SIAM J. COMPUT.
Vol. 20, No. 3, pp. 499–505, June 1991

007

A $4n$ LOWER BOUND ON THE COMBINATIONAL COMPLEXITY OF CERTAIN SYMMETRIC BOOLEAN FUNCTIONS OVER THE BASIS OF UNATE DYADIC BOOLEAN FUNCTIONS*

URI ZWICK†

Abstract. A simple, and easy-to-check, property of a symmetric boolean function is shown to imply a $4n - O(1)$ lower bound on the circuit complexity of the function over $U_2 = B_2 - \{\oplus, \equiv\}$, the basis of unate dyadic boolean functions. Among the functions to which this lower bound applies are the modular functions $\text{MOD}_k(n)$ for any fixed $k \geqq 3$ ($\text{MOD}_k(n)$ is the function which returns 1 if and only if $(\sum x_i) \bmod k = 0$). Finally, a $5n$ upper bound is obtained on the circuit complexity over U_2 of the function $\text{MOD}_4(n)$.

Key words. combinational complexity, boolean functions, lower bounds

AMS(MOS) subject classification. 68Q15

1. Introduction. In 1949, Shannon [Sh] showed that the circuit complexity of almost all boolean functions is exponential. However, attempts to obtain concrete lower bounds for functions in NP (see [KM], [HHS], [Sc-1], [Sc-2], [P], [St], [B]) yielded only linear results. The best lower bound of this kind known today over B_2, the full binary basis, is a $3n$ lower bound obtained by Blum [B]. Surveys of these results may be found in [BS], [D], [W].

In this note, we show that a $4n - O(1)$ lower bound may be proved if the linear functions XOR and its complement are removed from the basis. The best previous lower bound over this basis, denoted by U_2, was a $3n$ lower bound on the circuit complexity of the function $\text{MOD}_2(n)$ obtained by Schnorr [Sc-1]. The result presented here is, in a sense, a generalization of Schnorr's result. Another result which is close in spirit to the result presented here is the $2.5n$ lower bound (over B_2) obtained by Stockmeyer [St]. Some of the ideas in this work were inspired by the work of Lai and Muroga [LM].

For additional lower bounds over the bases $\{|\}$, $\{\vee, \neg\}$, $\{\vee, \wedge, \neg\}$, see [So] and [Re] (the symbol | denotes the NAND operation).

In proving the $4n - O(1)$ lower bound we use a simple variation of the elimination method. It is very unlikely that this method will enable us to produce significantly better (for example, nonlinear) lower bounds. In order to achieve such an improvement, a major breakthrough, like the one recently obtained in the theory of monotone circuits (see [Ra], [A], [AB]), is probably needed.

2. Preliminaries. A *circuit* (over U_2) is a directed acyclic graph whose nodes have indegree 0 or 2. Nodes with indegree 0 are called *inputs* and they are labeled by variables or constants. Nodes with indegree 2 are called *gates* and they are labeled by functions from U_2. Note that we may assume, without loss of generality, that all the gates are labeled by the eight nondegenerate U_2-functions $(x^a \wedge y^b)^c$, where $x^a = x \oplus a$.

Each gate in a circuit computes a function by applying the function labeling it to the functions computed by the nodes feeding it. Since we are interested in this note in the computation of scalar functions, we consider only those circuits in which just

* Received by the editors November 16, 1988; accepted for publication (in revised form) April 18, 1989. This paper formed part of a Ph.D. thesis written by the author in Tel-Aviv University under the supervision of Professor Noga Alon.

† Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Present address, The Mathematical Institute, University of Warwick, Coventry CV4 7AL, United Kingdom.

one gate has outdegree 0. This gate is called the *output gate* of the circuit, and the function computed by it is defined to be the function computed by the circuit.

The *size* of a circuit β, denoted by $C(\beta)$, is the number of gates contained in it. The *circuit complexity* of a function f, denoted by $C_{U_2}(f)$ or simply by $C(f)$, is the minimal size of a circuit computing f.

If β is a circuit and A is a gate in β, we denote by $d_\beta(A)$ the outdegree of A, and by $\text{res}_\beta(A)$ the function computed at A (the subscript β is omitted when no confusion arises). If x is a variable, we denote by $d_\beta(x)$ the sum of the outdegrees of the input nodes labeled by x. Actually, we can assume that each variable labels only one input node and then $d_\beta(x)$ is simply the outdegree of the input node labeled by x.

Finally, we denote by $d_1(\beta)$ the number of variables whose outdegree in β is exactly 1. The numbers $d_1(\beta)$ play a central role in the proof of the $4n$ lower bound.

If a gate A in a circuit β is fed by the node B and $\text{res}(B)$ is constant, then we can obtain a smaller circuit which computes f in the following way: If $\text{res}(A)$ is also constant, then we simply remove the incoming edges of A from the circuit and turn A into a constant input node. Otherwise, $\text{res}(A) = \text{res}(C)^a$, where C is the second node in the circuit feeding A and $a \in \{0, 1\}$. In this case we remove the node A and all its incoming and outgoing edges from the circuit. The gates which were fed by A will now be fed directly by C. If $a = 1$, a complement must be incorporated into these gates. (We assume here that A is not the output gate of the circuit.) This process can be carried on until a *simplified circuit* is obtained, i.e., a circuit with no constant input nodes and no gates with constant output.

In the next section, we encounter many situations in which we are given a circuit, some of whose gates are fed by constants. In each such case we explicitly identify a subset of these gates and remove them one by one, as explained in the previous paragraph.

If $g(x, y) \in U_2$, then there exists a constant $c \in \{0, 1\}$ such that $g(c, y)$ is a constant. We say that the constant c *blocks* the function g. Note that this property does not hold for B_2 and this is why proving lower bounds over this base is a harder problem.

3. The lower bound. We begin by defining the set of functions for which our lower bound applies.

DEFINITION 3.1. The sets $S(n)$, $M_k(n)$, $N_l(n)$, $MN_{k,l}(n)$ are defined in the following way:

(1) $f \in S(n)$ if and only if $f(x_1, \cdots, x_n)$ depends only on $\sum_{i=1}^n x_i$. Functions belonging to $S(n)$ are called *symmetric functions*. If $f \in S(n)$ and $f(x_1, \cdots, x_n) = v_k$, where $k = \sum x_i$, we associate with f the binary word $v(f) = v_0 v_1 \cdots v_n$. The word $v(f)$ is called the *value vector* of f.

(2) $f \in M_k(n)$ if and only if $f \in S(n)$ and every restriction of f to a subset of k variables is not constant. It is easy to see that $f \in M_k(n)$ if and only if $v(f)$ does not have a constant subword (i.e., $000\cdots$ or $111\cdots$) of length $k+1$.

(3) $f \in N_l(n)$ if and only if $f \in S(n)$ and every restriction of f to a subset $\{y_1, \cdots, y_l\}$ of f's variables is not linear, i.e., not $y_1 \oplus \cdots \oplus y_l$ or its complement. It is easy to see that $f \in N_l(n)$ if and only if $v(f)$ does not have an alternating subword (i.e., $0101\cdots$ or $1010\cdots$) of length $l+1$.

(4) Finally, we define: $MN_{k,l}(n) = M_k(n) \cap N_l(n)$. In other words, $f \in MN_{k,l}(n)$ if and only if $v(f)$ does not have a constant subword of length $k+1$ or an alternating subword of length $l+1$.

The structure of the sets $M_k(n)$, $N_l(n)$, $MN_{k,l}(n)$ for small k and l is very simple. It is easy to check that $M_1(n)$ contains only the two linear functions $x_1 \oplus \cdots \oplus x_n \oplus c$

that have the value vectors $0101\cdots$ and $1010\cdots$ and that $N_1(n)$ contains only the two constant functions that have the value vectors $000\cdots$ and $111\cdots$. Consequently, the sets $MN_{k,1}(n)$ for $n \geqq k \geqq 1$ and $MN_{1,l}(n)$ for $n \geqq l \geqq 1$ are empty. The first nonempty set of the form $MN_{k,l}(n)$ is $MN_{2,2}(n)$. If $f \in MN_{2,2}$, then $v(f)$ does not contain the subwords 000, 111, 010, 101. The only words with this property are $00110011\cdots$, $0110011\cdots$, $11001100\cdots$, and $1001100\cdots$, and therefore:

$$MN_{2,2} = \left\{ \left\lfloor \frac{(\sum x_i + c) \bmod 4}{2} \right\rfloor : c = 0, 1, 2, 3 \right\}.$$

As a further example, we note that $\mathrm{MOD}_k(n) \in MN_{k-1,3}(n)$ for $k > 2$.

If $f \in MN_{k,l}(n)$ for some $n \geqq k, l$, then $v(f)$ has a subword from the set {001, 110, 100, 011}. In particular, for every two variables $x, y \in \{x_1, \cdots, x_n\}$ the function f has a restriction of the form $(x^a \wedge y^a)^b$. It is also obvious that if $f \in MN_{k,l}(n)$ for some $n > k, l$, then every restriction of f obtained by fixing the value of one variable belongs to $MN_{k,l}(n-1)$.

We can now prove that if $f \in MN_{k,l}(n)$ and $n \geqq k+1, l$, then $C_{U_2}(f) \geqq 4(n-m)-1$, where $m = \max\{k+1, l\}$.

LEMMA 3.2. *Let β be a circuit which computes a function $f \in MN_{k,l}(n)$ for $n > k+1, l$. There exists a circuit δ which computes a function $f' \in MN_{k,l}(n-1)$ and which satisfies* $[C(\delta) - d_1(\delta)] \leqq [C(\beta) - d_1(\beta)] - 4$.

Proof. Let β be a circuit which computes a function $f \in MN_{k,l}(n)$, where $n > k+1, l$. If β is not simplified, then simplify it and denote the simplified circuit obtained by γ. It is easy to check that $[C(\gamma) - d_1(\gamma)] \leqq [C(\beta) - d_1(\beta)]$ (in every elementary simplification step described in the previous section, the size of the circuit decreased by 1 and d_1 could have increased by at most 1). In γ there exists a gate B, which is fed by two input variables. Let x, y be the variables feeding B. The outdegrees of x and y must be at least 2 (since otherwise we can assign a value to one of them and make the output independent of the other).

We say that the circuit γ is *degenerate* if it contains the situation shown in Fig. 3.1 (or the symmetrical situation with the roles of x and y switched). It is easy to transform γ into a nondegenerate circuit γ' which also computes f and which satisfies $C(\gamma') \leqq C(\gamma)$, $d_1(\gamma') = d_1(\gamma)$. This can be accomplished by either deleting A from the circuit, if its output does not depend on both x and y, or by replacing the edge $B \to A$ by an edge $y \to A$, and by adjusting the gate A if necessary, otherwise. Notice that in the latter case $\mathrm{res}(A) = (x^a \wedge y^b)^c$ since in order to compute $(x \oplus y)^d$ at least three U_2-gates are needed. The only variables whose outdegree could have been changed by these actions are x and y. But the outdegrees of x and y were, and must remain, at least 2, and therefore $d_1(\gamma') = d_1(\gamma)$.

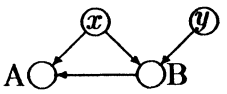

FIG. 3.1

We may therefore assume, without loss of generality, that the circuit γ is nondegenerate. We consider the following three cases.

Case 1. $d_\gamma(x) \geqq 3$ or $d_\gamma(y) \geqq 3$.

Assume without loss of generality that $d_\gamma(x) \geqq 3$. Let A, C be two additional gates fed by x, as shown in Fig. 3.2. Let D be a gate fed by B. Since γ is nondegenerate, $D \neq A, C$. We assign to x the constant c which blocks B and delete the gates A, B, C,

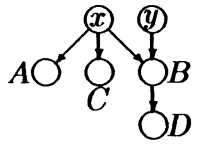

FIG. 3.2

D from β as explained in the previous section. Denote by δ the new circuit obtained. The circuit δ computes the function $f_{x:=c} \in MN_{k,l}(n-1)$. Note that if $d_\gamma(z)=1$, then z does not feed A, C, or D in γ (since otherwise choosing the right values for x and y would make the output of γ independent of z and this is a contradiction since $n>k+1$). Therefore, $d_\delta(z)$ remains 1 and thus we have $C(\delta)=C(\gamma)-4$ and $d_1(\delta)\geqq d_1(\gamma)$ as required.

If Case 1 does not hold, then $d_\gamma(x)=d_\gamma(y)=2$. Denote by A, B the gates fed by x and recall that B is also fed by y.

Case 2. $d_\gamma(B)\geqq 2$.

Denote by C, D two distinct gates fed by B (see Fig. 3.3). Since γ is nondegenerate C, $D\neq A$. We assign to x the constant c which blocks B. As in the previous case, we delete the gates A, B, C, D and we are left with a circuit δ which satisfies $C(\delta)=C(\gamma)-4$ and $d_1(\delta)\geqq d_1(\gamma)$, as required.

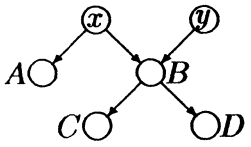

FIG. 3.3

The last case we have to consider is Case 3.

Case 3. $d_\gamma(B)=1$.

We break this case into two subcases.

Case 3.1. There exists an edge $y\to A$ in γ.

If $d_\gamma(A)\geqq 2$, then after switching the roles of x and y we are back in Case 2. We therefore assume that $d_\gamma(A)=d_\gamma(B)=1$. Denote by C the only gate fed by B, and by D the only gate fed by A. We claim that $C\neq D$, for otherwise the output of γ depends on x and y only through the gate $C=D$. If res (C) is constant or of one of the forms x^a, y^b, $(x^a\wedge y^b)^c$, then one of x or y may block the other, and this is clearly a contradiction. The only possibility left is that res $(C)=(x\oplus y)^d$ but this also leads to a contradiction, for in this case f cannot have a restriction to $\{x, y\}$ of the form $(x^a\wedge y^a)^b$.

The situation in this subcase is therefore as shown in Fig. 3.4. We assign to x the constant which blocks B, and delete the gates A, B, C from the circuit. Denote the resulting circuit by δ. Note that $d_\delta(y)=1$ since y feeds in δ only the gate D. Once again, if $d_\gamma(z)=1$, then also $d_\delta(z)=1$, and therefore $C(\delta)\leqq C(\gamma)-3$ and $d_1(\delta)\geqq d_1(\gamma)+1$, as required.

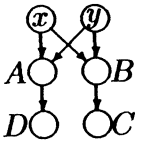

FIG. 3.4

Case 3.2. There is no edge $y \to A$ in γ.

Denote by C the unique gate fed by B. Denote by D the second gate fed by y. By the nondegeneracy of γ we obtain that $C \neq A, D$. We assume in this case that $D \neq A$, thus the situation is as shown in Fig. 3.5. As usual, we assign to x the constant which blocks B, and delete the gates A, B, C from the circuit. We can now repeat the arguments of the previous subcase.

In each one of the above cases we obtained a circuit which satisfies the conditions required and thus the proof is complete. □

THEOREM 3.3. *If* $f \in MN_{k,l}(n)$, *then* $C_{U_2}(f) \geqq 4(n-m)-1$, *where* $m = \max\{k+1, l\}$.

Proof. We prove by induction on n that if β is a circuit which computes $f \in MN_{k,l}(n)$, then $[C(\beta) - d_1(\beta)] \geqq 4(n-m)-1$. The basis of the induction for $n = m$ follows from the fact that $C(\beta) \geqq m-1$, $d_1(\beta) \leqq m$, and therefore $[C(\beta) - d_1(\beta)] \geqq -1$. The induction step follows immediately from Lemma 3.2. □

In fact, this theorem can be slightly improved to $C_{U_2}(f) \geqq 4(n-m)+(m-k)$ using the following lemma.

LEMMA 3.4. *If* $f \in M_k(n)$ *and* β *is a simplified circuit computing* f, *then* $d_1(\beta) < k$.

Proof. Suppose on the contrary that $d_\beta(x_1) = \cdots = d_\beta(x_k) = 1$. Denote by A_i the unique gate fed by x_i for $i \leqq k$. Since β is a simplified circuit, the second input of A_i is not constant. Denote by V_i the set of variables on which this second input depends. Since, by assigning appropriate values to the variables of V_i we can block x_i and thus obtain a constant restriction, we immediately get that $|V_i| \geqq n-k+1$. Thus each V_i contains at least one variable from the set $\{x_1, \cdots, x_k\}$. Denote one such variable by $x_{\pi(i)}$. For each $1 \leqq i \leqq k$ there exists a directed path in β from $x_{\pi(i)}$ to A_i and therefore also from $A_{\pi(i)}$ to A_i. But this is a contradiction since it implies the existence of a directed circuit in β. □

4. An upper bound. In this section, we present U_2-circuits of size $5n-7$ which compute the functions $\mathrm{MOD}_4(n)$ (for $n \geqq 3$). This shows that $4n - O(1) \leqq C_{U_2}(\mathrm{MOD}_4) \leqq 5n - O(1)$.

Using the same methods, it is easy to see that any symmetric boolean function $f(x_1, \cdots, x_n)$ which depends only on $(\sum x_i) \bmod 2^k$ can be computed using $(7-2^{3-k})n + O(2^k/k)$ U_2-gates or $(5-2^{3-k})n + O(2^k/k)$ B_2-gates. In particular, any symmetric boolean function can be computed using $7n + o(n)$ U_2-gates or $5n + o(n)$ B_2-gates.

The basic building block in our circuits is the binary full adder (FA) shown in Fig. 4.1(a). In Figs. 4.1(b) and 4.1(c) it is shown how an FA can be implemented using five B_2-gates or seven U_2-gates. It is easy to check that both implementations are optimal.

We now present the circuits for $\mathrm{MOD}_4(n)$. If $(s_{k-1}, \cdots, s_0)$ is the binary representation of $x_1 + \cdots + x_n$, then $\mathrm{MOD}_4(x_1, \cdots, x_n) = \mathrm{NOR}(s_1, s_0)$. The circuits compute s_1, s_0 using a tree of FAs.

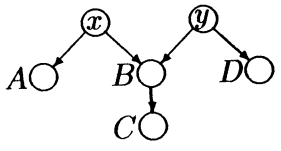

FIG. 3.5

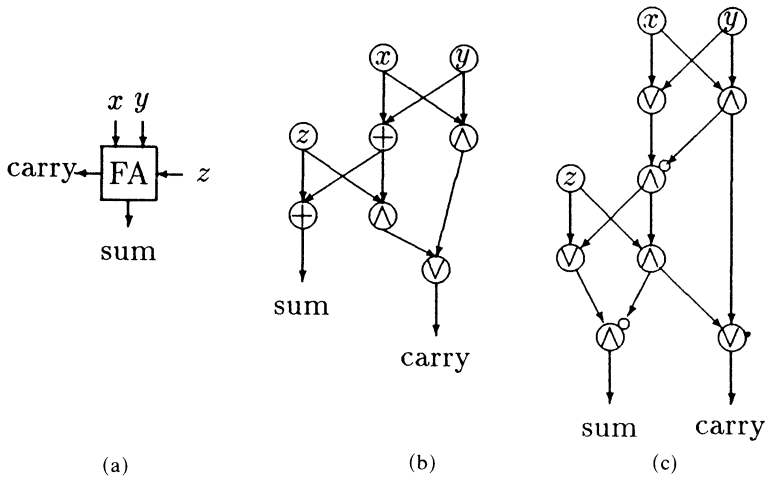

FIG. 4.1. *Implementation of full adders.*

Assume for simplicity that $n \geqq 3$ is odd. Construct a trenary tree of FAs in which every FA is fed by three inputs which are either input variables or sum outputs of previous FAs (this is possible since n is odd). The exact structure of the tree is immaterial. The number of FAs in the tree is $(n-1)/2$. The output of the root FA is s_0. The function s_1 is obtained by computing the XOR of all the carries produced by the $(n-1)/2$ FAs. This XOR can be computed using $3 \cdot ((n-1)/2) - 3$ U_2-gates. The value of the function $\mathrm{MOD}_4(n)$ is now obtained using one additional NOR gate. The total size of the circuit is $7 \cdot ((n-1)/2) + (3 \cdot ((n-1)/2) - 3) + 1 = 5n - 7$.

The same construction yields B_2-circuits of size $3n-3$ for the functions $\mathrm{MOD}_4(n)$. Stockmeyer [St] constructed more efficient circuits for $\mathrm{MOD}_4(n)$ over B_2 and showed that $C_{B_2}(\mathrm{MOD}_4(n)) = 2.5n - O(1)$.

Over U_2 there is still an unresolved gap between the $4n$ lower bound and the $5n$ upper bound presented for $\mathrm{MOD}_4(n)$. We believe that the upper bound is closer to the truth and that the function $\mathrm{MOD}_4(n)$ is the easiest function among the functions to which the lower bound presented in this note applies.

Acknowledgment. The author would like to thank Noga Alon for his help and supervision during the preparation of this work.

REFERENCES

[A] A. E. ANDREEV, *On a method for obtaining lower bounds for the complexity of individual monotone functions*, Dokl. Akad. Nauk. SSSR, 282 (1985), pp. 1033–1037. (In Russian.) English translation in Sov. Math. Dokl., 31 (1985), pp. 530–534.

[AB] N. ALON AND R. B. BOPPANA, *The monotone circuit complexity of boolean functions*, Combinatorica, 7 (1987) pp. 1–22.

[B] N. BLUM, *A Boolean function requiring 3n network size*, Theoret. Comput. Sci., 28 (1984), pp. 337–345.

[BS] R. B. BOPPANA AND M. SIPSER, *The complexity of finite functions*, in Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, J. van Leeuwen, ed., Elsevier, Amsterdam, the Netherlands, pp. 757–800.

[D] P. E. DUNNE, *The complexity of Boolean Networks*, Academic Press, London, 1988.

[HHS] L. H. HARPER, W. N. HSIEH, AND J. E. SAVAGE, *A class of boolean functions with linear combinational complexity*, Theoret. Comput. Sci., (1975), pp. 161–183.

[KM] B. M. KLOSS AND V. A. MALYSHEV, *Bounds on complexity of some classes of functions*, Vestnik Moskov. Univ. Ser. Mat. Mekh., 4 (1965), pp. 44–51. (In Russian.)

[LM] H. C. LAI AND S. MUROGA, *Logic networks with a minimum number of* NOR (NAND) *gates for parity functions of n variables*, IEEE Trans. Comput., 36 (1987), pp. 157–166.

[P] W. J. PAUL, *A 2.5n-lower bound on the combinatorial complexity of boolean functions*, SIAM J. Comput., 6 (1977), pp. 427–443.

[Ra] A. A. RAZBOROV, *Lower bounds for the monotone complexity of some boolean functions*, Dokl. Akad. Nauk. SSSR, 281 (1985), pp. 791–801. (In Russian.) English translation in Sov. Math. Dokl., 31 (1985), pp. 354–357.

[Re] N. P. RED'KIN, *Proof of minimality of circuits consisting of functional elements*, Problemy Kibernet., 23 (1970), pp. 83–101. (In Russian.)

[Sc-1] C. P. SCHNORR, *Zwei Lineare untere Schranken für die Komplexität Boolescher Funcktionen*, Computing, 13 (1974), pp. 155–171.

[Sc-2] ———, *The combinational complexity of equivalence*, Theoret. Comput. Sci., 1 (1976), pp. 289–295.

[Sh] C. E. SHANNON, *The synthesis of two-terminal switching circuits*, Bell Systems Tech. J., 28 (1949), pp. 59–98.

[So] E. P. SOPRUNENKO, *Minimal realization of functions by circuits using functional elements*, Problemy Kibernet., 15 (1965), pp. 117–134.

[St] L. STOCKMEYER, *On the combinational complexity of certain symmetric boolean functions*, Math. Systems Theory, 10 (1977), pp. 323–336.

[W] I. WEGENER, *The Complexity of Boolean Functions*, Wiley-Teubner Ser. Comput. Sci., 1987.