REPORT R-566 MAY, 1972 UILU-ENG 72-2227

CSL COORDINATED SCIENCE LABORATORY

BOUNDS TO COMPLEXITIES OF NETWORKS FOR SORTING AND FOR SWITCHING

DAVID E. MULLER
FRANCO P. PREPARATA

UNIVERSITY OF ILLINOIS – URBANA, ILLINOIS

"THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC RELEASE AND SALE; ITS DISTRIBUTION IS UNLIMITED."

UILU-ENG 72-2227

BOUNDS TO COMPLEXITIES OF NETWORKS FOR SORTING AND FOR SWITCHING

by

David E. Muller and Franco P. Preparata

This work was supported in part by the Joint Services Electronics Program (U. S. Army, U. S. Navy and U. S. Air Force) under Contract DAAB-07-67-C-0199, and in part by the National Science Foundation under Grant GP-23707.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

This document has been approved for public release and sale; its distribution is unlimited.

BOUNDS TO COMPLEXITIES OF NETWORKS FOR SORTING AND FOR SWITCHING*

David E. Muller and Franco P. Preparata
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Abstract

A network which sorts n numbers, when used to sort numbers of only two sizes, 0 and 1, can be regarded as forming the n frontal (unate) symmetric boolean functions of n arguments. When sorting networks are constructed from comparator modules they appear to require: (1) delay time or number of levels of order $(\log_2 n)^2$, (2) size or number of elements of order $n(\log_2 n)^2$, and (3) formula length or number of literals of order $n^{\log_2 n}$. If one permits the use of negations in constructing the corresponding boolean functions, these three measures of complexity can be reduced to the orders of $\log_2 n$, n, and n^5 respectively. The latter network however is incapable of sorting numbers and may be thought of as merely counting the number of inputs which are 1. One may incorporate this network, however, in a larger network which does sort and in time proportional to only $\log_2 n$.

*This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under contract DAAB-07-67-C-0199, and by NSF grant GP-23707.

BOUNDS TO COMPLEXITIES OF NETWORKS FOR SORTING AND FOR SWITCHING

1. It has been noted that boolean expressions are useful in the analysis of sorting networks [1,3]. Two basic operations often used in sorting networks are the formation of the maximum and the minimum of a pair of numbers. These operations are usually performed at the same time by a two-input, two-output device called a comparator module which may be regarded as being composed of two more basic elements. The first is a comparison element with binary output indicating which of the two inputs is larger and the second is a crossover switch which is set by the output of the first element so as to place the larger number on one line and the smaller on the other.

Using boolean notation, we write $a \vee b$ and ab for the maximum and minimum respectively of the two numbers a and b. A network of comparator modules sorts n numbers if and only if it realizes the n frontal (unate) symmetric boolean functions of n variables [3]. This fact is easily seen, since a sorting network can be used with numbers which are of just two sizes, 0 and 1. Conversely, if all input configurations of 0's and 1's are properly sorted, the output functions are uniquely defined as the frontal symmetric functions. These functions are also the ones which,when applied to arbitrary numbers appearing at the inputs, uniquely describe the properly sorted numbers at the outputs.

In this paper we consider networks constructed exclusively from comparator modules and equivalent networks constructed using other basic elements as well. We shall compare the two classes of networks from the viewpoints of three criteria of complexity. These criteria are: 1) delay, or number

of levels; 2) equipment, or number of elements; and 3) length of formula, or number of literals in the corresponding boolean expressions.

2. To determine the minimum number D(n) of levels of comparator modules required to sort, assuming fan-out is allowed, we need only consider the minimum time required to compute the frontal symmetric boolean function $S(\lceil n/2 \rceil)$ of degree $\lceil n/2 \rceil$, assuming just two-input AND and OR operations are available and that these take equal time. It has been shown [2,3] that

$$\lceil \log_2 n \rceil \leq D(n) \leq \frac{\lceil \log_2 n \rceil (1+\lceil \log_2 n \rceil)}{2} \qquad (1)$$

The value of D(n) is known exactly for small n and has been found to lie closer to the upper bound than the lower. We conjecture that D(n) approaches $\frac{\lceil \log_2 n \rceil (1+\lceil \log_2 n \rceil)}{2}$ asymptotically as n becomes large. This conjecture has been expressed by other workers [4] for the case in which comparator modules are used without fan-out. We prove later in this paper that by using the more basic elements described earlier, a sorting network can be designed which sorts n numbers in time proportional to $\log_2 n$.

All the boolean functions which can be constructed from comparator modules are frontal functions, i.e., they do not require the operation of complementation for their construction. One might think that there would be no advantage to be gained from introducing the operation of complementation if one wishes to construct a frontal function. However, this does not appear to be the case. Let R(n) be the minimum number of levels required to compute the frontal symmetric boolean function $S(\lceil n/2 \rceil)$, assuming not only

two-input AND and OR operations are available but also the NOT operation. Then we shall prove that

$$\lceil \log_2 n \rceil \leq R(n) \leq 6 \lceil \log_2 (n+1) \rceil . \qquad (2)$$

That $\lceil \log_2 n \rceil$ is a lower bound to R(n) may be easily seen from the fact that $S(\lceil n/2 \rceil)$ is a nontrivial function of all n variables. It remains to be shown that $6\lceil \log_2 (n+1) \rceil$ is an upper bound. This is accomplished by design of a network for $S(\lceil n/2 \rceil)$ requiring no more than $6\lceil \log_2 (n+1) \rceil$ levels.

3. Let $x_1,\ldots,x_n$ be a configuration of 0's and 1's. We first design a parallel counter which has as its inputs $x_1,\ldots,x_n$ and as its output the binary representation of the number of 1's in the configuration $x_1,\ldots,x_n$. That such a counter can be designed with a number of levels proportional to $\log_2 n$ is known [5]; to obtain the constant of proportionality 6, we use the following simple inductive argument.

The inputs $x_1,x_2,\ldots,x_n$ are conventionally assumed to be at level 0. When $n \leq 2^m-1$, for some given m, assume inductively that a counter can be designed with outputs $a_{m-1},\ldots,a_0$, where a_0 is the least and a_{m-1} the most significant digit and where each digit a_i is formed at a level no greater than 4m+2i+1. In the trivial cases when n=1 and 2 the result may be easily checked. The inductive step is illustrated in Figure 1. Assume next that n lies in the range $2^m \leq n \leq 2^{m+1}-1$. Let the configurations $x_1,\ldots,x_{2^m-1}$ and $x_{2^m},\ldots,x_{n-1}$ be fed into two such counters giving outputs $a_{m-1},\ldots,a_0$, and $b_{m'-1},\ldots,b_0$ respectively. We take the second input configuration to be empty in case

$2^m=n$. The number m' of digits in the second output configuration is $\lceil \log_2(n+1-2^m) \rceil$. Figure 1 illustrates the case in which m'=m. Now, using two-inputs AND-gates and OR-gates, a full adder stage may be easily designed giving both digit-out d_i and carry-out c_i at level no greater than 4 if it is assumed that digits-in a_i and b_i are at level 0 and carry-in c_{i-1} is at level 2. In fact,

$$d_i = (a_i\bar{b}_i \vee \bar{a}_i b_i)\bar{c}_{i-1} \vee (a_i b_i \vee \bar{a}_i \bar{b}_i)c_{i-1}$$
$$c_i = (a_i \vee b_i)c_{i-1} \vee a_i b_i. \qquad (3)$$

The NOT elements required in these equations are not regarded as adding a level because we may initially invert all the inputs and use a double line system in the remainder of the network, thereby only adding a single level

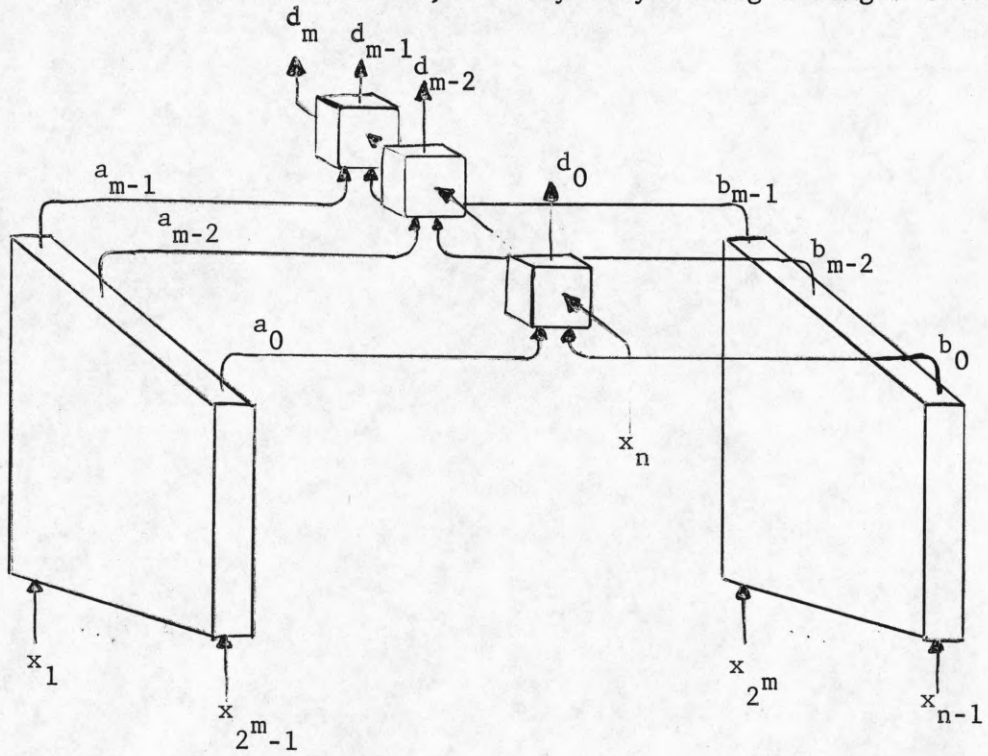

Figure 1. Illustration of the parallel counter.

to the entire counter. We construct m' such adder stages followed by m-m' simplified stages, called half-adders, in which the digit b_i is replaced by 0. The configurations $a_{m-1},\ldots,a_0$ and $b_{m'-1},\ldots,b_0$ are fed into this circuit, while the least significant carry-in c_{-1} is chosen to be x_n. Since a_i and b_i are at level no greater than $4m+2i+1$ and assuming inductively that c_{i-1} is at level no greater than $4m+2i+3$, we obtain d_i and c_i at level no greater than $4m+2(i+1)+3=4(m+1)+2i+1$, for $i=0,\ldots,m-1$. Also, take $d_m=c_{m-1}$, thus extending the result to $i=m$. Since $m+1=\lceil \log_2(n+1) \rceil$, the inductive step is complete.

To construct the symmetric boolean functions $S(1),S(2),\ldots,S(n)$ from $d_m,\ldots,d_0$, let $q_m,\ldots,q_0$ be the binary representation of some integer q in the range $1,\ldots,n$. Letting $S(q_0)=d_0$ if $q_0=1$ and $S(q_0)=1$ if $q_0=0$, we define inductively for $i=1,2,\ldots,m$:

$$S(q_i q_{i-1}\cdots q_0) = \begin{cases} d_i \vee S(q_{i-1}\cdots q_0) & \text{if } q_i=0 \\ d_i \, S(q_{i-1}\cdots q_0) & \text{if } q_i=1 \end{cases} \tag{4}$$

Clearly $S(q_i\ldots q_0)$ can be constructed at level no greater than $4(m+1)+2i+2$. Since $S(q_m\ldots q_0)$ is the symmetric boolean function $S(q)$, each $S(q)$ and, in particular, $S(\lceil n/2 \rceil)$ is obtained at level no greater than $6(m+1)=6\lceil \log_2(n+1) \rceil$.

4. It is interesting to calculate the amount of equipment required by the parallel counter designed here. Since each adder stage has three inputs and two outputs it decreases the number of lines by one, while each half-adder has two inputs and two outputs and hence does not change the number of lines. The total number of input lines to the circuit is n and the total number of output lines is m+1, so the number of adder stages is

$n-(m+1)=n-\lceil \log_2(n+1) \rceil$. Half-adders are inserted at m-m' digit positions in the inductive step described. By induction, we see that the number of half-adders is just equal to the number of 0's in the binary representation of n. At most m half adders are thus required.

As regards adder stages, the above argument is general in the sense that it shows that any circuit for parallel counting constructed from adder and half-adder stages requires the stated number of adder stages. Other circuits, however, may use more half-adders than the one designed here, but they cannot use fewer because of the following argument.

Each adder or half-adder stage in such a circuit is used to add digits of a given weight. The final output digits $d_m,\ldots,d_0$ have weights $2^m,\ldots,2^0$ respectively. The total number of input lines into any given weight position 2^i is just the integer part of $n/2^i$. This number is even or odd depending on whether the i-th digit in the binary representation of n is 0 or 1. At each weight position an adder stage has three inputs and one output so it does not change the parity of the number of lines of that weight. A half-adder, however, has two inputs and one output of the same weight, so it does change the parity of the number of lines having the given weight. There is exactly one output line from the circuit at each weight position and hence the parity of the number of lines at the output is odd, so if the i-th digit in the binary representation of n is 0, it is necessary to have a half-adder at that weight position in order to change the parity from even to odd. This means that at least as many half-adders must be included in the circuit as there are 0's in the binary representation of n. Our circuit is minimal since it achieves this lower bound.

Each adder stage may be constructed to conserve equipment using AND-, OR- and NOT-gates. Thus the entire parallel counter can be realized with a number of gates proportional to n.

To realize the functions $S(1),\ldots,S(n)$ we may use a decoder based on the construction given at the end of section 3. From the inductive definition (4), it is clear that $S(q_i=1,0,\ldots,0)=d_i$, whereas $S(q_i,\ldots,q_0)$ for $q_i\ldots q_0 \neq 2^i$ adds one more gate to the network which realizes $S(q_{i-1},\ldots,q_0)$. Denoting by G_i the number of gates required to generate the set of functions $\{S(q_i,\ldots,q_0)\}$ for all q in the range $1,\ldots,n$, we have the equation $G_i = G_{i-1} + (2^{i+1}-2)$. Thus the number of gates is bounded above by G_m, which is easily shown to be proportional to n.

5. The network just described allows a simple calculation of the length of an expression of the function $S(\lceil n/2 \rceil)$, using the connectives $\vee$ and $\wedge$ and literals in both forms (uncomplemented and complemented). The length of an expression of a function f is defined as the number of literals in the expression, and the minimum length of an expression for f is denoted by L(f). We assume for simplicity that $n=2^{m+1}-1$, i.e., $S(\lceil n/2 \rceil) = d_m$; the extension to the general case is immediate. Note that $L(d_m)$ is the number of inputs to a tree network which realizes d_m, that is, a network whose gates have no fan-out. Thus a trivial upper bound to $L(d_m)$ is $(n+1)^6$, since we have shown that the network realizing d_m has at most $6\lceil \log_2(n+1) \rceil$ levels. In our case, $\lceil \log_2(n+1) \rceil = \log_2(n+1)$, so a binary tree network with $6 \log_2(n+1)$ levels has at most $2^{6\log_2(n+1)} = (n+1)^6$ inputs. A sharper upper-bound, of order $(n+1)^5$ is provided by the following argument, whose explanation is aided by Figure 2.

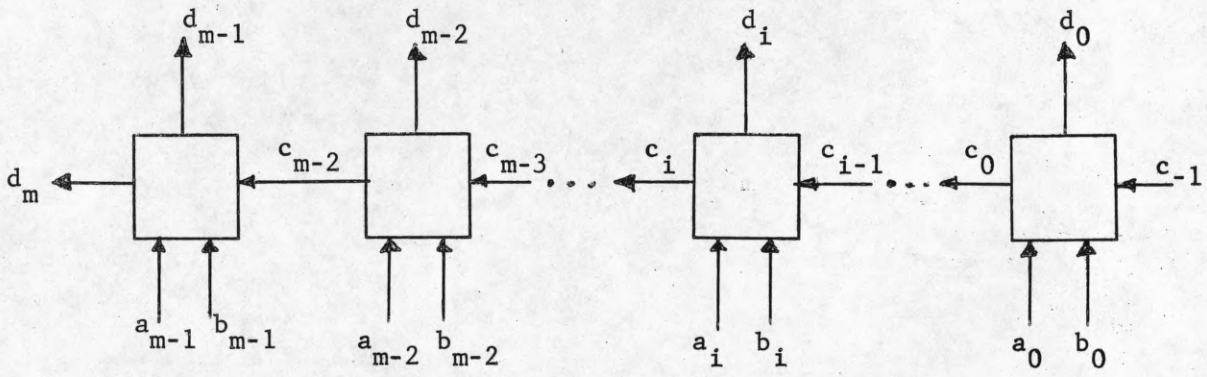

Figure 2. The final string of adder stages in the parallel counter.

By an inductive process we construct a multiple output tree network which realizes the functions $d_m, d_{m-1}, \ldots, d_0$, with several output lines possibly representing any given function. Define $\nu(d_i)$ to be the number of lines representing the function d_i and let $\nu(a_i)$, $\nu(b_i)$, and $\nu(c_i)$ be the multiplicities of the input lines necessary to construct the functions d_i with the assumed multiplicities. From the adder's equations (3) we obtain the inequalities $\nu(a_i) \geq 4\nu(d_i) + 2\nu(c_i)$, $\nu(b_i) \geq 4\nu(d_i) + 2\nu(c_i)$, and $\nu(c_{i-1}) \geq 2\nu(d_i) + \nu(c_i)$. These are inequalities rather than equations, because not all input lines need be used in the actual construction. These inequalities as well as the boundary condition $\nu(c_{m-1}) = \nu(d_m)$ are satisfied by letting $\nu(d_i) = 2^{m-i}$, $\nu(c_i) = 2^{m+1+i} - 3$, and $\nu(a_i) = \nu(b_i) = 2^{m+3-i} = 2^4 \cdot 2^{(m-1)-i}$. This is equivalent to replicating 16 times each of two networks which realize $a_{m-1}, \ldots, a_0$ and $b_{m-1}, \ldots, b_0$, respectively. We recognize that each of these two networks obeys the same rule for the multiplicities of the output lines as the original network. Therefore letting $F_m = \sum_{i=0}^{m} \nu(d_i)L(d_i)$ we have

$$F_m \leq 32F_{m-1} + \nu(c_{-1}) \leq 32F_m + 2^{m-2} - 3$$

It follows that $F_m \leq K32^m - \frac{4}{15}2^m + \frac{3}{31}$, for some constant K. The boundary condition $F_1 = 25$ can be used to determine $K \cong 0.796$. Since $L(d_m) < F_m$ we conclude that $L(d_m) < 0.796 \times 32^m \cong 0.025(n+1)^{\log_2 32} = 0.025(n+1)^5$. A.R. Meyer, M.J. Fischer, and B. Vilfan proved the polynomial growth of $L(S(\lceil n/2 \rceil))$ in n [6] based on a redundant representation of configurations of binary digits interpreted as numbers. We see from the above argument that a polynomial growth can be proved without resorting to such redundant number representation, although it does seem to require the use of literals in complemented as well as uncomplemented form. In fact, we conjecture that there is no fixed power of n which is an upper bound to $L(S(\lceil n/2 \rceil))$ for sufficiently large n, when only uncomplemented literals are used.

6. Using the results obtained in the first five sections for the upper bounds to the various measures of complexity we obtain the following theorems.

Theorem 1: Either there is a frontal function which can be computed in less time if inverters are used than if inverters are not used, or sorting can be accomplished using a network of comparators in time proportional to $\log_2 n$.

Theorem 2: Either there is a frontal function whose network requires less equipment if inverters are used than if inverters are not used, or the median of a set of numbers can be found using a network of comparators whose size is proportional to n.

Theorem 3: Either there is a formula which can be represented without complemented variables but which requires fewer literals if complemented variables are used, or there is a formula without complemented variables for $S(\lceil n/2 \rceil)$ having length bounded by some fixed power of n.

The evidence seems to indicate that the frontal function $S(\lceil n/2 \rceil)$ cannot be computed as rapidly or as economically if inverters are not used because this would imply the existence of a faster and cheaper method of sorting using comparators than is now known.

Each of these theorems poses an open question and the answers to these questions are not entirely independent. For example, if one could show that sorting is possible with comparators in time proportional to $\log_2 n$, then one could conclude that there is a formula without complemented variables for $S(\lceil n/2 \rceil)$ having length bounded by some fixed power of n.

7. It is worth pointing out a basic difference between two types of networks which compute the unate symmetric functions of n boolean variables. The first type constructed exclusively from AND-gates and OR-gates and having uncomplemented literals at its inputs is a sorting network. This property requires that each oriented cutset of this network contain at least n lines, to be traversed by the n numbers being sorted. By contrast, the second type of network, consisting of a parallel counter followed by a decoder, may be constructed so that it has an oriented cutset with no more than $\lceil \log_2 (n+1) \rceil$ lines. This gives intuitive content to the fact that this network, which computes the cardinality of a set, is unable to sort. Obviously, the celebrated zero-one theorem [3] applies only to the first kind of network.

Despite its inability to sort, the parallel counter described earlier may be used in the design of a network which sorts n numbers in time proportional to $\log_2 n$. This network, which we now describe (Figure 3), consists of basic comparison elements with binary output, 2-input AND-gates and OR-gates and single-pole double-throw switches.

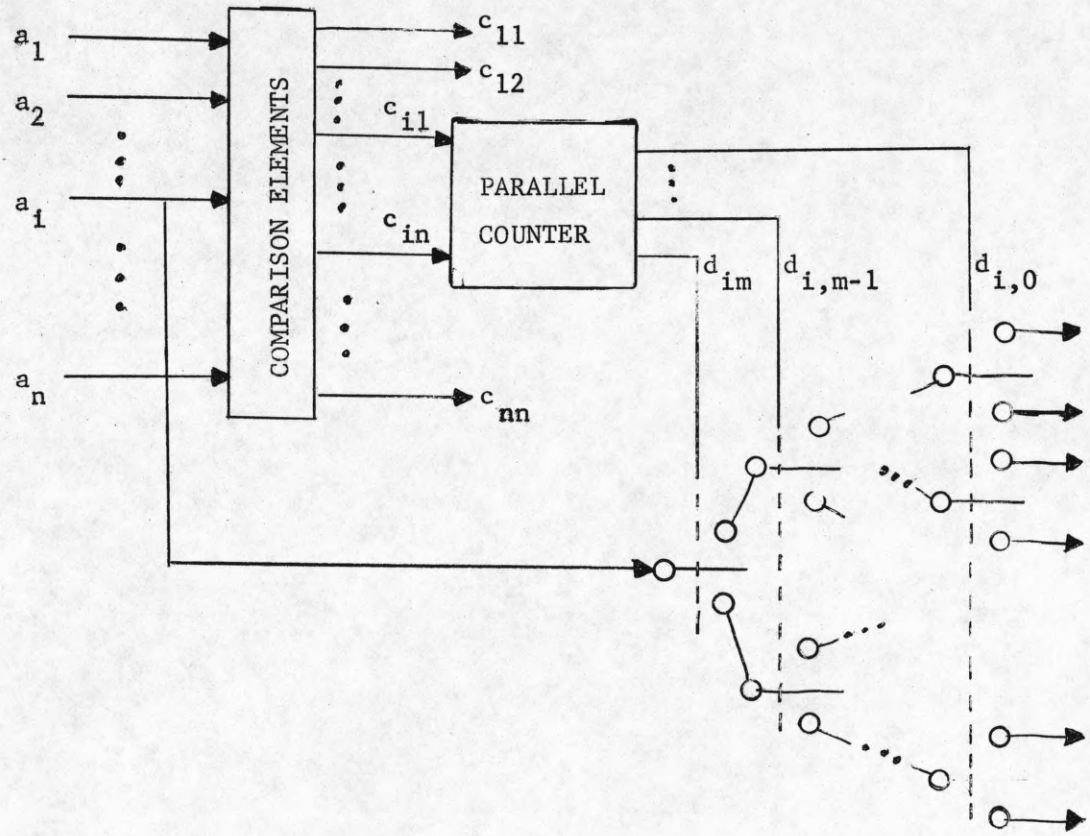

Figure 3. Diagram of a sorting network not constructed from comparator modules.

Let n numbers $a_1, a_2, \ldots, a_n$ be given. At first each number a_i is compared with every other number a_j, thereby obtaining the binary digit c_{ij} as follows:

$$c_{ij} = \begin{cases} 1 & \text{if } a_i \geq a_j, \text{ for } i > j \\ & \text{if } a_i > a_j, \text{ for } i < j \\ 0 & \text{otherwise.} \end{cases}$$

This is done in constant time or, if fan-out is restricted, in time proportional to $\log_2 n$. Then, for each set $\{c_{i1}, c_{i2}, \ldots, c_{in}\}$ of n binary digits, we compute the binary representation $d_{im}, d_{i,m-1}, \ldots, d_{i0}$ of $\sum_j c_{ij}$ by means of the parallel counter described above. This operation also requires a time of order $\log_2 n$. Finally, we use the configuration $d_{im}, d_{i,m-1}, \ldots, d_{i0}$ to drive a binary tree consisting of $(m+1) = \lceil \log_2 (n+1) \rceil$ levels of single-pole double-throw switches. Specifically, the settings of all the switches of the i-th tree at the j-th level from the root are congruent and are controlled by the binary variable $d_{i,m+1-j}$. It is clear that if we feed a_i at the root of its corresponding tree and k of the digits $\{c_{i1}, \ldots, c_{in}\}$ are equal to 1, a_i will emerge at the (k+1)-st terminal of the tree. Since no other number emerges at the (k+1)-st terminal of its corresponding tree, we may simply connect together the homologous terminals of the n trees, and sorting is completed in time proportional to $\log_2 n$.

It is interesting that although the delay of the sorting networks just described has a slower rate of growth than the best known networks consisting of comparator modules, the latter are better from the point of view of equipment complexity. In fact we note the following:

(i) the computation of the digits $\{c_{ij}\}$ requires n(n-1) comparison elements;

(ii) each of the n networks computing $\{d_{i,m}, \ldots, d_{i0}\}$ requires a number of elements proportional to n;

(iii) each of the n switch trees contains (n-1) switches.

We conclude that the network requires a number of elements proportional to n^2.

References

[1] S. Y. Levy and M. C. Paull, "An Algebra with Application to Sorting Algorithms," Proc. 3rd Princeton Conf. Info. Sci. Syst., pp. 286-291, March 1969.

[2] K. E. Batcher, "Sorting Networks and Their Applications," Proc. SJCC, 1968; pp. 307-313.

[3] D. E. Knuth, The Art of Computer Programming, Vol. III, Chapter 5, Addison-Wesley (in press).

[4] M. W. Green, "Some Improvements in Nonadaptive Sorting Algorithms," Proc. 6th Princeton Conf. Info. Sci. Syst., March 1972.

[5] C. C. Foster and F. D. Stockton, "Counting Responders in an Associative Memory," IEEE Trans. on Computers, C-20, No. 12, pp. 1580-1583, 1971.

[6] B. Vilfan, "The Length of Formula Representations of Boolean Functions," unpublished manuscript, Dept. of Elec. Eng., M.I.T., 1971.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois, 61801

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

BOUNDS TO COMPLEXITIES OF NETWORKS FOR SORTING AND FOR SWITCHING

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

David E. Muller and Franco P. Preparata

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May, 1972	14	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAB-07-67-C-0199; NSF Grant No. GP-23707	R-566
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	UILU-ENG 72-2227

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program through U. S. Army Electronics Command, Fort Monmouth, New Jersey, 07703

13. ABSTRACT

A network which sorts n numbers, when used to sort numbers of only two sizes, 0 and 1, can be regarded as forming the n frontal (unate) symmetric boolean functions of n arguments. When sorting networks are constructed from comparator modules they appear to require: (1) delay time or number of levels of order $(\log_2 n)^2$, (2) size or number of elements of order $n(\log_2 n)^2$, and (3) formula length or number of literals of order $n^{\log_2 n}$. If one permits the use of negations in constructing the corresponding boolean functions, these three measures of complexity can be reduced to the orders of $\log_2 n$, n, and n^5 respectively. The latter network however is incapable of sorting numbers and may be thought of as merely counting the number of inputs which are 1. One may incorporate this network, however, in a larger network which does sort and in time proportional to only $\log_2 n$.

DD FORM 1 NOV 65 1473

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computational complexity						
Sorting						
Counting						
Symmetric Boolean functions						
Computation time						
Length of formula						