# The Complexity of Word Circuits

Xue Chen[1], Guangda Hu[1], and Xiaoming Sun[2,⋆]

[1] Department of Computer Science and Technology, Tsinghua University
[2] Institute for Theoretical Computer Science and Center for Advanced Study, Tsinghua University

**Abstract.** A *word circuit* [1] is a directed acyclic graph in which each edge holds a $w$-bit word (i.e. some $x \in \{0,1\}^w$) and each node is a gate computing some binary function $g : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}^w$. The following problem was studied in [1]: How many binary gates are needed to compute a ternary function $f : (\{0,1\}^w)^3 \to \{0,1\}^w$. They proved that $(2+o(1))2^w$ binary gates are enough for any ternary function, and there exists a ternary function which requires word circuits of size $(1-o(1))2^w$. One of the open problems in [1] is to get these bounds tight within a low order term. In this paper we solved this problem by constructing new word circuits for ternary functions of size $(1+o(1))2^w$. We investigate the problem in a general setting: How many $k$-input word gates are needed for computing an $n$-input word function $f : (\{0,1\}^w)^n \to \{0,1\}^w$ (here $n \geq k$). We show that for any fixed $n$, $(1-o(1))2^{(n-k)w}$ basic gates are necessary and $(1+o(1))2^{(n-k)w}$ gates are sufficient (assume $w$ is sufficiently large). Since word circuit is a natural generalization of boolean circuit, we also consider the case when $w$ is a constant and the number of inputs $n$ is sufficiently large. We show that $(1 \pm o(1))\frac{2^{wn}}{(k-1)n}$ basic gates are necessary and sufficient in this case.

## 1 Introduction

*Word circuit*, defined in [1], is an acyclic graph where each edge holds a $w$-bit word and each node computes some binary word function $g : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}^w$. In this paper we extends this definition so that each node computes a $k$-input word function $g : (\{0,1\}^w)^k \to \{0,1\}^w$, where $k$ is a parameter. We call this $k$-input word function a *basic gate*, or simply gate if no confusion. For a word circuit $C$, the size of it is defined as the number of basic gates used in $C$. It is a natural question to ask: *How many basic gates are needed for computing an $n$-input word function $f : (\{0,1\}^w)^n \to \{0,1\}^w$?* Here the number of input $n \geq k$. We use symbols $x_1, x_2, \dots, x_n$ to denote the input of the word function (or the word circuit), and symbols $b_1, b_2, \dots, b_{nw}$ to denote the input bits, i.e. $x_i = b_{(i-1)w+1} b_{(i-1)w+2} \cdots b_{iw}$ $(i = 1, \dots, n)$.

⋆ Corresponding author. xiaomings@tsinghua.edu.cn. Supported in part by the National Natural Science Foundation of China Grant 60553001, 60603005, 60621062, the National Basic Research Program of China Grant 2007CB807900,2007CB807901 and Tsinghua University Initiative Scientific Research Program 2009THZ02120.

M.T. Thai and S. Sahni (Eds.): COCOON 2010, LNCS 6196, pp. 308–317, 2010.

This problem was first considered by Hansen et al. [1] for binary word gates ($k = 2$) and ternary word functions ($n = 3$). They proved that $(1-o(1))2^w$ gates are necessary and $(2 + o(1))2^w$ gates are sufficient. One of the open problem remains is to get the bound tight within a lower order term.

We answered this question in this work. We give a tight bound for the generalized problem: For every fixed $n$, there exist an $n$-input word function which needs $(1 - o(1))2^{(n-k)w}$ basic gates to compute, and $(1 + o(1))2^{(n-k)w}$ basic gates are always sufficient to build a circuit computing any $n$-input word function. Here $w$ is sufficient large (the $o(\cdot)$ notation approximates as growing of $w$). These are proved in section 2. The lower bound is proved by Shannon's counting arguments [2]. The upper bound is much more sophisticated. In [1] they build a formula for ternary word functions. Here we balanced the role of the three input words, and build a "grid" circuit. More efforts are needed for the construction of the general case.

We also consider the problem on another aspect. That is, the word length $w$ is fixed, while the desired number of inputs $n$ is sufficient large. Boolean circuits can be considered as the special case when $w = 1$ and $k = 2$. A well known result for this case is that $(1 \pm o(1))\frac{2^n}{n}$ gates are necessary and sufficient (see [2–4]). As a generalization we show that for any fixed $w$ and $k$, $(1 \pm o(1))\frac{2^{wn}}{(k-1)n}$ basic gates are necessary and sufficient. This result is given in section 3. The lower bound is proved by a similar counting argument as above. The upper bound is a modification of Lupanov's construction for boolean circuits.

## 2 Fixed Number of Input, Large Word Length

In this section, we assume that $k, n$ are constants and $w$ is sufficiently large.

### 2.1 Lower Bound

First, we have the following counting lemma:

**Lemma 1.** *A word circuit of size $s$ can compute at most $(s+n-1)^{sk}2^{sw2^{kw}}/s!$ different $n$-input functions $f : (\{0,1\}^w)^n \to \{0,1\}^w$.*

*Proof.* For each gate there are at most $(s + n - 1)^k$ ways to choose its $k$ inputs, namely the output of the other $(s-1)$ gates and $x_1, x_2, \ldots, x_n$. There are $(2^w)^{2^{kw}}$ different types of a gate. Finally each circuit is counted $s!$ times for the gates can be numbered in $s!$ different ways. Thus the lemma is proved. □

By this lemma, the lower bound is shown as following:

**Theorem 1.** *For any constant $n > k \geq 2$, there exists a word function $f$ : $(\{0,1\}^w)^n \to \{0,1\}^w$ so that no word circuit computing $f$ consists of less than $(1 - o(1))2^{(n-k)w}$ basic gates.*

*Proof.* The number of different functions $f : (\{0,1\}^w)^n \to \{0,1\}^w$ is $2^{w2^{nw}}$. By Lemma 1 we have

$$(s+n-1)^{sk} 2^{sw2^{kw}} \geq 2^{w2^{nw}} s! > 2^{w2^{nw}}.$$

Take the logarithm of both sides, we have

$$sk \log_2(s+n-1) + sw2^{kw} > w2^{nw}.$$

An easy calculation gives $s \geq (1 - O(\frac{w}{2^{kw}}))2^{(n-k)w}$. □

## 2.2 Upper Bound for $k = 2, n = 3$

Here we give a matched upper bound for the special case $k = 2, n = 3$, which was first considered in [1].

**Theorem 2.** *Every ternary function $f : (\{0,1\}^w)^3 \to \{0,1\}^w$ can be computed by a word circuit that consists of $(1 + o(1))2^w$ binary gates.*

*Proof.* We use $x_1, x_2, x_3$ to denote the 3 input words as described previously. Let $x_{2,1}$ be the first $\lfloor w/2 \rfloor$ bits of $x_2$, and $x_{2,2}$ be the last $\lceil w/2 \rceil$ bits.

Partition the $2^{w+\lfloor w/2 \rfloor}$ possibilities of input $(x_1, x_{2,1})$ into $r_1 = \lceil \frac{2^{w+\lfloor w/2 \rfloor}}{2^w - 1} \rceil = 2^{\lfloor w/2 \rfloor} + 1$ sets $Q_{1,1}, Q_{1,2}, \ldots, Q_{1,r_1}$, so that $|Q_{1,j}| \leq 2^w - 1 (1 \leq j \leq r_1)$. (The way to divide is arbitrary.) For each $Q_{1,j}$, we build a gate $h_{1,j}$ taking $x_1, x_2$ as inputs.[1] If $(x_1, x_{2,1}) \in Q_{1,j}$, $h_{1,j}$ outputs the index of $(x_1, x_{2,1})$ in $Q_{1,j}$ (a unique $w$-bit integer from 1 to $|Q_{1,j}|$), otherwise the output is $0^w$.

Similarly, we partition the $2^{\lceil w/2 \rceil + w}$ possibilities of input $(x_{2,2}, x_3)$ into $r_2 = \lceil \frac{2^{\lceil w/2 \rceil + w}}{2^w - 1} \rceil = 2^{\lceil w/2 \rceil} + 1$ sets $Q_{2,1}, Q_{2,2}, \ldots, Q_{2,r_2}$, whose size is at most $2^w - 1$. We build a gate $h_{2,j}$ corresponding to $Q_{2,j}$ $(1 \leq j \leq r_2)$, namely the input of $h_{2,j}$ is $x_2, x_3$, and the output is $0^w$ if $(x_{2,2}, x_3) \notin Q_{2,j}$, or the index of $(x_{2,2}, x_3)$ in $Q_{2,j}$ otherwise.

For any input $x_1, x_2, x_3$, only one of outputs of gates $h_{1,1}, h_{1,2}, \ldots, h_{1,r_1}$ is not $0^w$. This is because these gates correspond to sets $Q_{1,1}, Q_{1,2}, \cdots, Q_{1,r_1}$, which partition all the possibilities of $(x_1, x_{2,1})$. Similarly, only one of outputs of $h_{2,1}, h_{2,2}, \cdots, h_{2,r_2}$ is not $0^w$. If we know two gates $h_{1,j_0}$ and $h_{2,l_0}$ do not equal to $0^w$ for some $j_0 \in \{1 \ldots, r_1\}$, $l_0 \in \{1 \ldots, r_2\}$, we can identify the values of $x_1, x_2, x_3$ (and thus $f(x_1, x_2, x_3)$).

For each $h_{1,j}$ $(1 \leq j \leq r_1)$, we construct $r_2$ gates $g_{j,1}, g_{j,2}, \ldots, g_{j,r_2}$ like a chain (we call these gates the *$j$-th chain*, see Figure 1) in the following way:

$$g_{j,1}(h_{1,j}, h_{2,1}) = \begin{cases} h_{1,j} & \text{if } h_{2,1} = 0^w; \\ 0^w & \text{if } h_{1,j} = 0^w; \\ f(x_1, x_2, x_3) & \text{otherwise, } x_1, x_2, x_3 \text{ are known.} \end{cases}$$

[1] Here we use a $k$-input gate as a 2-input one. In later proof we will use a $k$-input gate as a 2-input or 1-input one without claim.

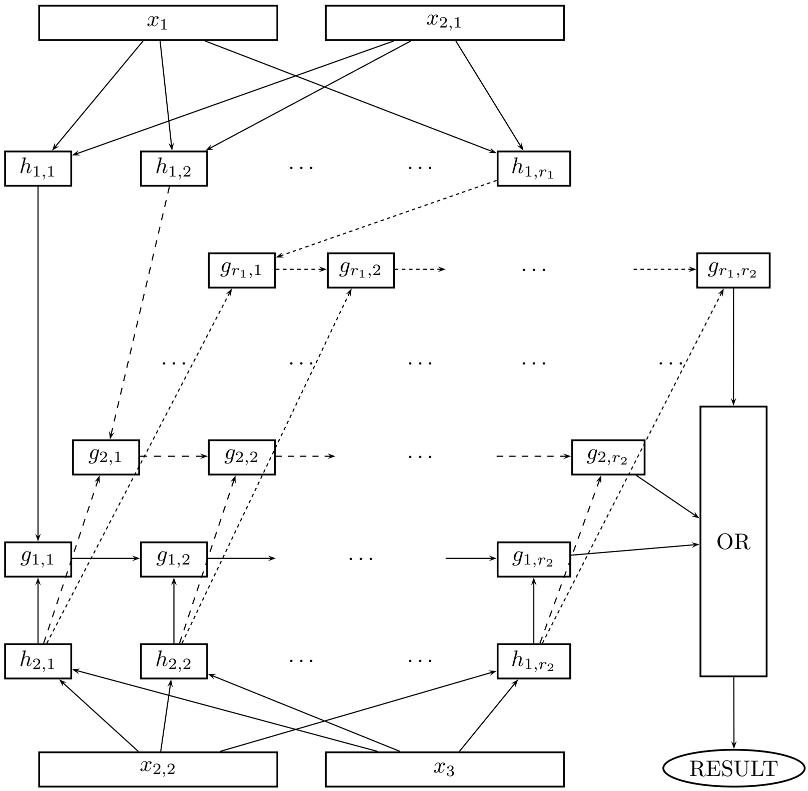


**Fig. 1.** Full construction for $k = 2, n = 3$

For $2 \le l \le r_2$,

$$g_{j,l}(g_{j,l-1}, h_{2,l}) = \begin{cases} g_{j,l-1} & \text{if } h_{2,l} = 0^w; \\ 0^w & \text{if } g_{j,l-1} = 0^w; \\ f(x_1, x_2, x_3) & \text{otherwise, } x_1, x_2, x_3 \text{ are known.} \end{cases}$$

It is easy to see that if $h_{1,j} = 0^w$, the end of the $j$-th chain $g_{j,r_2}$ must output $0^w$. Let's consider the case $h_{1,j} \neq 0^w$. As claimed above, there is only one of $h_{2,1}, h_{2,2}, \ldots, h_{2,r_2}$ nonzero, say $h_{2,l}$. By the construction, all gates in the $j$-th chain other than $g_{j,l}$ outputs its first input word. Thus the first input of $g_{j,l}$ must be $h_{1,j}$, which is nonzero. Therefore $x_1, x_2, x_3$ can be identified at $g_{j,l}$ and the gate is well defined. The output of gates $g_{j,l}, g_{j,l+1}, \ldots, g_{j,r_2}$ are all $f(x_1, x_2, x_3)$.

At last, we compute the disjunction of all gates $g_{j,r_2}$ $(1 \le j \le r_1)$. This takes $(r_1 - 1)$ 2-input OR gates. The final output is the desired function value. (See Figure 1 for the full construction) The total number of gates is $r_1 + r_2 + r_1 \times r_2 + r_1 - 1 = 2 \times (2^{\lfloor w/2 \rfloor} + 1) + 2^{\lceil w/2 \rceil} + (2^{\lceil w/2 \rceil} + 1) \times (2^{\lfloor w/2 \rfloor} + 1) = 2^w + O(2^{w/2}) = (1 + o(1))2^w$. $\square$

### 2.3 Proofs for the General Upper Bound

Now we give the construction for general cases. The following theorem shows an upper bound for any fixed $k$ and $n$ that matches the lower bound stated in Theorem 1.

**Theorem 3.** *For any fixed $n > k \geq 2$, every function $f : (\{0,1\}^w)^n \rightarrow \{0,1\}^w$ can be computed by a word circuit that consists of $(1 + o(1))2^{(n-k)w}$ basic gates.*

*Proof.* The construction contains two steps:

1. We divide the whole input $(b_1, b_2, \ldots, b_{nw})$ into $k$ parts $S_1, S_2, \ldots, S_k$, each contains $\lfloor nw/k \rfloor$ or $\lceil nw/k \rceil$ consecutive bits. The first step is for each part $S_i$ $(1 \leq i \leq k)$, build a word circuit $C_i$ satisfying the following conditions:

   For every $i \in [k]$, let $a_i$ be the number of bits in $S_i$ ($a_i = \lfloor nw/k \rfloor$ or $\lceil nw/k \rceil$). Word circuit $C_i$ takes these $a_i$ bits as input (which is contained in at most $\lceil a_i/w \rceil + 1$ words). We partition the whole input set $\{0,1\}^{a_i}$ into $r_i$ subsets $Q_{i,1}, Q_{i,2}, \ldots, Q_{i,r_i}$ so that $|Q_{i,j}| \leq 2^w - 1$ for all $1 \leq j \leq r_i$. (We define the way to partition and the number $r_i$ precisely later.) The output of circuit $C_i$ contains $r_i$ words $h_{i,1}, h_{i,2}, \ldots, h_{i,r_i}$. For every $1 \leq j \leq r_i$, $h_{i,j} = 0^w$ if the input is not in set $Q_{i,j}$, otherwise $h_{i,j}$ returns the index of the input in set $Q_{i,j}$ (a $w$-bit integer in $[2^w - 1]$).

   Here is the construction of $C_i$: Let $y_1, y_2, \ldots, y_{n_i}$ be the input words of $C_i$, then $n_i \leq \lceil a_i/w \rceil + 1$. Since $n > k$ and $w$ sufficiently large, we have $n_i \geq 2$. Let $u_j$ be the number of bits of word $y_j$ that are in set $S_i$ (the *useful* input bits in word $y_j$ of circuit $C_i$), we have $\sum_{j=1}^{n_i} u_j = a_i$. Since the partition of $S_i$ are consecutive, there are at most two of $\{u_1, u_2, \ldots, u_{n_i}\}$ small than $w$. W.l.o.g. let's assume they are $u_{n_i-1}$ and $u_{n_i}$, and assume $u_{n_i-1} \geq u_{n_i}$. (If there is only one less than $w$, assume it is $u_{n_i}$). Now we partition the $2^{u_1+u_2}$ possibilities of $(y_1, y_2)$ into $m = \lceil \frac{2^{u_1+u_2}}{2^w-1} \rceil$ sets $T_1^1, T_2^1, \ldots, T_m^1$, each contains at most $2^w - 1$ elements. For each set $T_j^1$ $(1 \leq j \leq m)$, we build a gate $z_j^1$ taking $y_1, y_2$ as input. The output of $z_j^1$ is $0^w$ if $(y_1, y_2) \notin T_j^1$, otherwise it is the index of $(y_1, y_2)$ in set $T_j^1$. These $m$ gates are the first layer.
   Next we build the second layer (if $n_i > 2$) on the outputs of the gates $z_1^1, z_2^1, \ldots, z_m^1$ and input $y_3$. Partition the $2^{u_1+u_2+u_3}$ possibilities of $(y_1, y_2, y_3)$ into $m \times 2^{u_3}$ sets: $T_{j,l}^2 = \{(y_1, y_2, y_3) \mid (y_1, y_2) \in T_j^1, y_3 = l\}$ $(1 \leq j \leq m, 1 \leq l \leq 2^{u_3})$. For every $j$, we build $2^{u_3}$ gates taking the output of $z_j^1$ and the input $y_3$ as input, say $z_{j,1}^2, z_{j,2}^2, \ldots, z_{j,2^{u_3}}^2$. The gate $z_{j,l}^2$ $(1 \leq l \leq 2^{u_3})$ outputs $z_j^1$ if $y_3 = l$, or $0^w$ otherwise. (See Figure 2)
   One can easily check $z_{j,l}^2$ outputs $0^w$ if $(y_1, y_2, y_3) \notin T_{j,l}^2$, and it is the index (a unique $w$-bit integer for each $(y_1, y_2, y_3)$) otherwise.
   The third layer (for $n_i > 3$) is build on the outputs of the second layer and $y_4$. For each output of the second layer, say $z_{j,l_1}^2$ $(1 \leq j \leq m, 1 \leq l_1 \leq 2^{u_3})$. We build $2^{u_4}$ gates $z_{j,l_1,l_2}^3$ $(1 \leq l_2 \leq 2^{u_4})$ on $z_{j,l_1}^2$ and $y_4$ in the same way as above: The gate $z_{j,l_1,l_2}^3$ outputs $z_{j,l_1}^2$ if $y_4 = l$, and it is $0^w$ otherwise.

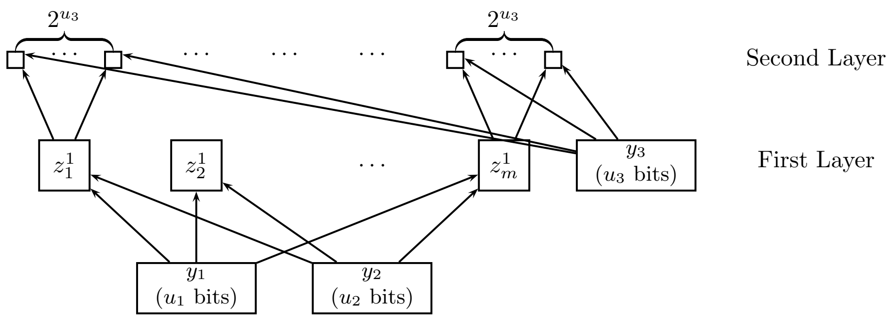


**Fig. 2.** First two layers of $C_i$

Continue this to build $n_i - 1$ layers in total. One can see that the output of the last layer satisfies our requirement of $C_i$ and the corresponding sets $(1 \leq j \leq m, 1 \leq l_1 \leq 2^{u_3}, 1 \leq l_2 \leq 2^{u_4}, \ldots, 1 \leq l_{n_i-2} \leq 2^{u_n})$

$$\begin{aligned} T^{n_i-1}_{j,l_1,l_2,\ldots,l_{n_i-2}} = \{&(y_1, y_2, \ldots, y_n) \mid \\ &(y_1, y_2) \in T^1_j, y_3 = l_1, y_4 = l_2, \ldots, y_n = l_{n_i-2}\} \end{aligned}$$

can be put into a one to one mapping with the $Q_{i,1}, Q_{i,2}, \ldots, Q_{i,r_i}$. Thus

$$\begin{aligned} r_i &= m2^{u_3+u_4+\cdots+u_{n_i}} = \left\lceil \frac{2^{u_1+u_2}}{2^w - 1} \right\rceil 2^{u_3+u_4+\cdots+u_{n_i}} \\ &< 2^{a_i-w} + O(1)2^{u_3+u_4+\cdots+u_{n_i}}. \end{aligned}$$

By $n > k$ and $a_i \geq \lfloor nw/k \rfloor$, one can see $u_1 + u_2 - w \to \infty$ as $w \to \infty$. It follows that $2^{u_3+u_4+\cdots+u_{n_i}}/2^{a_i-w} = 2^{w-u_1-u_2} = o(1)$. Therefore $r_i = (1+o(1))2^{a_i-w}$, and the number of gates used in $C_i$ is $O(r_i) = O(2^{a_i-w}) = O(2^{(n/k-1)w})$.

2. This step is to build circuit on the outputs of $C_1, C_2, \ldots, C_k$. Say the output of $C_1$: $h_{1,1}, h_{1,2}, \ldots, h_{1,r_1}$ are the *master line*, the outputs of $C_2, C_3, \ldots, C_k$ are the *slave lines*. We list the Cartesian production of slave lines as (the order is unimportant) $p_1 = (h_{2,1}, h_{3,1}, \ldots, h_{k,1})$, $p_2 = (h_{2,2}, h_{3,1}, \ldots, h_{k,1})$, $\ldots$, $p_R = (h_{2,r_2}, h_{3,r_3}, \ldots, h_{k,r_k})$, where $R = r_2 r_3 \cdots r_k$.
   For each $h_{1,j}$ in the master line, we construct a list of gates $g_{j,1}, g_{j,2}, \ldots, g_{j,R}$ linking like a chain (we call them the *j-th* chain). $g_{j,1}$ takes the output of $h_{1,j}$ and gates in $p_1$ as input. For $2 \leq l \leq R$, $g_{j,l}$ takes the output of $g_{j,l-1}$ and gates in $p_l$ as input.
   The gate $g_{j,l}$ functions as following:
   (1) Outputs the first input word if one of the other $k-1$ input words is $0^w$.
   (2) Otherwise if the first input word is $0^w$, the gate outputs $0^w$.

(3) If the first input is also nonzero, we claim that $x_1, x_2, \ldots, x_n$ can be determined by the input. As $h_{i,j}$ is nonzero only when the bits in $S_i$ are in $Q_{i,j}$, we see the bits in $S_2, S_3, \ldots, S_k$ can all be determined. Also we see all gates in the $j$-th chain other than $g_{j,l}$ must be the first case, i.e. output its first input word. Thus the first input of $g_{j,l}$ must equal to $h_{1,j}$, and the bits in $S_1$ can also be determined. The gate $g_{j,l}$ outputs $f(x_1, x_2, \ldots, x_n)$ in this case.

Formally, these gates are defined as following:

$$g_{j,1}(h_{1,j}, p_1) = \begin{cases} h_{1,j} & \text{if one of gates in } p_1 \text{ is } 0^w; \\ 0^w & \text{if } h_{1,j} = 0^w; \\ f(x_1, \ldots, x_n) & \text{otherwise, all } x_1, \ldots, x_n \text{ are known.} \end{cases}$$

For $2 \leq l \leq R$,

$$g_{j,l}(g_{j,l-1}, p_l) = \begin{cases} g_{j,l-1} & \text{if one of gates in } p_l \text{ is } 0^w; \\ 0^w & \text{if } g_{j,l-1} = 0^w; \\ f(x_1, \ldots, x_n) & \text{otherwise, all } x_1, \ldots, x_n \text{ are known.} \end{cases}$$

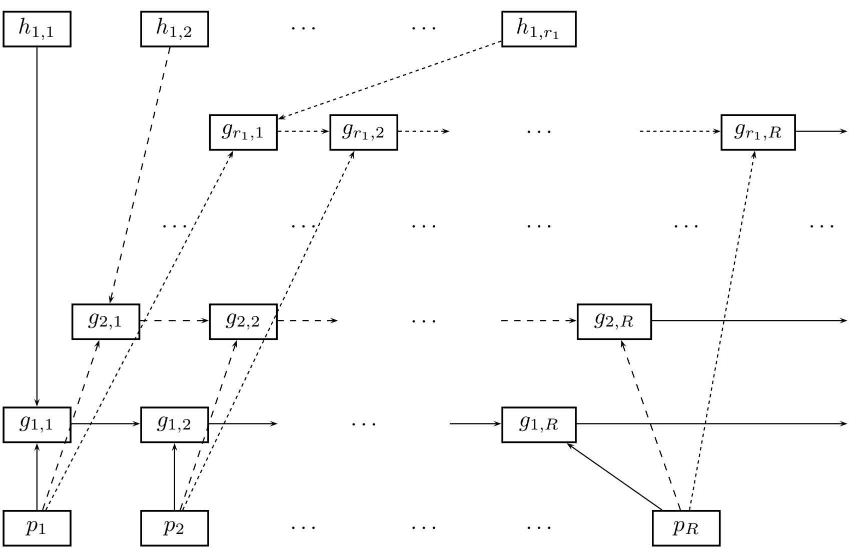


**Fig. 3.** Construction of $g_{j,l}$ $(1 \leq j \leq r_1, 1 \leq l \leq R)$

As we have shown that at most one gate in the $j$-th chain does not transfer its first input forward, one can see the last gate

$$g_{j,R} = \begin{cases} 0^w & \text{if bits in } S_1 \text{ are not in } Q_{1,j}; \\ f(x_1, x_2, \ldots, x_n) & \text{otherwise.} \end{cases}$$

Then we use $(r_1 - 1)$ gates to compute the disjunction of $g_{1,R}, g_{2,R}, \ldots, g_{r_1,R}$. This part of circuit can be any tree consisting of 2-input OR gates that takes

the outputs of $g_{1,R}, g_{2,R}, \ldots, g_{r_1,R}$ as leaves. The root of this tree outputs the value of $f(x_1, x_2, \ldots, x_n)$.
In this step, $r_1 \times R + r_1 - 1 = r_1 r_2 \cdots r_k + r_1 - 1 = (1 + o(1))2^{(n-k)w}$ gates are used in total.

The total number of gates is

$$k \cdot O(2^{(n/k-1)w}) + (1 + o(1))2^{(n-k)w} = (1 + o(1))2^{(n-k)w}.$$

Thus the theorem is proved. □

## 3 Fixed Word Length, Large Number of Input

In this section, we assume that $w, k$ are fixed and $n$ is sufficiently large. First we give the lower bound using similar counting argument as the previous section.

**Theorem 4.** *For any fixed $w \geq 1, k \geq 2$ and sufficiently large number $n$, there exists a function $f : (\{0,1\}^w)^n \to \{0,1\}^w$ so that no circuit consists of less than $(1 - o(1))\frac{2^{nw}}{(k-1)n}$ $k$-input gates computes $f$.*

*Proof.* Similarly as in Theorem 1, by Lemma 1 we have

$$(s + n - 1)^{sk} 2^{sw2^{kw}} / s! \geq 2^{w2^{nw}}.$$

Take the logarithm of both sides, by Stirling's Formula, we have

$$sk \log_2(s + n - 1) + sw2^{kw} - s \log_2 s + s \log_2 e - \frac{1}{2} \log_2 s + O(1) \geq w2^{nw},$$

For sufficiently large $n$, it implies $s \geq (1 - o(1))\frac{2^{nw}}{(k-1)n}$. □

To show that $(1 + o(1))\frac{2^{nw}}{(k-1)n}$ gates are sufficient to compute every function, we consider a function $f : (\{0,1\}^w)^n \to \{0,1\}^w$ as a combination of $w$ one-bit output functions $(\{0,1\}^w)^n \to \{0,1\}$. We first construct word circuits that computes these one-bit functions.[2]

**Lemma 2.** *For any fixed $w \geq 1, k \geq 2$, every function $F : (\{0,1\}^w)^n \to \{0,1\}$ can be computed by a word circuit of size at most $(1 + o(1))\frac{2^{nw}}{(k-1)nw}$.*

*Proof.* The following proof is based on Lupanov's construction for boolean circuits ([3, 4]).

1. First use $nw$ gates to break the input words into bits: For each input word $x_i$ $(1 \leq i \leq n)$, we build $w$ gates taking $x_i$ as input. The $j$-th gate $(1 \leq j \leq w)$ outputs the $j$-th bit of $x_i$. We simply use $b_1, b_2, \ldots, b_{nw}$ to denote the output.

[2] Here the highest bit of the output word is the result, all lower bits are not used. In later proof we will use a $w$-bit word as a single bit without claim.

2. Let $t = \lceil 3\log_2(nw) \rceil$, we use binary AND/OR gates to compute all minterms on $\{b_1, b_2, \ldots, b_t\}$ and $\{b_{t+1}, b_{t+2}, \ldots, b_{nw}\}$. In this step, $O(2^t + 2^{nw-t})$ gates are enough.
3. Divide the $2^t$ possibilities of $(b_1, b_2, \ldots, b_t)$ into $p$ sets $Q_1, Q_2, \ldots, Q_p$, so that each contains at most $s = nw - \lceil 5\log_2(nw) \rceil$ elements and $p = \lceil 2^t/s \rceil$.
   For $1 \le i \le p$ and a 0-1 string $v$ of length $|Q_i|$, we build gates to compute the following value:
   $$F^1_{i,v} = \begin{cases} j\text{-th bit of } v & \text{if } (b_1, b_2, \ldots, b_t) \text{ is the } j\text{-th element of } Q_i; \\ 0 & \text{otherwise.} \end{cases}$$
   For given $i$ and $v$, $F^1_{i,v}$ equals to the disjunction of some minterms on $\{b_1, b_2, \ldots, b_t\}$. Since the length of $v$ is at most $s$, we see each minterm is used as most $2^s$ times. Thus this step takes at most $2^s 2^t$ gates.
4. Every possibility of $(b_1, b_2, \ldots, b_t)$ is corresponding to a tuple $(i, j)$, where $(b_1, b_2, \ldots, b_t)$ is the $j$-th element of $Q_i$. Thus we may write $F(b_1, b_2, \ldots, b_{nw})$ as $F(i, j, b_{t+1}, b_{t+2}, \ldots, b_{nw})$. Define $G(i, b_{t+1}, b_{t+2}, \ldots, b_{nw})$ to be the 0-1 string of length $|Q_i|$, that the $j$-th bit equals to $F(i, j, b_{t+1}, b_{t+2}, \ldots, b_{nw})$ $(1 \le i \le p,\ 1 \le j \le |Q_i|)$.
   For $1 \le i \le p$ and a 0-1 string $v$ of length $|Q_i|$, we build gates to compute the following value:
   $$F^2_{i,v} = \begin{cases} 1 & \text{if } G(i, b_{t+1}, b_{t+2}, \ldots, b_{nw}) = v; \\ 0 & \text{otherwise.} \end{cases}$$
   Similarly, $F^2_{i,v}$ equals to the disjunction of some minterms on the last $nw - t$ bits $\{b_{t+1}, b_{t+2}, \ldots, b_{nw}\}$, say $e_1, e_2, \ldots, e_d$. If $d \le k$, we can compute $F^2_{i,v}$ by one gate. Otherwise we can use $\lceil (d-1)/(k-1) \rceil$ $k$-input OR gates (only the highest bit of every word is used) to compute it: The first gate takes $k$ minterms as input, all other gate takes at most $k-1$ minterms and the output of the previous gate as input. The output of the last gate is the disjunction. (See Figure 4)

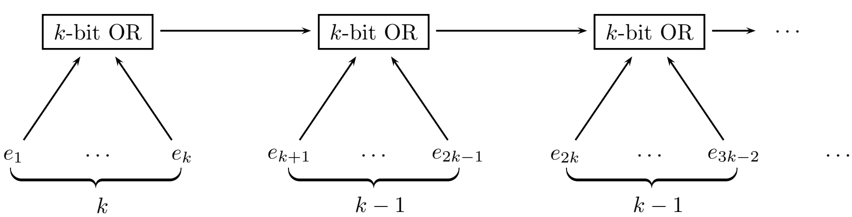


**Fig. 4.** Disjunction of $e_1, e_2, \ldots, e_d$

   Every minterm on $\{b_{t+1}, b_{t+2}, \ldots, b_{nw}\}$ is used at most $p$ times. Among the OR gates there are only $O(p2^s)$ (the number of all possible $(i, v)$) taking less than $k-1$ minterms as input. Thus this step takes at most $p2^{nw-t}/(k-1) + O(p2^s)$ gates.

5. It is not difficult to see $F = \bigvee_i \bigvee_v (F^1_{i,v} \bigwedge F^2_{i,v})$ (see [3, 4] for details). Thus we can compute $F$ by the result of the previous steps. This step takes $O(p2^s)$ gates.

The gates used in total is

$$O(2^t + 2^{nw-t}) + 2^s 2^t + p2^{nw-t}/(k-1) + O(p2^s) = (1 + o(1))\frac{2^{nw}}{(k-1)nw}.$$

Thus the lemma is proved. □

By Lemma 2 we get an upper bound matching the lower bound in Theorem 4 immediately.

**Theorem 5.** *For any fixed $w \geq 1, k \geq 2$, every function $f : (\{0,1\}^w)^n \to \{0,1\}^w$ can be computed by a circuit that consists of $(1 + o(1))\frac{2^{nw}}{(k-1)n}$ k-input gates.*

*Proof.* The function $f$ can be considered as a combination of $w$ one-bit output functions. We construct circuits for each of these functions, then use at most $w - 1 = O(1)$ gates to combine the $w$ outputs into one word. The whole circuit takes $(1 + o(1))\frac{2^{nw}}{(k-1)n}$ gates. □

## Acknowledgments

The third author would like to thank Kristoffer Hansen and Peter Miltersen for helpful discussions.

## References

1. Hansen, K.A., Lachish, O., Miltersen, P.B.: Hilbert's thirteenthv problem and circuit complexity. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 153–162. Springer, Heidelberg (2009)
2. Lupanov, O.B.: A method of circuit synthesis. In: Izvestiya VUZ, Radio_zika, pp. 120–140 (1959)
3. Shannon, C.E.: The synthesis of two-terminal switching circuits. Bell System Technical Journal 28, 59–98 (1949)
4. Wegener, I.: The Complexity of Boolean Functions, pp. 87–92. John Wiley & Sons Ltd./ B. G. Teubner (1987)