arXiv:1905.10747v2 [cs.CC] 30 Jun 2020

On the monotone complexity of the shift operator

Igor S. Sergeev*

Abstract

We show that the complexity of minimal monotone circuits implementing a monotone version of the permutation operator on n boolean vectors of length q is $\Theta(qn \log n)$. In particular, we obtain an alternative way to prove the known complexity bound $\Theta(n \log n)$ for the monotone shift operator on n boolean inputs.

Introduction. The recent paper [1] shows that a plausible hypothesis from network coding theory implies a lower bound $\Omega(n \log n)$ for the complexity of the n-input boolean shift operator when implemented by circuits over a full basis. As a corollary, the same bound holds for the multiplication of n-bit numbers. (Definitions of boolean circuits and complexity see e.g. in [12].) Curiously, nearly at the same time an upper bound $O(n \log n)$ for multiplication has been proved in [5]. Actually, for the shift operator, the bound $O(n \log n)$ is trivial.

The shift can be implemented by monotone circuits. Lamagna [7, 8] and independently Pippenger and Valiant [11] proved that its complexity is bounded by $\Omega(n \log n)$ with respect to the circuits over the basis $\{\vee, \wedge\}$. Essentially the same bound was established by Chashkin [3] for the close problem of implementation of the real-valued shift operator by circuits over the basis of 2-multiplexors and binary boolean functions. We show that the argument from [3] works for the boolean setting as well thus obtaining yet another proof of the known result. On the other hand, an upper bound $O(n \log n)$ is easy to obtain when a suitable encoding of the shift value is chosen.

A version of the shift operator may be seen as a partially defined order-n boolean convolution operator. It is known that the complexity of the

*e-mail: isserg@gmail.com

convolution is $n^{2-o(1)}$ [4], while the complexity of the corresponding shift operator is $\Theta(n \log n)$.

A more general form of shift is permutation. By analogy, one can introduce a monotone permutation operator. If a special encoding on the set of permutations is chosen, then the permutation operator on n boolean inputs can be implemented with complexity $O(n \log n)$ employing the optimal sorting network from [2]. If size-n boolean vectors are given as inputs, then there exists a version of the permutation operator, which is the restriction of the $n \times n$ boolean matrix multiplication operator. The boolean matrix multiplication complexity is known to be $\Theta(n^3)$ [9, 10]. It can be compared with the complexity $\Theta(n^2 \log n)$ of the corresponding permutation operator. (The lower bound follows from the bound on the complexity of the shift operator.)

Preliminaries. Further, $L(F)$ denotes the complexity of implementing the operator F by circuits over the basis $\{\vee, \wedge\}$.

Let $\mathbb{B} = \{0, 1\}$ and $A = \{\alpha_0, \ldots, \alpha_{n-1}\} \subset \mathbb{B}^m$ be an antichain of cardinality n. By $X = (x_0, \ldots, x_{n-1})$, $x_i = (x_{i,0}, \ldots, x_{i,q-1})^T$, denote the (q, n)-matrix of boolean variables. Let $Y = (y_0, \ldots, y_{m-1})$ denote the vector of boolean variables encoding elements of the antichain A. By $v \ggg k$ we denote the vector obtained from v via a cyclic shift by k positions to the right.

Monotone cyclic shift $(nq+m, nq)$-operator $S_{q,A}(X, Y) = (s_0, \ldots, s_{n-1})$ is a partially defined operator taking values $X \ggg k$ for $Y = \alpha_k$, where $k = 0, \ldots, n-1$.

Consider a few examples of encoding shift values. The vector $(v, \overline{v})$, where $\overline{\cdot}$ is the componentwise negation, we call *doubling* of the vector v. Typically, the shift value k is encoded by its binary representation $[k]_2$. For the monotone version, one can use doubling of $[k]_2$. In this case, $m = 2(\lfloor \log_2 n \rfloor + 1)$. The described encoding corresponds to the antichain $A_0 = \left\{ \left([k]_2, \overline{[k]_2} \right) \middle| \, 0 \le k < n \right\}$.

Another natural choice for A is the set A_1 of all weight-1 vectors in $\mathbb{B}^n$. In this case, $m = n$. Let $q = 1$. Define

$$c_i(X, Y) = \bigvee_{j+k = i \bmod n} x_j y_k.$$

The operator $C(X, Y) = (c_0, \ldots, c_{n-1})$ is called a cyclic *boolean convolution* of the vectors X and Y.

By the definition of the shift operator, $S_{1,A_1}(X, Y)$ coincides with $C(X, Y)$ on inputs from $\mathbb{B}^n \times A_1$. It can be checked that

$$S_{1,A_1}(X, Y) = C(X, Y) \vee x_0 \cdot \ldots \cdot x_{n-1} \cdot g \vee r(X, Y),$$

where g is an undefined boolean vector, and $r(X, Y) = 0$ for $|Y| \le 1$ (here $|v|$ denotes the weight of the vector v). The complexity of convolution is known to be almost quadratic, $L(C) = \Omega(n^2/\log^6 n)$ [4]. Supposedly, a trivial upper bound $L(C) = O(n^2)$ is tight. At the same time, $L(S_{1,A_1}) = O(n \log n)$. We show below that in fact $L(S_{1,A}) = \Omega(n \log n)$ for any A.

Now let $\Pi = \{\pi_0, \ldots, \pi_{n!-1}\} \subset \mathbb{B}^m$ be an antichain of cardinality $n!$. We can assign to its elements different permutations π on the set $\{0, \ldots, n-1\}$. Denote $\pi(X) = \bigl(x_{\pi(0)}, \ldots, x_{\pi(n-1)}\bigr)$. The monotone permutation operator $P_{q,\Pi}(X, Y)$ is defined on inputs $Y \in \Pi$ as $P_{q,\Pi}(X, Y) = \pi(X)$, where the permutation π corresponds to the value of Y. Since a cyclic shift is a special case of permutation, any permutation operator can be viewed as a shift operator defined on a larger domain.

Trivially, any permutation π may be represented by the vector of numbers $([\pi(0)]_2, \ldots, [\pi(n-1)]_2)$. Let Π_0 denote the corresponding coding set (it constitutes an antichain).

Otherwise, permutations may be specified as square boolean matrices with all rows and columns having weight 1. Denote the set of such matrices by $\Pi_1 \subset \mathbb{B}^{n \times n}$. The corresponding permutation operator performs the multiplication of the permutation matrix $Y = \{y_{j,k}\}$ by the matrix of variables X. Define

$$z_{i,k}(X, Y) = \bigvee_{j=0}^{n-1} x_{i,j}\, y_{j,k}.$$

Then $Z(X, Y) = \{z_{i,k}\} : \mathbb{B}^{q \times n} \times \mathbb{B}^{n \times n} \to \mathbb{B}^{q \times n}$ is the operator of boolean product of matrices X and Y. By definition, the operators P_{q,Π_1} and Z take the same values on inputs from $\mathbb{B}^{q \times n} \times \Pi_1$. It is known that $L(Z) = qn(2n-1)$ [10] (see also [12]), which means: the naive method to multiply boolean matrices is optimal. On the other hand, $L(P_{q,\Pi_1}) = O(qn \log n + n^2)$ (see below). Moreover, we manage to show that $L(P_{q,\Pi}) = \Omega(qn \log n)$ for any Π, and this bound is achievable.

Upper complexity bounds. For $v = (v_0, \ldots, v_{m-1}) \in \mathbb{B}^m$ let $Y^v = \bigwedge_{v_i=1} y_i$ denote the monomial of variables y_i corresponding to the vector v. Let $L(A)$ stand for the complexity of computation of the set of monomials $\{Y^\alpha \mid \alpha \in A\}$.

Theorem 1. *$L(S_{q,A}) \le L(A) + O(qn \log n)$.*

Proof. The standard circuit for the shift operator consists of $\log_2 n$ layers of n multiplexors in each. It can be built according to the binary representation of the shift value k. The first layer shifts the input by either 0 or 1 positions, depending on the value of the least significant bit of k. The second layer shifts by 0 or 2 positions, etc.

The monotone circuit employs indicators $Y^{i,\beta} = \bigvee_{\lfloor k/2^i \rfloor = \beta \bmod 2} Y^{\alpha_i}$ of equality of bits of Y to zeros or ones. Instead of multiplexors, there are used similar monotone subcircuits that calculate operators of the form $Y^{i,1}a \vee Y^{i,0}b$.

It remains to note that all boolean sums $Y^{i,\beta}$ can be computed with complexity $O(n)$. □

In particular, since $L(A_0) = O(n)$ and $L(A_1) = 0$, we obtain $L(S_{1,A_0}), L(S_{1,A_1}) \in O(n \log n)$.

To derive the upper bounds on the complexity of the permutation operator, we use a circuit Σ sorting n elements with complexity $O(n \log n)$ provided by [2]. Such a circuit consists of comparator gates that order a pair of inputs.

Theorem 2.
(i) *There exists an antichain* Π *such that* $L(P_{q,\Pi}) = O(qn \log n)$.
(ii) $L(P_{q,\Pi_1}) = O(qn \log n + n^2)$.

Proof. A set Π can be specified following the circuit Σ. Assign to any permutation π a linear order $x_{\pi(0)} > x_{\pi(1)} > \ldots > x_{\pi(n-1)}$ on the set of inputs of Σ (in general, we do not consider these inputs boolean). Let Σ receive inputs ordered in correspondence to a given permutation π. Assign to each comparator e a boolean parameter y_e whose value is determined by the result of the comparison. Let the doubling of the vector of parameters y_e, $e \in \Sigma$, encode a permutation π.

Now, we transform the circuit Σ to a monotone circuit for $P_{q,\Pi}(X, Y)$, replacing any comparator e receiving vector inputs a, b with a subcircuit that evaluates vectors $ay_e \vee b\overline{y_e}$ and $a\overline{y_e} \vee by_e$.

Let us prove (ii). First, recode Y from Π_1 to Π_0. To do this, one simply needs to compute positions $y'_0, \ldots, y'_{n-1}$ of 1s in the columns of the matrix Y. The position of 1 in a weight-1 column may be calculated by a trivial circuit of linear complexity. Therefore, the complexity of the recoding is $O(n^2)$.

Next, arrange the inputs x_i in accordance to the ordering of numbers y'_i with the use of the circuit Σ. At each node of the obtained circuit two y'_i inputs are compared and, depending on the result of the comparison, the order of the vectors x_i accompanied by the numbers y'_i is determined. The complexity of comparison is linear, so the complexity of the subcircuit at each node is $O(q + \log n)$. □

Lower complexity bounds. The proof of the following theorem closely follows the proof of the main result in [3].

Theorem 3. *For any choice of antichain A of cardinality n the following inequality holds:* $L(S_{q,A}) \geq qn \log_2 n - O(qn)$.

Proof. Essentially, it suffices to consider the case $q = 1$. Let S be a monotone circuit of complexity L that computes $S_{1,A}(X, Y)$.

a) First, note that for any assignment $Y = \alpha_k$ for each i, the circuit S contains a path connecting the input x_i and the output $s_{i+k \bmod n}$, and passing exclusively through the gates whose outputs return the function x_i.

Indeed, $s_{i+k \bmod n}(X, \alpha_k) = x_i$ by definition. It remains to check that if $x_i = f \vee g$ or $x_i = fg$, where f and g are monotone functions, then either $f = x_i$ or $g = x_i$. From $x_i = f \vee g$ it follows that $f \leq x_i$ and $g \leq x_i$. Assume that $f \neq x_i$ and $g \neq x_i$. It means that $f = g = 0$ under the assignment $x_i = 1$, $x_j = 0$ for all $j \neq i$. But then $f \vee g = 0 \neq x_i$. A contradiction. The case $x_i = fg$ follows by a dual argument.

So, moving from an output $s_{i+k \bmod n}$ towards the inputs of the circuit, for any gate, we can select an appropriate input computing the function x_i. Finally, we obtain the desired path.

b) Denote the path providing by the above argument by $p_{i,k}$. Let $\chi(e)$ stand for the number of paths $p_{i,k}$, $0 \leq i, k < n$, passing through the gate e in the circuit S. Note that $\chi(e) \leq n$ for all $e \in S$. Indeed, any assignment $Y = \alpha_k$ uniquely defines the function of variables X computed at the output of any gate e. Thus, e does not belong to two different paths $p_{i,k}$ and $p_{j,k}$. Consequently,

$$\sum_{e \in S} \chi(e) \leq Ln. \tag{1}$$

c) Let us estimate the sum $\sum_{e \in S} \chi(e)$ in another way. Denote by $\chi(e, j)$ the number of paths $p_{i,k}$ passing through e to the output s_j. By construction, $\sum_j \chi(e, j) = \chi(e)$.

Consider the subcircuit S_j obtained by combining all n paths $p_{i,k}$ leading to the output s_j, i.e. satisfying the condition $i + k = j \bmod n$. By construction, S_j is a connected binary[1] directed graph with n inputs and one output. We manage to bound $\sum_{e \in S_j} \chi(e, j)$ following a simple argument from [6][2].

Due to the binarity property, the subcircuit S_j has an input at a distance of at least $\log_2 n$ edges from the output. In other words, some path making up S_j contains at least $\log_2 n$ gates. Exclude this path and consider a subcircuit obtained by combining the remaining $n - 1$ paths. Then, it contains a path of length at least $\log_2(n - 1)$. We proceed this way until there is no path remained. The argument leads to the bound

$$\sum_{e \in S_j} \chi(e, j) \geq \log_2 n! = n \log_2 n - O(n), \tag{2}$$

[1] Any vertex receives at most two incoming edges.

[2] In [6], the argument was used to bound the monotone complexity of the boolean sorting operator, see also [12].

following by

$$\sum_{e\in S}\chi(e)=\sum_{j}\sum_{e\in S_j}\chi(e,j)\geq n^2\log_2 n-O(n^2). \tag{3}$$

Putting together (1) and (3), we establish the inequality $L\geq n\log_2 n-O(n)$.

d) For $q>1$, we consider separately the components of the input and output vectors at the same positions. This results in q groups of paths $p_{i,j}$. The inequality (1) remains valid, and the inequality (2) holds for any of qn outputs. Thus, the required bound finally follows. $\square$

Since a permutation operator is a more completely defined shift operator, as a corollary we establish $L(P_{q,\Pi})\geq qn\log_2 n-O(qn)$ for any Π.

The research is supported by RFBR grant, project no. 19-01-00294a.

References

[1] Afshani P., Freksen C., Kamma L., Larsen K. G. Lower bounds for multiplication via network coding. arXiv:1902.10935.

[2] Ajtai M., Komlós J., Szemerédi E. Sorting in $c\log n$ parallel steps. Combinatorica. 1983. **3**(1), 1–19.

[3] Chashkin A. V. On the complexity of a cyclic shift of a set of real numbers. Discrete Analysis and Operations Research. Ser. 1. 2006. **13**(4), 89–92 (in Russian). [Eng. translation in J. Applied and Industrial Math. 2007. **1**(2), 175–177.]

[4] Grinchuk M. I., Sergeev I. S. Thin circulant matrices and lower bounds on the complexity of some Boolean operators. Discrete Analysis and Operations Research. 2011. **18**(5), 38–53 (in Russian). [Eng. translation in arXiv:1701.08557.]

[5] Harvey D., van der Hoeven J. Integer multiplication in time $O(n\log n)$. Tech. report no. 02070778, HAL, 2019.

[6] Lamagna E. A., Savage J. E. Combinational complexity of some monotone functions. Proc. 15th IEEE Symp. on Switching and Automata Theory. New Orleans, 1974, 140–144.

[7] Lamagna E. A. The complexity of monotone functions. Ph.D. thesis. Brown Univ., 1975.

[8] Lamagna E. A. The complexity of monotone networks for certain bilinear forms, routing problems, sorting, and merging. IEEE Trans. on Comp. 1979. **28**, 773–782.

[9] Mehlhorn K. On the complexity of monotone realizations of matrix multiplication. Univ. Saarlandes, Tech. report 74–11, 1974.

[10] Paterson M. S. Complexity of monotone networks for Boolean matrix product. Theor. Comput. Sci. 1975. **1**, 13–20.

[11] Pippenger N., Valiant L. G. Shifting graphs and their applications. J. ACM. 1976. **23**(3), 423–432.

[12] Wegener I. The complexity of Boolean functions. Stuttgart: Wiley–Teubner, 1987.