On the Negation-Limited Circuit Complexity of Merging

Kazuyuki Amano[1], Akira Maruoka[1], and Jun Tarui[2]

[1] Graduate School of Information Sciences, Tohoku University
Sendai 980–8579, Japan
{ama|maruoka}@ecei.tohoku.ac.jp
[2] Department of Communications and Systems
University of Electro-Communications
Chofu, Tokyo 182–8585, Japan
jun@sw.cas.uec.ac.jp

Abstract A negation-limited circuit is a combinational circuit that consists of AND, OR gates and a limited number of NOT gates. In this paper, we investigate the complexity of negation-limited circuits. The (n, n) merging function is a function that merges two presorted binary sequences $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$ into a sequence $z_1 \leq \cdots \leq z_{2n}$. We prove that the size complexity of the (n, n) merging function with $t = (\log_2 \log_2 n - a)$ NOT gates is $\Theta(2^a n)$.

1 Introduction

To derive a strong lower bound on the size of a combinational circuit, that consists of AND, OR and NOT gates, for an explicitly defined function is one of the most challenging problems in theoretical computer science. But so far, only linear lower bounds have been known (See [7] for an excellent survey on the complexity of Boolean functions). In sharp contrast, there has been substantial progress in obtaining strong lower bounds on the size of a monotone circuit, that consists of AND and OR gates. Razborov[13] obtained superpolynomial lower bounds on the size of a monotone circuit for the clique function. Shortly thereafter, Andreev[1] and Alon and Boppana[3] obtained exponential lower bounds on the size of a monotone circuit for the clique function and for some other functions. In short, we have strong lower bounds on circuit size when we restrict ourselves to circuits without NOT gates, whereas we have very weak lower bounds when we consider circuits with an arbitrary number of NOT gates. So it is natural to consider the size of a circuit with a limited number of NOT gates, which we call the negation-limited circuit complexity.

There has been obtained many interesting results in this line of research. Markov[10] proved that $r = \lceil \log_2(n+1) \rceil$ NOT gates are sufficient to compute any collection of Boolean functions on n variables and that there exists a function that requires r NOT gates to compute. Fisher[8] showed that restricting the number of available NOT gates to r entails only a polynomial blowup in circuit size. Beals, Nishino and Tanaka[6] strengthen this result by showing that, for any Boolean function f, the size of a smallest circuit with at most r NOT gates

computing f is at most $2\text{size}(f) + O(n \log n)$, where $\text{size}(f)$ is the size of a smallest circuit with an arbitrary number of NOT gates computing f. Recently, Amano and Maruoka[4] proved that the clique function is not computable by a polynomial size circuit with at most $(1/6) \log \log n$ NOT gates.

Tardos[11] proved that there exists a function that can be computed by a circuit of polynomial size whereas it requires monotone circuits of exponential size. In contrast, Berkowitz[5] and Beals, Nishino and Tanaka[6] proved that NOT gates are almost powerless for slice functions. A function f is a *slice* function if $f(x_1, \ldots, x_n) = 0$ for any $\sum_{i=1}^{n} x_i < k$ and $f(x_1, \ldots, x_n) = 1$ for any $\sum_{i=1}^{n} x_i > k$ for some $0 \leq k \leq n$ (for input contains exactly k ones, the output of f is not constrained). They proved that for any slice function f, the monotone circuit complexity of f is at most $2\text{size}(f) + O(n \log n)$ where $\text{size}(f)$ is the circuit complexity of f.

The effect of NOT gates on circuit complexity looks like a mystery and better understanding of it is necessary to obtain a good lower bound for circuit complexity.

In this paper, we consider the problem : *how fast a complexity of a function deceases when the number of available NOT gates in a circuit increases?* That is one of the most interesting problems on negation-limited circuit complexity.

The merging function is a Boolean function that takes two presorted binary sequences each of length n as inputs, and merges them into a sorted sequence of length $2n$. It was shown that the merging function has circuit complexity $\Theta(n)$ and has monotone circuit complexity $\Theta(n \log n)$ (See e.g., [7], [9, Chap.5.3.4]). In this paper, we prove the nontrivial lower and upper bounds on the negation-limited circuit complexity of merging. Remarkably, both bounds we obtained are matched up to a constant factor for *any* values of the number of available NOT gates. Precisely, we prove

Theorem 1 (Main) The size complexity for merging two presorted binary sequences each of length n with $t = (\log_2 \log_2 n - a)$ NOT gates is $\Theta(2^a n)$.

Roughly speaking, this theorem says that the size of a smallest circuit for the merging function is halved when the number of available NOT gates increases one, and if we can use $\log_2 \log_2 n$ NOT gates then the merging function can be computed by a circuit of linear size. In addition, the proof of the upper bound is constructive, so, for any number t of available NOT gates, we give a circuit for merging with t NOT gates whose size is optimal up to a constant factor.

The paper is organized as follows: First, in Section 2 we give some basic notations and definitions. In Section 3, we prove the lower bound of our main theorem by using the technique of partial assignment to the input variables. In Section 4, we prove the upper bound of our main theorem by giving the construction of a circuit for merging. Finally, we provide some open problems related our work in Section 5.

2 Preliminaries

A *circuit* is a combinational circuit that consists of AND gates of fan-in two, OR gates of fan-in two and NOT gates. In particular, a circuit without NOT gates is called *monotone*. A Boolean function f is monotone if $f(x_1,\ldots,x_n) \leq f(y_1,\ldots,y_n)$ whenever $x_i \leq y_i$ for all i. Monotone functions are exactly those computed by monotone circuits. The *size* of a circuit C is the number of gates in the circuit C.

Let F be a collection of Boolean functions $f_1,\ldots,f_m$. The *circuit complexity* (*monotone circuit complexity*, resp.) of F is the size of a smallest circuit (monotone circuit, resp.) that computes F, i.e., that outputs $f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n)$ for the inputs $x_1,\ldots,x_n$. Following Beals, Nishino and Tanaka[6], we call a circuit including at most t NOT gates a *t-circuit*.

The (n,n) *merging function*, denoted by MERGE(n,n), is a collection of functions that merges two presorted binary sequences $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$ into a sequence $z_1 \leq z_2 \leq \cdots \leq z_{2n}$, i.e., $z_i = 1$ if and only if the total number of 1's in the input sequences is not less than $2n-i+1$, for all i. All logarithms in this paper are base two.

Now we restate our main theorem.

Theorem 1 (Main) The size complexity of MERGE(n,n) with $t = (\log_2 \log_2 n - a)$ NOT gates is $\Theta(2^a n)$.

3 Lower Bound

In this section, we prove the lower bound of Theorem 1, that is, the size of any t-circuit, where $t = (\log_2 \log_2 n - a)$, computing MERGE$(n,n)$ is $\Omega(2^a n)$.

Proof: The outline of the proof is as follows: First, we prove that if C is a t-circuit for MERGE(n,n) then C has a monotone subcircuit which computes MERGE$(n/2^t, n/2^t)$. Combining this and the fact that the monotone circuit complexity of MERGE$(n/2^t, n/2^t)$ is $\Omega((n/2^t)\log(n/2^t)) = \Omega(2^a n)$ to obtain the lower bound for the size of C.

Let C be an optimal t-circuit for MERGE(n,n). Without loss of generality, we may assume that $n = 2^r$ for some positive integer r. Let g_i, for $1 \leq i \leq t$, be the function computed by the input of i-th NOT gates counting from the inputs to the outputs. A *minterm* of a function f is a minimal assignment to some subset of the variables which determined f to be 1.

Let $X = \{x_1,\ldots,x_n\}$ and $Y = \{y_1,\ldots,y_n\}$ be the sets of input variables of MERGE(n,n). Let X^h (respectively, Y^h) be the second half of X (respectively, the second half of Y), i.e., $X^h = \{x_{n/2+1},\ldots,x_n\}$ and $Y^h = \{y_{n/2+1},\ldots,y_n\}$.

Now we define a partial assignment p_1 to $X \cup Y$ that makes g_1 a constant function. Note that g_1 is a monotone since g_1 is the input of the lowest NOT gates in C. If g_1 has a minterm such that all the variables appeared in it are contained in $X^h \cup Y^h$, then let p_1 be the partial assignment that assigns the

value 1 to all the variables in $X^h \cup Y^h$. Otherwise, if g_1 does not have such minterms, then let p_1 be the partial assignment that assigns the value 0 to all the variables in $(X - X^h) \cup (Y - Y^h)$. For a Boolean function f and a partial assignment p, we denote the induced function of f under p by $f|_p$. It is clear that $g_1|_{p_1}$ is the constant 1 for the first case and is the constant 0 for the second case. It is also clear that $g_2|_{p_1}$ is a monotone.

Next, we define a partial assignment p_2 from p_1 by adding the assignments to half of the variables unassigned by p_1. Let $X_1 \subseteq X$ and $Y_1 \subseteq Y$ be the sets of all variables unassigned by p_1. Let X_1^h be the second half of X_1 and Y_1^h be the second half of Y_1. Note that $|X_1| = |Y_1| = n/2$ and $|X_1^h| = |Y_1^h| = n/4$. If $g_2|_{p_1}$ has a minterm such that all the variables appeared in it are contained in $X_1^h \cup Y_1^h$, then let p_2 be the partial assignment obtained from p_1 by adding the assignments of the value 1 to all the variables in $X_1^h \cup Y_1^h$. Otherwise, let p_2 be the partial assignment obtained from p_1 by adding the assignments of the value 0 to all the variables in $(X_1 - X_1^h) \cup (Y_1 - Y_1^h)$. It is clear that both $g_1|_{p_2}$ and $g_2|_{p_2}$ are constant functions and $g_3|_{p_2}$ is a monotone.

By continuing above procedure recursively for all the NOT gates in C, we obtain the partial assignment p_t such that (i) p_t leaves two sets of $n/2^t$ consecutive variables $\{x_{i+1}, x_{i+2}, \ldots, x_{i+n/2^t}\}$ in X and $\{y_{i+1}, y_{i+2}, \ldots, y_{i+n/2^t}\}$ in Y, for some i, (ii) p_t assigns the value 0 to $x_1, \ldots, x_i, y_1, \ldots, y_i$ and the value 1 to $x_{i+n/2^t+1}, \ldots, x_n, y_{i+n/2^t+1}, \ldots, y_n$, and (iii) p_t makes the inputs of all the NOT gates in C a constant.

Under p_t we obtain the induced circuit of C by eliminating all gates whose values become determined. By (i) and (ii), the sequence of the outputs $z_{2i+1}, \ldots, z_{2i+2n/2^t}$ of the induced circuit is equal to the merged sequence of $x_{i+1}, \ldots, x_{i+n/2^t}$ and $y_{i+1}, \ldots, y_{i+n/2^t}$. By (iii), the induced circuit has no NOT gates.

Thus the circuit C "contains" a monotone circuit for MERGE$(n/2^t, n/2^t)$ and the size of C is lower bounded by the monotone circuit complexity of MERGE$(n/2^t, n/2^t)$. Since the monotone circuit complexity of MERGE(N, N) is known to be $\Omega(N \log N)$ (See e.g., [7],[9]), the lower bound is $\Omega((n/2^t)\log(n/2^t)) = \Omega((n/2^t)\log n) = \Omega(2^a n)$ as desired.

□

4 Upper Bound

In this section, we prove the upper bound of Theorem 1, that is, there exists a t-circuit, where $t = (\log_2 \log_2 n - a)$, computing MERGE$(n, n)$ of size $O(2^a n)$.

Proof: We describe below how to construct a circuit of size $O(n)$ with $\log \log n$ NOT gates for bitonic sorting. We say that a binary sequence $z_1, \ldots, z_{2n}$ is *bitonic* if $z_1 \geq \cdots \geq z_k \leq \cdots \leq z_{2n}$ for some k, $1 \leq k \leq 2n$. A circuit that sorts for bitonic sequences can obviously compute MERGE(n, n) since a sequence $x_n, \ldots, x_1, y_1, \ldots, y_n$ is bitonic when $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$. When only $(\log \log n - a)$ NOT gates are available, exactly the same construction yields a circuit of size $O(2^a n)$.

Let $x_1, ..., x_{2n}$ be a bitonic sequence. The first layer of the standard bitonic sorter (with usual min/max gates expressed as AND/OR gates) produces the smaller half $y_1 = x_1 \wedge x_{n+1}, y_2 = x_2 \wedge x_{n+2}, \cdots, y_n = x_n \wedge x_{2n}$ and the larger half $z_1 = x_1 \vee x_{n+1}, \cdots, z_n = x_n \vee x_{2n}$. (One of y's or z's is an all 0 or all 1 sequence, and the other is a bitonic sequence.)

Let N be the number of 1's among x_i's. Now, put $b = y_1 \vee y_2 \vee \cdots \vee y_n$. Then, $b = 1$ if and only if N is equal to or greater than $n + 1$. Put $c = \neg(b)$, and put $w_1 = c \wedge z_1, \cdots, w_n = c \wedge z_n$, and $u_1 = w_1 \vee y_1, \cdots, u_n = w_n \vee y_n$.

If $b = 1$, all z's are 1's and since $c = 0$, all w's are set to 0, and u's $= y$'s. If $b = 0$, w's equal z's, and since all y's are 0's, u's $= z$'s.

Recursively construct a circuit for a bitonic sequence u of length n, repeating this procedure $t = \log\log n$ times. We obtain the most significant t bits b's of the binary representation of $N - 1$ (if $N = 0$, then all b's are 0's), and are left with the bitonic sequence u-last of length $2n/\log n$. We can sort u-last by AKS-sorting network[2] of size $O(n)$ into the sorted list S.

From t bits b's, their negation c's and the list S, a monotone circuit of linear size described below can produce the sorting of the original x's : Let us denote b's by $b_1^1, \ldots, b_t^1$, where b_i^1 is the i-th significant bit of b's and let $b_1^0, \ldots, b_t^0$ be their negations. Note that we have already computed b_i^0's as c's. For $0 \leq i \leq \log n$, put $l_i = \vee_{k=i}^{\log n - 1}(b_1^{k_1} \wedge \cdots \wedge b_t^{k_t})$, where k_j is the j-th significant bit of the binary representation of k. Note that, for $0 < i < \log n$, $l_i = 1$ if and only if N is greater than $i(2n/\log n)$, and that l_0 is always 1 and $l_{\log n}$ is always 0. Finally, for $1 \leq i \leq 2n$, output $l_{\lceil i/(2n/\log n)\rceil} \vee$(the $\{(i-1) \bmod (2n/\log n)+1\}$-st largest bit of S) as the i-th largest bit of the sorted list of x's.

It is easy to check that the size of the entire circuit constructed above is $\Theta(n)$. This completes the proof.

□

5 Concluding Remarks and Open Problems

In the current paper, we established the asymptotic growth rate of the size of a smallest circuit with t negation gates for merging up to a constant factor, for any values of t. Actually, same result holds for bitonic sorting, which subsumes merging. It is an interesting open problem that whether we can extend our technique to obtain the negation-limited circuit complexity for (general) sorting.

The depth of our circuit for merging is $O((t + 1)\log n)$ when the number of available NOT gates is t. In particular, we obtained the linear size circuit with $\log\log n$ NOT gates of depth $O(\log n \log\log n)$. It is obvious that any circuit for merging has depth $\Omega(\log n)$. To see if there exists a linear size circuit of depth $O(\log n)$ with $\log\log n$ NOT gates for merging is an also interesting open problem.

References

1. A.E. Andreev, "On a Method for Obtaining Lower Bounds for the Complexity of Individual Monotone Functions", *Soviet Math. Dokl.*, Vol. 31, No. 3, pp. 530–534, 1985.
2. M. Ajtai, J. Komlós and E. Szmerédi, "An $O(n \log n)$ Sorting Network", *Combinatorica*, Vol. 3, No. 1, pp. 1–19, 1983.
3. N. Alon and R.B. Boppana, "The Monotone Circuit Complexity of Boolean Functions", *Combinatorica*, Vol. 7, No. 1, pp. 1–22, 1987.
4. K. Amano and A. Maruoka, "A Superpolynomial Lower Bound for a Circuit Computing the Clique Function with $(1/6) \log \log n$ Negation Gates", *Proc. 23rd MFCS, LNCS*, No. 1450, pp. 399–408, 1998.
5. S.J. Berkowitz, "On Some Relationships between Monotone and Non-monotone Circuit Complexity", *Tech. Rep. Univ. of Tronto*, 1982.
6. R. Beals, T. Nishino and K. Tanaka, "More on the Complexity of Negation-Limited Circuits", *Proc. 27th STOC*, pp. 585–595, 1995.
7. R.B. Boppana and M. Sipser, "The Complexity of Finite Functions", *Handbook of Theoretical Computer Science*, pp. 757–804, Elsevier Science, 1990.
8. M.J. Fischer, "The Complexity of Negation-Limited Networks–A Brief Survey", *LNCS*, No. 33, pp. 71–82, 1974.
9. D.E. Knuth, *"The Art of Computer Programming Vol. 3 : Sorting and Searching (Second Edition)"*, Addison Wesley, 1998.
10. A.A. Markov, "On the Inversion Complexity of a System of Functions", *J. ACM*, Vol. 5, pp. 331–334, 1958.
11. É. Tardos, "The Gap Between Monotone and Non-Monotone Circuit Complexity is Exponential", *Combinatorica*, Vol. 7, No. 4, pp. 141–142, 1987.
12. K. Tanaka and T. Nishino, "On the Complexity of Negation-Limited Boolean Networks", *Proc. 26th STOC*, pp. 38–47, 1994.
13. A.A. Razborov, "On the Method of Approximations", *Proc. 21st STOC*, pp. 167–176, 1989.