The Power of Negative Thinking in Constructing Threshold Circuits for Addition

Thomas Hofmeister*
Lehrstuhl Informatik I
Universität Dortmund
Postfach 50 05 00
D-W-4600 Dortmund 50, Germany
hofmeist@zorro.informatik.uni-dortmund.de

Abstract

It has recently been shown by [BS], [SB], and [AB] that it is possible to add two binary numbers in threshold circuits of depth 2 with polynomial size. In this paper, it is shown that the most significant bit of addition, which is a monotone function, needs exponential size when computed in monotone threshold circuits of depth 2. In fact, it is shown that even \(o(n/\log n)\) negations do not suffice for polynomial size. This is the first example of such a gap for this model.

1 Introduction

The motivation for investigating the power of threshold circuits in computing Boolean functions is twofold. On one hand, threshold circuits are closely related to neural networks and therefore it is interesting to find efficient circuits. On the other hand, the class \(TC^0\) of all functions which have polynomial-size, constant-depth threshold circuits, is one of the weakest classes of functions for which no non-trivial lower bound on explicitly defined functions is known. It is well-known that \(AC^0 \subseteq TC^0 \subseteq NC^1\) and by [FSS] it is known that \(AC^0 \neq TC^0\), but it is not clear whether \(TC^0 \neq NC^1\). For \(NC^1\), the best known lower bounds are still only linear, so trying to prove lower bounds for the perhaps weaker model \(TC^0\) is near at hand. But, unfortunately, even for the model \(TC^0\) where the depth is restricted to \(d\), there is not much known if \(d \geq 3\). For depth 2, [HMPST] show some exponential bound for the "Inner Product". In the meantime, this result has been proved by very different approaches, amongst them the one used in [KW2], who also show bounds for more general depth-2-circuits.

*Supported by DFG Grant No. We 1066 / 2 - 2.
circuits for the addition of two binary numbers. A non-constructive method is given by [BS] and [SB]. As far as other arithmetic functions are concerned, it is known that multiplication is contained in $TC^2$ (this was independently shown in [HHK] and [SB]) and not contained in $TC^2$. It is an open problem whether multiplication can be done by $TC^2$-circuits. Other examples of important functions which are computable in small-depth circuits with polynomial size are division and powering (depth 4) and sorting $n$-bit numbers (depth 3) (see [SBKH]).

In this paper, we finish the characterization of the complexity of addition with respect to $AC^3$ and $TC^0$. It is known that addition is not in $AC^3$, but in $AC^3$ and in $TC^2$. The most significant bit of the sum is a monotone function. This function is known to be in $MonAC^3$ and hence in $MonTC^2$. We show that it is not in $MonTC^2$ and that even $o(n/\log n)$ many negations are not sufficient to compute the function in polynomial size and depth 2. Hence, the most significant bit of an addition is an example for the exponential power of negation in this model. [GHR] recently showed that every function which is computable by one threshold gate of arbitrary size (i.e. with arbitrarily many incoming wires) can be computed in depth 2 and polynomial size. It is a consequence of our result that addition serves as an example that the same statement does not hold for monotone threshold gates and monotone depth-2 threshold circuits.

The proof of our result is done by assigning some parameter to the threshold circuit which depends on the threshold gates from level 1. It can be shown that fixing some variables leads to a large reduction of this parameter. The desired result is then achieved by using induction.

2 Definitions

In this chapter, we try to define all notions used in the sequel. For notions which are not defined in this paper, we refer to [W1].

Our interest in this paper is focused on the complexity of the Boolean function $add_n$, which is defined for $x, y \in \{0, 1\}^n$ as $add_n(x, y) = 1 \iff \sum_{i=0}^{n-1}(x_i + y_i) \cdot 2^i \geq 2^n$. The function $add_n$ thus represents the most significant bit in the sum of the binary numbers $x$ and $y$, and, obviously, is a monotone function. It should be noted that in $(\lor, \land)$-circuits, $add_n$ can be computed in depth 3 and polynomial size by the formula $\bigvee_{i=0}^{n-1}(x_i \land y_i \land \bigvee_{j=0}^{n-1}(x_j \lor y_j))$.

The model which we consider is the model of (unweighted) threshold circuits. A positive (negative, respectively) threshold gate $T^k_{+}$ ($T^k_{-}$, respectively) on $n$ binary variables is a gate which computes one if and only if at least (at most, respectively) $k$ of its inputs are 1. A positive threshold gate will also be called monotone.

Definition 2.1 We use the following notations for sets of gates: (Note that a function is called negative iff it is the negation of a monotone function.)

- $THR$ for the set of all positive and negative threshold gates
- $PTH$ for the set of all positive threshold gates
- $NTH$ for the set of all negative threshold gates
- $MON$ for the set of all gates which compute monotone functions
- $NEG$ for the set of all gates which compute negative functions

Since constant gates are easy to eliminate, we will assume that none of these sets contains a constant function.

It turns out to be helpful to investigate circuits in some normal form. Later on, we show how to transform general circuits into this normal form without changing much the number of negations or the size.

Definition 2.2 A circuit is called levelled whenever we can partition the set of gates into levels such that wires are only between levels $i$ and $i+1$. When $X_1, ..., X_r$ are sets of gates and $f$ is a Boolean function (a sequence of functions, respectively), we denote by $CC(X_1, ..., X_r \parallel f)$ the set of all levelled depth-$r$-circuits which compute the function $f$ and where all gates from level $i$ are taken from the set of gates $X_i$. In these circuits, variables are only given uncomputed and negations on wires are forbidden. When talking about circuits where negations on wires are allowed, in particular on input variables, we denote the corresponding sets of circuits by $CC_{neg}$ instead of $CC$.

Let us now define some parameters for threshold gates and circuits.

Definition 2.3 For a threshold gate $G = T^k_{\pm}$, we define its "boundary distance" by $bounddist(G) := \min(k, n - k + 1)$.

In particular, this means that whenever a threshold gate $G$ computes the constant $c \in \{0, 1\}$, there have to be at least $bounddist(G)$ many inputs of $G$ which compute the constant $c$.

Just as a side note: The boundary distance has been used in characterizing which threshold functions
can be computed in $AC^0$-circuits: It is now well-known that a sequence of threshold functions can be computed by $AC^0$-circuits if and only if the sequence of their boundary distances is polylogarithmic.

**Definition 2.4** The size of a circuit $S$ is the number of its wires and is denoted by $size(S)$. By $fan_i(S)$ we denote the maximum fan-in of gates on level $i$. Since we are investigating circuits for addition, it makes sense to say that the number of variables of the circuit is $n$ whenever the input variables are $x_{n-1}, y_{n-1}, \ldots, x_0, y_0$.

**Definition 2.5** If $e$ is a wire in a circuit, we denote by source($e$) the input gate of this wire. Depending on whether source($e$) is a monotone or a negative gate, we will call the wire positive or negative.

**Definition 2.6** When $S \in CC(PTH \cup NEG, PTH || adda_0)$, we denote by pos($S$) the number of positive wires between level 1 and level 2 and, analogously, by neg($S$) the number of negative wires. It should be noted that $pos(S) + neg(S) = fan_2(S)$.

### 3 A Lower Bound on the Number of Negations

In this chapter, it is shown that circuits of the form $CC(PTH \cup NEG, PTH || adda_0)$ with polynomial size cannot exist when the number of negative wires between level 1 and level 2 is $o(n/\log n)$. This of course shows in particular that there are no monotone depth-2-threshold circuits of polynomial size for the most significant bit of addition; in fact, in this case, an exponential lower bound can be shown to hold. In the next chapter it is shown how we can handle general circuits from $CC_{neg}(THR, THR || adda_0)$.

At first, let us note the following trivial property of the investigated circuits:

**Lemma 3.1** Let $S \in CC(PTH \cup NEG, PTH || adda_0)$ be given. Then $pos(S) \geq neg(S) + 1$.

**Proof:** On the input where all variables are zero, exactly $neg(S)$ many wires between level 1 and level 2 compute one. On the input where all variables are one, exactly $pos(S)$ many wires compute one. Since the output gate is a monotone threshold gate in a circuit computing the most significant bit of addition, it has to compute zero on the all-zero-input and one on the all-one-input. It follows that $pos(S) \geq neg(S) + 1$. □

We prove the main theorem by defining some particular parameter for threshold circuits and investigating how this parameter changes when we fix some of the variables.

**Definition 3.2** Assume we are given a circuit $S \in CC(PTH \cup NEG, PTH || adda_0)$. For a wire $e$ between a gate $G$ on level 1 and the output gate on level 2 within this circuit, we define the number $value(e)$ to be $bouddist(G) + 1$, if $G$ is a monotone threshold gate and $value(e) = 1$ else. If $e_1, \ldots, e_{fan_1(S)}$ is the list of all such wires in circuit $S$, we define the "product value" of $S$ by the number $product(S) := \prod_{e \in S} value(e)$.

Roughly speaking, we multiply the boundary distances of the monotone threshold gates on level 1. The term "+1" in the definition of $value(e)$ has been chosen in such a way that gates of the form $T_{\geq 1}$ contribute a factor of more than 1.

Our proof will proceed inductively by setting some of the variables to a constant and by then removing constant wires and gates by correcting the gates which rely on these. It should be noted that by fixing some of the variables, monotone threshold gates are reduced to monotone threshold gates and negative gates are reduced to negative gates.

Proceeding inductively, we have a choice of two possibilities, either we can fix $x_{n-1} := 1$ and $y_{n-1} := 0$ and get a circuit $S_1$ for $adda_0$ or we fix $x_{n-1} := 0$ and $y_{n-1} := 1$ and also get a circuit $S_2$ for $adda_0$. For our purposes, it is better to consider that circuit of these two which has the smaller product value. We will denote this circuit by $Reduce(S)$.

Now we come to the most important lemma which will repeatedly be applied in the rest of this chapter. This lemma tells us that fixing the right variables yields a circuit where the product value is decreased by a certain amount.

**Lemma 3.3** Let $S \in CC(PTH \cup NEG, PTH || adda_0)$. Then:

$$product(Reduce(S)) \leq \left(\frac{2}{3}\right)^{\frac{fan(S) + \text{size}(S)}{4}} product(S).$$

**Proof:** Assume that the output gate is of the form $T_{\geq m}^{fan(S)}$. Let $A$ denote the input where $x_{n-1} = 1, y_{n-1} = 1$ and all other variables are zero. On this input, the circuit has to compute one and therefore at least $m$ wires between level 1 and 2 must compute one. First, let us investigate the case that $m \geq \frac{fan(S)}{2}$.

Then, on input $A$, at least $\frac{fan(S)}{2}$ many wires must compute one. Since there are exactly $neg(S)$ many
negative wires, there must be at least \(\frac{\text{pos}(S)}{2} - \text{neg}(S)\) many positive wires which compute one. Let us call these wires \(e_1, \ldots, e_r\).

It is our goal to diminish many boundary distances simultaneously by fixing variables. We can either fix \(x_{n-1} := 0, y_{n-1} := 1\) or \(x_{n-1} := 1, y_{n-1} := 0\). For some of the gates \(\text{source}(e_i)\), one of the two fixings is better than the other one. We write down for each gate which fixing is the better one and finally choose the one which is good for the majority of the gates.

Let us assume for a moment that we find for each \(\text{source}(e_i)\) a fixing which reduces the number \(\text{value}(e_i)\) by a factor of \(q < 1\).

The parameter \(\text{product}()\) of the circuit contains these \(\text{value}(e_i)\) as factors and, furthermore, the values of some other wires. We can forget about the effect of the fixing upon the other wires, since the values of these cannot be increased. (Remember that negative gates are reduced to negative wires and the boundary distances of monotone threshold gates are only decreased.)

Hence, according to the pigeonhole principle, it is clear that one of the two fixings must decrease the parameter \(\text{product}()\) of the circuit by a factor of at least \(q^2 \leq q^{\text{bestproduct}(S) - \text{bestproduct}(S)} = q^{\text{bestproduct}(S) - \text{bestproduct}(S)}\). This proves the claim of the lemma if we succeed in showing that for every monotone threshold gate \(\text{source}(e_i)\) one of the two fixings reduces the value of the wire \(e_i\) by at least \(q = 2/3\).

The rest of the proof is devoted to this. We investigate a monotone threshold gate \(G\) which computes one on input \(A\). Let this gate be of the form \(T^N_{\leq k}\). At least \(k\) of \(G\)'s input wires must compute one. If one denotes by \(#x\) \((\#y)\), respectively) the number of wires which enter \(G\) and are labelled with the variable \(x_{n-1}\) \((y_{n-1}\), respectively\)), it must hold that \(#x + \#y \geq k\). We can of course assume without loss of generality that \(#x \geq \#y\) and find that \(#x \geq \frac{k}{3}\) \(\text{bounddist}(G)\).

We have two cases:

1. \(\text{bounddist}(G) = k\). We fix \(x_{n-1} := 1, y_{n-1} := 0\) whereby the gate \(G\) is transformed into a gate \(G'\) with \(G' = T^{N-k}_{\geq \#x-\#y}\) and thus \(\text{bounddist}(G') \leq k - \#x \leq \frac{\text{bounddist}(G)}{2}\).

2. \(\text{bounddist}(G) = N - k + 1\). We fix \(x_{n-1} := 0, y_{n-1} := 1\) whereby the gate \(G\) is transformed into a gate \(G'\) with \(G' = T^{N-k}_{\geq \#x-\#y}\) and thus \(\text{bounddist}(G') \leq N - k + 1 - \#x \leq \frac{\text{bounddist}(G)}{2}\).

Hence one of the two fixings at least halves the boundary distance of \(G\), the number \(\text{value}(e_i) = \text{bounddist}(G) + 1\) is thus decreased by a factor of at least \(2/3\). Finally, since \(\text{Reduce}(S)\) was chosen to be the circuit with the smaller value of the parameter \(\text{product}\), we are done with the proof for the case \(m \geq \frac{\text{bestproduct}(S)}{2}\). The case \(m < \frac{\text{bestproduct}(S)}{2}\) can be treated in an analogous fashion, by investigating the behaviour of the circuit on the input \(B\) where \(x_{n-1} = 0, y_{n-1} = 0\) and all other variables are one.

A first result which follows from lemma 3.3 is the following:

**Corollary 3.4** Let \(S_n \in \text{CC}(\text{PTH} \cup \text{NEG}, \text{PTH} \parallel \text{addn})\). Then \(\text{pos}(S_n) \geq c* \cdot n/\log \text{size}(S_n)\) where \(c* = 1/4 \cdot \log 3/2\).

**Proof:** The proof works by induction. Let us define \(\text{bestproduct}(n) := \min\{\text{product}(S) | S \in \text{CC}(\text{PTH} \cup \text{NEG}, \text{PTH} \parallel \text{addn})\}\). Using the property stated in Lemma 3.1 and inserting it into Lemma 3.3, we conclude that \(\text{bestproduct}(n) \geq \left(\frac{3}{4}\right)^{1/4} \cdot \text{bestproduct}(n-1)\).

Obviously, \(\text{bestproduct}(1) \geq 2\) and it follows that \(\text{bestproduct}(n) \geq \left(\frac{3}{4}\right)^{1/4} n\). On the other hand, \(\text{product}(S_n)\) is by definition bounded above by \(\text{size}(S_n)^{\text{pos}(S_n)}\). We conclude that \(\text{size}(S_n)^{\text{pos}(S_n)} \geq \left(\frac{3}{4}\right)^{1/4} n\). Taking logarithms yields the result.

If we are interested in polynomial-size circuits, the following can be shown to hold:

**Corollary 3.5** A sequence \((S_n)\) of circuits from \(\text{CC}(\text{PTH} \cup \text{NEG}, \text{PTH} \parallel \text{addn})\) which has polynomial size, satisfies \(\text{pos}(S_n) = \Omega(n/\log n)\).

**Proof:** Corollary 3.4 can only be satisfied for \(\text{pos}(S_n) = \Omega(n/\log n)\).

The following corollary is dual to the above. It is listed here since it is needed later.

**Corollary 3.6** A sequence \((S_n)\) of circuits from \(\text{CC}(\text{NTH} \cup \text{MON}, \text{NTH} \parallel \text{addn})\) which has polynomial size, satisfies \(\text{neg}(S_n) = \Omega(n/\log n)\).

**Proof:** Whenever the output gate is of the form \(T^N_{\leq k}\), we replace it by the gate \(T^N_{\geq N-k}\) and replace all gates on level 1 by their negated gates. This yields a circuit which computes the same function and which is of the form used in Corollary 3.5. Positive wires are thereby transformed into negative wires and vice versa.

**Remark:** By defining the product a little bit differently, one can show that even the number of gates on level 1 must be \(\Omega(n/\log n)\). This also holds for polynomial-size monotone threshold circuits even if arbitrary depth is allowed. As a contrast, note that the
most significant bit of addition can in fact be computed by a single threshold gate if superpolynomial size is allowed. It should furthermore be noted that results for general threshold circuits similar to Corollary 3.5 can easily be obtained by using a communication complexity argument.

In order to elucidate the idea behind the final proof for the lower bound on negations, we take a look at a special case where the proof is more straightforward. From now on, let \( c_i \) denote some suitably chosen positive constants.

**Theorem 3.7** If \((S_n)\) is a sequence of circuits from \(CC(PTH \cup NEG, PTH \parallel add_n)\) which has the property that \(Reduce(S_n) = S_{n-1}\), then the following holds: If \(neg(S_n) = o(pos(S_n))\), then \(S_n\) has superpolynomial size.

**Proof:** Assume that \(S_n\) has polynomial size and choose a number \( k \) such that \(pos(S_n) = \Omega(n^k)\). (According to Corollary 3.5, we find such a \( k \) with, say, \( k > 0.99\).)

Due to the assumption that \(neg(S_n) = o(pos(S_n))\), it holds that \(pos(S_n) - neg(S_n) = \Omega(n^k)\). With the help of Lemma 3.3, we conclude that there is a constant \( c > 0 \) such that \(product(S_n) \geq (3/2)c^{n^k} \cdot product(S_{n-1})\).

This gives \(product(S_n) \geq (3/2)c^{n^k} + c(n-1)^k + \ldots \geq t^{n^{k+1}}\) for some suitable constant \( t > 1 \). On the other hand, we know that \(product(S_n)\) is bounded above by \(size(S_n)^{pos(S_n)}\) and find: \(size(S_n)^{pos(S_n)} \geq n^{k+1}\).

Under the assumption that the size is bounded by a polynomial, it follows that \(pos(S_n) \geq c_1 \cdot n^{k+1} / \log n \geq c_2 \cdot n^{k+0.99}\). Thus, we have shown that under the assumption \(pos(S_n) = \Omega(n^k)\) it follows that \(pos(S_n)\) is even in \(\Omega(n^{k+0.99})\). It follows that \(pos(S_n)\) is superpolynomial. \(\square\)

We have thus seen that for very special polynomial-size circuits the number of negative wires cannot be much smaller than the number of positive wires. For general circuits, we can only prove a less strong statement.

The difficulty with sequences of general circuits is that one does not know the value \(pos(Reduce(S_n))\) in advance. Thus, the proof of the following result must examine more carefully the behaviour of the value \(pos\).

**Theorem 3.8** Every sequence \((S_n)\) of circuits from \(CC(PTH \cup NEG, PTH \parallel add_n)\) with \(neg(S_n) = o(n/\log size(S_n))\) satisfies \(size(S_n) = \Omega(\sqrt n)\).

**Proof:** Let \((S_n)\) be a sequence of circuits with \(neg(S_n) = o(n/\log size(S_n))\). We investigate the behaviour of \(pos\) during the first \([n/2]\) \(Reduce\) steps. For this purpose, define \(R_n := S_n\) and \(R_k := Reduce(R_{k+1})\) for \([n/2] \leq i \leq n - 1\). Observe that due to the definition of the \(Reduce\) step, \(pos(R_n) \geq pos(R_{n-1}) \geq \ldots \geq pos(R_{[n/2]})\), \(neg(R_n) \geq neg(R_{n-1}) \geq \ldots \geq neg(R_{[n/2]}\) and \(size(R_n) \geq size(R_{n-1}) \ldots \geq size(R_{[n/2]})\).

We apply Corollary 3.4 to \([n/2]\) to find that \(pos(R_{[n/2]}) \geq c \cdot \log size(R_{[n/2]})) = \Omega(n/\log size(S_n))\). Since \(neg(R_n) = o(n/\log size(S_n))\), there is an \( n_0 \) such that for all \( n \geq n_0\), \(neg(R_n) \leq \frac{1}{2} \cdot pos(R_{[n/2]})\) and, in particular, \(neg(R_n) \leq \frac{1}{2} \cdot pos(R_j)\) for all \( j = [n/2], \ldots, n\). We assume for the rest of this proof that \(n \geq n_0\).

Our aim is to determine the value of \(pos(R_k)\) for arbitrary \([n/2] \leq j \leq k \leq n\). Lemma 3.3 can be applied for \(j \leq i \leq k\) in the following way:

\[
product(R_j) \geq (3/2)^{pos(R_j)-neg(R_j)} \cdot product(R_{j-1}) \geq (3/2)^{pos(R_j)-neg(R_j)} \cdot product(R_{j-1})
\]

Arguing inductively, we conclude that \(product(R_k) \geq (3/2)^{pos(R_k)-neg(R_k)} \cdot product(R_{k-1}) \geq (3/2)^{pos(R_k)-neg(R_k)} \cdot product(R_{k-1})\).

On the other hand, \(product(R_k)\) is bounded by \(size(R_k)^{pos(R_k)} \leq (size(R_n)^{pos(R_n)})\). Comparing both bounds yields:

\[
\frac{pos(R_k)}{pos(R_j)} \geq c_2 \cdot \frac{(k-j)}{\log size(R_n)} = q_{k-j}.
\]

We then know that for all suitable \( d, pos(S_n) = pos(R_n) \geq pos(R_{n-d}) \cdot q_d\). Iterating this inequality \( k \) times (where \( n - d \geq [n/2]\) must be fulfilled) yields the bound \(pos(S_n) \geq pos(R_{n-d}) \cdot q_d^k \geq q_d^k\).

Let us choose \( d := [2 \cdot \log size(S_n)/c_2], k = [n/2d]\). This gives \(q_d \geq 2\), hence \(pos(S_n) \geq t^k\). Inserting the value for \( k \), we get \(log size(S_n) \geq log pos(S_n) \geq k \geq \frac{n}{\log size(S_n)/c_2} - 1\). This inequality shows \(log size(S_n) = \Omega(\sqrt n)\) which is the statement of the theorem. \(\square\)

Theorem 3.8 can be applied to obtain the following two bounds.

**Corollary 3.9** There are no threshold circuits \((S_n) \in CC(PTH \cup NEG, PTH \parallel add_n)\) which simultaneously have \(neg(S_n) = o(\sqrt n)\) and size \(\Omega(\sqrt n)\).

**Proof:** Every such circuit would satisfy the conditions of Theorem 3.8. It would follow that \(size(S_n) \geq 2\Omega(\sqrt n)\), a contradiction. \(\square\)
In particular, the above theorem can be seen as a bound for monotone circuits.

If we are only interested in the class of polynomial-size circuits, then the following is of interest:

**Corollary 3.10** There are no polynomial-size circuits from $CC(PTH \cup NEG, PTH \parallel add_n)$ which have $o(n/\log n)$ many negative wires.

**Proof:** Every polynomial-size circuit with $\text{neg}(S_n) = o(n/\log n)$ fulfills the conditions of Theorem 3.8, hence it must have size $2^{\Omega(\sqrt{n})}$, a contradiction. \qed

Note that the above corollary only holds because there are no negations on the wires. The next chapter shows how to deal with those.

### 4 Dealing with general circuits

Up to now, we have shown a lower bound on the number of "negative wires" in circuits of the form $CC(PTH \cup NEG, PTH \parallel add_n)$. But, we are interested in a bound for the "more general" unlevelled circuits of depth 2, where negations are allowed in arbitrary places, in particular on the input variables. In order to define what we understand by the number of negations, we note that we only allow gates from $PTH$ and count the number of negations on the wires. We first show that these circuits can be transformed into circuits of the kind we investigated up to now. The circuits can easily be levelled by a standard method. The number of negations is not changed and the size is at most doubled.

Now we have to deal with the negations. At first, we ensure that no gate $G$ enters the output gate by unnegated as well as by negated wires. This can be done by leaving out some of the wires and adjusting the output gate. The size of the circuit and the number of negations is at most decreased. In case there is a negation on the output wire, we remove it and replace the output gate by a negative threshold gate. In case there are negations on wires between level 1 and level 2, we remove them by changing their input gates into negative threshold gates. (This is possible since we have guaranteed that no gate has outgoing negated and unnegated wires.) It is easy to see that the number of negative wires in the resulting circuit is not larger than the number of negations in the given circuit.

The only disturbing thing remaining are possible negations on the input variables and the fact that the output gate might be negative. The negations on input wires are removed by fixing the corresponding variables, the possibly negative output gate is also dealt with in an easy fashion:

**Theorem 4.1** Every sequence of circuits $(S_n)$ from $CC_{neg}(PTH, PTH \parallel add_n)$, which only has $o(n/\log n)$ many negations, requires superpolynomial size.

**Proof:** Assume that there are polynomial-size circuits $S_n$ and that they are already transformed according to the above comments, with possibly some negations left on wires leaving variables.

We remove these negations as follows: Let $I(n) := \{i\mid\text{there is a negation on a wire leaving } x_i \text{ or } y_i\}$. We then fix for all $i \in I(n)$ the variable $x_i$ by 1 and $y_i$ by 0. It is easy to see that the resulting circuits compute the most significant bit of addition on the remaining variables. They can thus be seen as circuits from $CC(THR, THR \parallel add_n \parallel I(n))$. Since $|I(n)| = o(n/\log n)$, it follows for big enough $n$ that $N_n := n - |I(n)| \geq n/2$.

By applying *Reduce* in a suitable manner, we extend these circuits to a sequence $P_n$ from $CC(THR, THR \parallel add_n \parallel I(n))$. Since $N_n$ is dense enough, this can be done in such a way that $P_n$ has polynomial size and only $o(n/\log n)$ many negative wires. By Corollary 3.6, we find that there is an $n_0$ such that all circuits $P_n$ with $n > n_0$ have a positive threshold gate on level 2, since otherwise the number of negative wires would be in $\Omega(n/\log n)$ and not in $o(n/\log n)$. As a consequence, we can use Corollary 3.10 in order to deduce that $P_n$ cannot have polynomial size. This is a contradiction. \qed

**Acknowledgements**

My thanks go to Johan Håstad for pointing out to me that a superpolynomial bound in an earlier version of this paper could in fact be improved to an exponential bound.

**References**


