & \text{if } h_i = 0^w \end{cases}$$

The output of the circuit is the gate g_m . It is readily verified that the circuit computes the function f correctly, and the size of the circuit is $2m = (2+o(1))2^w$.

We consider it an interesting open problem to get bounds tight within a low order term, similar to the bounds known for Boolean circuits. Note that the word circuit constructed in our upper bound proof is actually a *formula*. Thus, there is at most a difference of a factor of roughly two between the maximum possible word formula size and the maximum possible word circuit size. Again, this contrasts the case of Boolean circuits and formulas, where the hardest n -bit function has circuit size $\approx 2^n/n$ and formula size $\approx 2^n/\log n$ by results of Lupanov [6, 7] (see, e.g., Wegener [11] for an exposition).

3 An explicit lower bound

In this section, we prove our main technical result:

Theorem 2. *There exists a polytime computable function $F : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ so that for all $w \geq 8$, the w 'th slice of F cannot be computed by a word circuit of size 4.*

The section is organized as follows. First we present some structural theorems about word circuits in general and word circuits of size at most 4 in particular. Next, we define the function F and establish certain properties of F . Finally, we combine the two groups of statements to complete the proof that no word circuit of size at most 4 computes F .

Definition 1. Let C be a word circuit. We say a gate g is *redundant* if either of the following holds.

1. One of the inputs of g is a gate g^* and the other input of g is an input of g^* .
2. The inputs of g are gates g_1 and g_2 that have the exact same set of inputs.

A word circuit is called *redundancy free* if it does not contain any redundant gates.

Claim 3 *Let C be a word circuit that contains a redundant gate. There exists a redundancy free circuit C^* of size at most the size of C that computes the same function C does.*

Proof. We replace a redundant gate g of the first kind, computing $g(u, g^*(u, v))$ with a gate g' computing $g'(u, v) = g(u, g^*(u, v))$. We replace a redundant gate g of the second kind, computing $g(g_1(u, v), g_2(u, v))$ with a gate g' computing $g'(u, v) = g(g_1(u, v), g_2(u, v))$. If C turns out to be again redundant, we repeat this transformation. This process is easily seen to eventually terminate as each transformation structurally simplifies the circuit.

Definition 2. Let C be a word circuit. If there exists a variable in C that is connected to the output of the circuit by a unique path, we say C is *weak*. We also refer to such a variable as a *weak variable*. In general, a weak circuit may have more than one weak variable. In such a case we pick one of them arbitrarily to be *the* weak variable. A gate in C is called a *weak gate* if it is on the path connecting the weak variable to the output. Each one of the weak gates has an input that is not on the path connecting the weak variable to the output. We refer to this input as the *control input* of the gate.

Lemma 1. Let C be a redundancy free word circuit of size exactly 4, that computes a function that depends on all 3 inputs. Then C is weak and satisfies one of the following.

1. There exist functions Y_1, Y_2 on the non-weak inputs so that the control input of each weak gate is one of Y_1, Y_2 .
2. The control input of all but at most one of the weak gates is one of the non-weak input variables.

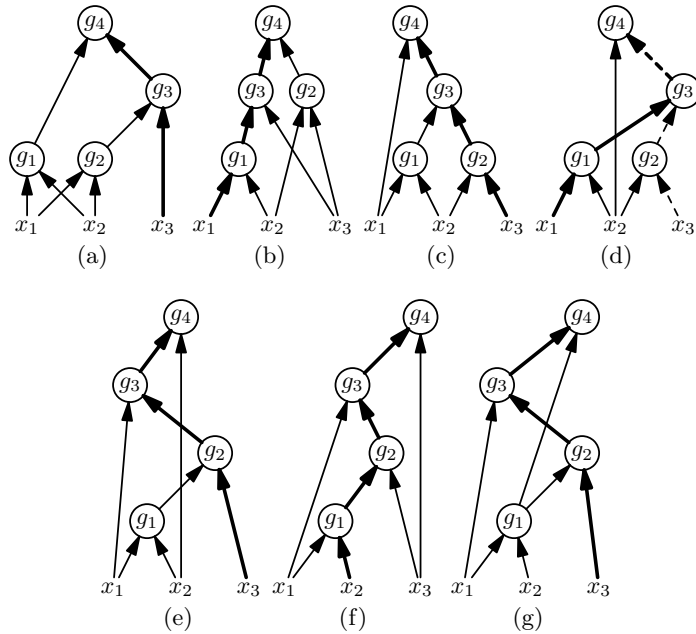


Fig. 1. All redundancy free circuits of size 4. The paths from weak variables are bold/dashed.

Proof. Let C be a redundancy free word circuit of size 4. Since every gate takes 2 inputs the circuit contains 8 wires. Except for the output gate, all gates as well

as the input variables must have an outgoing wire, thus accounting for 6 of the wires. If an internal gate has 3 outgoing wires, one of these must take another one of these as input and hence C is redundant. We are left with the following 3 cases.

1. There is an input variable that has 3 outgoing wires. In this case all gates have 1 outgoing wire and hence there is a unique path from the other 2 input variables to the output gate.
2. There are two input variables that have 2 outgoing wires. Here again all gates have 1 outgoing wire, hence there is a unique path from the last input variable to the output gate.
3. There is one input variable that has 2 outgoing wires and a gate g having 2 outgoing wires. If one of the other input variables does not have a path to g , then it will have a unique path to the output. If both of the other input variables had a path to g , then they had to be directly connected to g , as otherwise the circuit would be redundant. But then all of the last 3 gates would depend only on g and the third input variables, meaning that the circuit would be redundant.

In all cases we identify a weak variable, and hence C must be weak. The gates that are not on the path from the weak input variable to the output must depend solely on the other 2 input variables. We can have either 2 or 1 of these. When C contains 2 non-weak gates, both of the weak gates must take these as inputs and we have the first case of the statement. When C has a 1 non-weak gate, it is either connected to 2 or 1 weak gates in which case we have the first or the second case of the statement, respectively.

The above proof identifies several classes of circuits of size more than 4 that are not weak. For instance all 3 input variables may have 2 outgoing wires. All the redundancy free circuits of size 4 (up to relabeling) are shown in Figure 1. The first case of Lemma 1 occurs for circuits (a) and (g), whereas the second case occurs for the remaining circuits.

Remark From here on, we assume $w \geq 8$.

Definition 3. We define $F : (\{0, 1\}^w)^3 \rightarrow \{0, 1\}^w$ as follows

1. If $\alpha_i = \alpha_j = 0^w$ for distinct i, j then $F(\alpha_0, \alpha_1, \alpha_2) = 0^w$.
2. If $\alpha_i = 0^w$ and $\alpha_j, \alpha_k \neq 0^w$ for distinct i, j, k then $F(\alpha_0, \alpha_1, \alpha_2) = \alpha_j^\ell \cdot \alpha_k^{-\ell}$, where $\ell = 1$ if $k - j \equiv 1 \pmod{3}$ and $\ell = -1$ otherwise. Here, the multiplication is over $\text{GF}(2^w)$, i.e., we identify $\{0, 1\}^w$ with $\text{GF}(2^w)$.
3. If $\alpha_0, \alpha_1, \alpha_2$ are distinct and all different from 0 then if there exists $\ell \in \{0, 1, 2\}$ such that $\alpha_0 + \alpha_1 + \alpha_2 \in \{8\ell + 1, \dots, 8\ell + 8\}$ we set $F(\alpha_0, \alpha_1, \alpha_2) = \alpha_\ell$ and otherwise $F(\alpha_0, \alpha_1, \alpha_2) = \alpha_0 + \alpha_1 + \alpha_2$. Here, the addition is modulo 2^w , i.e., we identify $\{0, 1\}^w$ with $\mathbf{Z}/2^w\mathbf{Z}$.
4. If $\alpha_0, \alpha_1, \alpha_2$ are not 0 and $\alpha_i = \alpha_j$ then if $\alpha_k < \alpha_i$ we set $F(\alpha_0, \alpha_1, \alpha_2) = 1$ and otherwise $F(\alpha_0, \alpha_1, \alpha_2) = 1 + \alpha_k - \alpha_i$. Here, the arithmetic and the inequalities are defined by identifying $\{0, 1\}^w$ with $\{0, 1, 2, \dots, 2^w - 1\}$.

Before analyzing the properties of F , we introduce some useful concepts. We let the variables of F be denoted X_0, X_1, X_2 . For $\alpha, \beta, \gamma \in \{0, 1\}^w$ and distinct $i, j, k \in \{0, 1, 2\}$ we let $F_{\uparrow X_i=\alpha, X_j=\beta}(\gamma)$ be the value of F when $X_i = \alpha, X_j = \beta$ and $X_k = \gamma$. For $\alpha, \beta \in \{0, 1\}^w$ and distinct $i, j \in \{0, 1, 2\}$ we let $F_{\uparrow X_i=\alpha, X_j=\beta}$ be the function we get from F over the last variable X_k when $X_i = \alpha, X_j = \beta$. For any function T and S that is a subset of T 's domain we let $T(S) = \{T(\gamma) : \gamma \in S\}$. When S is the domain of T we abuse notation and write $|T|$ instead of $|T(S)|$. Let $T : \{0, 1\}^w \rightarrow \{0, 1\}^w$. For $\alpha \in \{0, 1\}^w$, let $T^{-1}(\alpha) = \{\beta : T(\beta) = \alpha\}$. We let $M(T)$ be the $(2^w + 1)$ -tuple $M(T)_\ell = |\{\alpha : |T^{-1}(\alpha)| = \ell\}|$ for $\ell \in \{0, \dots, 2^w\}$ and we let $\text{Mask}(F, i, j) = \{M(F_{\uparrow X_i=\alpha, X_j=\beta}) : \alpha, \beta \in \{0, 1\}^w\}$. Also, F may be replaced by a circuit C in all the above notation.

Claim 4 *For every distinct $i, j \in \{0, 1, 2\}$ and distinct $\alpha, \beta \in \{1, \dots, 2^w - 1\}$ the following are satisfied:*

- $|F_{\uparrow X_i=0, X_j=0}| = 1$.
- $|F_{\uparrow X_i=\alpha, X_j=0}| = 2^w$.
- $|F_{\uparrow X_i=\alpha, X_j=\beta}| \geq 2^w - 26$.
- $|F_{\uparrow X_i=\alpha, X_j=\alpha}| = 2^w - \alpha$.
- Let S be the set of all (γ, δ) , where $\gamma, \delta \in \{0, 1\}^w$ and $|F_{\uparrow X_i=\gamma, X_j=\delta}| < 2^w - 2^5$. Then, $|S| < 2^w$.

Proof. The first two equalities follow from the definition of F . Let S be the set of all $\gamma \in \{0, 1\}^w$ satisfying $\gamma \notin \{0, \alpha, \beta\}$ and $\alpha + \beta + \gamma \notin \{1, \dots, 24\}$. Obviously, $|S| \geq 2^w - 26$. By the definition of F for every $\gamma \in S$ we have that $F_{\uparrow X_i=\alpha, X_j=\beta}(\gamma) = \alpha + \beta + \gamma$ and hence $|F_{\uparrow X_i=\alpha, X_j=\beta}| \geq |F_{\uparrow X_i=\alpha, X_j=\beta}(S)| \geq 2^w - 26$. By Condition 4 of Definition 3 we have that $|F_{\uparrow X_i=\alpha, X_j=\alpha}| = 2^w - \alpha$ for every $\alpha \in \{1, \dots, 2^w - 1\}$.

From this, we also have that $|F_{\uparrow X_i=\gamma, X_j=\delta}| < 2^w - 2^5$ only if $\gamma = \delta$ and either $\gamma = 0$ or $\gamma > 2^5$. Hence the set of all (γ, δ) , where $\gamma, \delta \in \{0, 1\}^w$ and $|F_{\uparrow X_i=\gamma, X_j=\delta}| < 2^w - 2^5$, has less than 2^w members.

Claim 5 $|\text{Mask}(F, i, j)| > 2^w$ for every distinct $i, j \in \{0, 1, 2\}$.

Proof. To prove the claim we show that $\text{Mask}(F, i, j)$ contains more than 2^w different tuples. Condition 1 of Definition 3 asserts that $M(F_{\uparrow X_i=0, X_j=0})$ is zero on all coordinates except for $M(F_{\uparrow X_i=0, X_j=0})_{2^w} = 1$. Condition 2 asserts that $M(F_{\uparrow X_i=0, X_j=1})$ is zero on all coordinates except for $M(F_{\uparrow X_i=0, X_j=1})_1 = 2^w$. Condition 4 asserts that for every $\alpha \in \{1, \dots, 2^w - 1\}$ we have $M(F_{\uparrow X_i=\alpha, X_j=\alpha})$ is zero on all coordinates except for $M(F_{\uparrow X_i=\alpha, X_j=\alpha})_1$ which is equal to $2^w - \alpha - 1$ and $M(F_{\uparrow X_i=\alpha, X_j=\alpha})_{\alpha+1}$ which is 1. This constitutes at least 2^w different tuples. We next show that there is at least one more tuple that we have not described yet.

For every $\alpha \in \{8i + 1, \dots, 8i + 8\}$ we have that $F_{\uparrow X_i=1, X_j=2}(\alpha) = 1$ and for every $\alpha \in \{8j + 1, \dots, 8j + 8\}$ we have that $F_{\uparrow X_i=1, X_j=2}(\alpha) = 2$. Consequently, one of the following is true: there exists a coordinate $k > 1$ such that $M(F_{\uparrow X_i=1, X_j=2})_k > 1$, or, there exists two distinct coordinates $k_1, k_2 > 1$ such

that $M(F_{\uparrow X_i=1, X_j=2})_{k_1} \geq 1$ and $M(F_{\uparrow X_i=1, X_j=2})_{k_2} \geq 1$. Note that in both cases $M(F_{\uparrow X_i=1, X_j=2})$ is different from all the other tuples mentioned above.

Claim 6 $F_{\uparrow X_i=\alpha, X_j=\beta} \neq F_{\uparrow X_i=\gamma, X_j=\delta}$ for every distinct $i, j \in \{0, 1, 2\}$ and $\alpha, \beta, \gamma, \delta \in \{0, 1\}^w$ such that $(\alpha, \beta) \neq (\gamma, \delta)$.

Proof. Let i, j be distinct members of $\{0, 1, 2\}$ and let $\alpha, \beta, \gamma, \delta \in \{0, 1\}^w$ such that $(\alpha, \beta) \neq (\gamma, \delta)$. Let $\ell = 1$ if $i - j \equiv 1 \pmod{3}$ and otherwise $\ell = -1$. We prove the claim by showing that there exists κ such that $F_{\uparrow X_i=\alpha, X_j=\beta}(\kappa) \neq F_{\uparrow X_i=\gamma, X_j=\delta}(\kappa)$. This directly implies the claim.

1. Assume that $\alpha = \beta = 0$. Consequently at least one of γ, δ differs from 0. Hence, by Condition 4 and Condition 2 of Definition 3 we get that $F_{\uparrow X_i=\gamma, X_j=\delta}(\max\{\gamma, \delta\}) = 1$. By Condition 1 of Definition 3 we get that $F_{\uparrow X_i=\alpha, X_j=\beta}(\max\{\gamma, \delta\}) = 0$.
2. Assume that exactly one of α, β is different from 0 and at least one of γ, δ is different from 0 (due to symmetry the case that $\gamma = \delta = 0$ was dealt with previously). Without loss of generality assume $\alpha = 0$ and $\beta \neq 0$.
 - Assume that $\gamma = 0$ and $\delta \neq 0$. Thus, $\beta \neq \delta$ and hence Condition 2 of Definition 3 asserts that $F_{\uparrow X_i=\gamma, X_j=\delta}(\beta) \neq 1$ and $F_{\uparrow X_i=\alpha, X_j=\beta}(\beta) = 1$.
 - Assume that $\gamma \neq 0$ and $\delta = 0$ and $\beta \neq \gamma^{-1}$. Then by Condition 2 of Definition 3 we get that $F_{\uparrow X_i=\alpha, X_j=\beta}(1) = \beta^\ell \neq \gamma^{-\ell} = F_{\uparrow X_i=\gamma, X_j=\delta}(1)$.
 - Assume that $\gamma \neq 0$ and $\delta = 0$ and $\beta = \gamma^{-1} = \eta$. Let κ be such that $\kappa \neq \kappa^{-1}$. Now by Condition 2 of Definition 3 we get that $F_{\uparrow X_i=\alpha, X_j=\beta}(\kappa) = \eta^\ell \kappa^\ell \neq \eta^\ell \kappa^{-\ell} = F_{\uparrow X_i=\gamma, X_j=\delta}(\kappa)$.
 - Assume that $\gamma, \delta \notin \{0\}$. Thus $\gamma^\ell \delta^{-\ell} \neq 0$ and hence Condition 2 of Definition 3 asserts that $F_{\uparrow X_i=\gamma, X_j=\delta}(0) \neq 0$. Condition 1 of Definition 3 asserts that $F_{\uparrow X_i=\alpha, X_j=\beta}(0) = 0$.
3. Assume that $\alpha = \beta \neq 0$ and $\gamma, \delta \notin \{0\}$ (due to symmetry the case that at least one of γ, δ is 0 was dealt with previously).
 - Assume that $\gamma = \delta$. Thus $\alpha \neq \gamma$ and hence Condition 4 of Definition 3 implies that $F_{\uparrow X_i=\gamma, X_j=\delta}(\max\{\alpha, \gamma\}) \neq F_{\uparrow X_i=\alpha, X_j=\beta}(\max\{\alpha, \gamma\})$, since it asserts that one side of the inequality is 1 and the other side is strictly greater than 1.
 - Assume that $\gamma \neq \delta$. Then Condition 2 of Definition 3 implies that $F_{\uparrow X_i=\gamma, X_j=\delta}(0) \neq 1$ and $F_{\uparrow X_i=\alpha, X_j=\beta}(0) = 1$.
4. Assume that $\alpha \neq \beta, \gamma \neq \delta$ and $\alpha, \beta, \gamma, \delta \notin \{0\}$ (All other cases have already been dealt with).
 - Assume $\alpha + \beta = \gamma + \delta$. This implies that $\alpha \neq \gamma$. Choose $\kappa \notin \{0, \alpha, \beta, \gamma, \delta\}$ such that $\alpha + \beta + \kappa \in \{8i + 1, \dots, 8i + 8\}$. Then by Condition 3 of Definition 3 we have that $F_{\uparrow X_i=\alpha, X_j=\beta}(\kappa) = \alpha \neq \gamma = F_{\uparrow X_i=\gamma, X_j=\delta}(\kappa)$.
 - Assume $\alpha + \beta \neq \gamma + \delta$. Then as $2^w > 54$ there exists κ such that Condition 3 of Definition 3 asserts that $F_{\uparrow X_i=\alpha, X_j=\beta}(\kappa) = \alpha + \beta + \kappa \neq \gamma + \delta + \kappa = F_{\uparrow X_i=\gamma, X_j=\delta}(\kappa)$.

Lemma 2. *Let C be a weak circuit for which there exist functions Y_1, Y_2 such that the control input of each weak gate is equivalent to one of Y_1, Y_2 . If C computes F , then C has depth at least 2^{w-5} .*

Proof. Let the input variables be X_0, X_1, X_2 and let the weak variable be X_0 . Assume for the sake of contradiction that C has depth strictly less than 2^{w-5} .

For each $i \in \{1, 2\}$ let G_i be the set of all weak gates g such that Y_i is equivalent to the control input of g . Observe that Claim 4 asserts that $|F_{\uparrow X_1=0, X_2=0}| = 1$. Hence as the depth of C is at most 2^{w-5} we conclude that there exists $i \in \{1, 2\}$ and a gate $g \in G_i$ such that $|g_{\uparrow Y_i(0,0)}| < 2^w - 2^5$. Let S be the set of all (α, β) such that $Y_i(\alpha, \beta) = Y_i(0, 0)$. Now as $|g_{\uparrow Y_i(0,0)}| < 2^w - 2^5$ we have that $|C_{\uparrow X_1=\alpha, X_2=\beta}| < 2^w - 2^5$ for every $(\alpha, \beta) \in S$. Hence, $|S| < 2^w$ by Claim 4.

We now show that we can also derive that $|S| = 2^w$. That is, we get the required contradiction. Let $\alpha, \beta, \gamma, \delta \in \{0, 1\}^w$ be such that $(\alpha, \beta) \neq (\gamma, \delta)$. By Claim 6 we have that $F_{\uparrow X_1=\alpha, X_2=\beta} \neq F_{\uparrow X_1=\gamma, X_2=\delta}$ and thus $(Y_1(\alpha), Y_2(\beta)) \neq (Y_1(\gamma), Y_2(\delta))$. Thus, $(Y_1(X_1, X_2), Y_2(X_1, X_2))$ is a bijection of (X_1, X_2) and therefore $|S| = 2^w$.

Lemma 3. *Let C be a weak circuit such that the control input of all but at most one of the weak gates is equivalent to one of the non-weak variables. Then, C does not compute F .*

Proof. Let the input variables be X_0, X_1, X_2 and let the weak variable be X_0 . Assume for the sake of contradiction that C computes F . Thus, $|\text{Mask}(C, 1, 2)| > 2^w$ by Claim 5. We shall get the required contradiction by showing that we can also derive that $|\text{Mask}(C, 1, 2)| \leq 2^w$.

By Claim 4 we have that $|C_{\uparrow X_1=\alpha, X_2=\beta}| = 2^w$ if either $\alpha = 0$ or $\beta = 0$ but not both. Thus, $|g_{\uparrow Y=\alpha}| = 2^w$ for every weak gate g whose control input Y is equivalent to one of X_1, X_2 . Assume there does not exist a weak gate g whose control input Y is neither equivalent to X_1 nor to X_2 . Then, $|\text{Mask}(C, 1, 2)| = 1$ because the output of C is a permutation of X_0 regardless for any values assigned to X_1, X_2 . Assume that t is a weak gate, whose control input Y is neither equivalent to X_1 nor to X_2 . Recall that t can be the only such gate and hence $|\text{Mask}(C, 1, 2)| = |\{M(g_{\uparrow Y=\alpha}) : \alpha \in \{0, 1\}^w\}| \leq 2^w$.

From Lemma 1, Lemma 2 and Lemma 3 we conclude:

Corollary 1. *Any redundancy free word circuit that computes F for any fixed $w \geq 8$ must be of size at least 5.*

Finally by applying Claim 3 we get the statement of the main Theorem.

4 Converting ACC^0 circuits to word circuits

Our simulation of ACC^0 circuits by word circuits can be regarded as a simple application of a technique from the word RAM literature [3]: *word parallelism*, i.e., the technique of using a gate operating on words as a miniature vector processor. The technique works for ACC^0 circuits as the intermediate results needed in the simulation of such a circuit have bounded bit-size and can therefore be compactly represented within a word. In contrast, for slightly more powerful circuit classes (such as TC^0), we do not know a simulation that converts non-trivial circuits into constant size word circuits.

Lemma 4. *Let k and m be positive integers so that $km \leq w$. Let $+^{(1)}, +^{(2)}, \dots, +^{(m)}$ be associative and commutative operations on $K = \{0, 1\}^k$. For some constant a , embed the domain K^m into $\{0, 1\}^w$ and the domain K^{ma} into $(\{0, 1\}^w)^a$ by identifying consecutive blocks of bits with elements of K . Let sets $S_1, \dots, S_m \subseteq \{1, 2, \dots, ma\}$ be given. Consider a function $g : K^{ma} \rightarrow K^m$ defined in the following way: $g(x_1, x_2, \dots, x_{ma}) = (\sum_{i \in S_1}^{(1)} x_i, \dots, \sum_{i \in S_m}^{(m)} x_i)$ where $\sum^{(j)}$ is summation with respect to $+^{(j)}$. Then g has a word circuit of size $a - 1$.*

Proof. The word circuit is a tree (i.e., a formula) consisting of gates computing pointwise addition of two elements of K^m , with the j 'th entries added with respect to the operation $+^{(j)}$. At the bottom of the tree we have for each segment of inputs $x_i, x_{i+1}, \dots, x_{i+m-1}$ corresponding to a word a unary gate mapping this word into a word representing an element of K^m whose j 'th entry is $\sum_{\ell \in \{i, i+1, \dots, i+m-1\} \cap S_j}^{(j)} x_\ell$. These unary gates feed into the pointwise addition gates of the tree. The correctness of the construction is immediate. To make the size of the circuit exactly $a - 1$, we merge the unary gates with their immediate successors in the circuit.

Theorem 7. *Let a family of ACC^0 circuits computing a family of functions $g : \{0, 1\}^{3w} \rightarrow \{0, 1\}^w, w \geq 2k$ be given so that the w 'th circuit contains aw gates, has depth d and so that all counting gates count modulo some integer less than 2^k . Then, viewed as a function mapping three words to one, g is decomposable and has a word circuit of size less than $(d(d+1)/2)(2ka+1)^2 + 4ka + 2$.*

Proof. Let $m = \lfloor w/k \rfloor$. Divide the circuit into d layers so that each layer gets inputs from (possibly all) previous layers. Layer 0 is the layer of input bits. Add dummy gates so that each layer contains exactly $\ell = m \lceil aw/m \rceil$ gates. The *expanded* representation of the bits computed by a layer is the string of words defined as follows: Divide the bits computed by the gates in the layer into $\ell/m = \lceil aw/m \rceil \leq aw/m + 1 \leq 2ka + 1$ words. This gives us m bits per word, allowing us to embed each bit into the domain $\{0, 1\}^k$, representing 0 as 0^k and 1 as 1^k .

The expanded representation of layer 0 is easily seen to be computable by a word circuit of size at most $2ka + 1$ (one gate computes each word in the representation). By Lemma 4, assuming the computation of previous layers has been handled, the computation of one of the words of the expanded representation of layer j can be done by a word circuit of size at most $j(2ka + 1) - 1$. That is, the computation of the entire j 'th layer can be simulated by a word circuit of size less than $j(2ka + 1)^2$. So, the entire ACC^0 circuit can be simulated by a word circuit of size less than $\frac{1}{2}d(d+1)(2ka+1)^2$. Finally, a simple word circuit of size at most $2ka + 1$ converts the expanded representation of the output bits of the output layer into a single word.

This almost finishes the proof, up to one subtle point which was not relevant in this paper until now and hence ignored: To be decomposable, a function must be computed on its entire infinite domain $\cup_w (\{0, 1\}^w)^3$ by *one* word circuit, not just by a family of constant sized ones. That is, we should worry whether we

are using differently structured constant sized circuits for different word lengths. However, by inspecting the construction, we find that the structure of the circuit is in fact exactly the same for every word length. Indeed, this was our reason for adding the seemingly wasteful dummy gates.

Acknowledgments

We thank Miklós Ajtai for introducing us to the problem considered in this paper, for explaining its motivation and for helpful discussions during the course of this work.

References

1. V. I. Arnol'd. On functions of three variables. *Dokl. Akad. Nauk SSSR*, 114:679–681, 1957.
2. Dimitriy Cherkhin. Lower bounds for boolean circuits with finite depth and arbitrary gates. Technical Report TR08-032, Electronic Colloquium on Computational Complexity, 2008.
3. Torben Hagerup. Sorting and searching on the word RAM. In *STACS '98: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, pages 366–398, London, UK, 1998. Springer-Verlag.
4. A. N. Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Dokl. Akad. Nauk SSSR (N.S.)*, 108:179–182, 1956.
5. Michal Koucký, Pavel Pudlák, and Denis Thérien. Bounded-depth circuits: separating wires from gates. In *STOC '05: Proceedings of the 37th annual ACM symposium on Theory of computing*, pages 257–265, New York, NY, USA, 2005. ACM.
6. O. B. Lupanov. The synthesis of contact circuits. *Dokl. Akad. Nauk SSSR (N.S.)*, 119:23–26, 1958.
7. O. B. Lupanov. Complexity of formula realization of functions of logical algebra. *Prob. Kibernetki*, 3:782–811, 1962.
8. P. Pudlák. Communication in bounded depth circuits. *Combinatorica*, 14(2):203–216, 1994.
9. Ran Raz and Amir Shpilka. Lower bounds for matrix product, in bounded depth circuits with arbitrary gates. *SIAM J. on Computing*, 32(2):488–513, 2003.
10. A. G. Vitushkin. On Hilbert's thirteenth problem and related questions. *Russian Mathematical Surveys*, 59(1):11–25, 2004.
11. Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., New York, NY, USA, 1987.