SOME RESULTS ON CIRCUIT DEPTH

William Finlay McColl

A dissertation submitted to the University of Warwick
for the degree of Doctor of Philosophy.

Computer Science Department,
Warwick University,
Coventry, England.

October 1976

CONTENTS

ACKNOWLEDGEMENTS

This dissertation contains the results of research carried out at the Universities of Warwick and Edinburgh. I take this opportunity to thank my supervisor M. S. Paterson for the constant guidance and encouragement which I received throughout this research. Section 3.2 and the appendix contain results which were derived jointly with him.

Thanks are also due to the S.R.C. for financial support and to Mrs. A. Houston for typing this manuscript.

SUMMARY

An important problem in theoretical computer science is to develop methods for estimating the complexity of finite functions. For many familiar functions there remain important gaps between the best known lower and upper bounds. We investigate the inherent complexity of Boolean functions taking circuits as our model of computation and depth (or delay) to be the measure of complexity. The relevance of circuits as a model of computation for Boolean functions stems from the fact that Turing machine computations may be efficiently simulated by circuits.

Important relations among various measures of circuit complexity are obtained as well as bounds on the maximum depth of any function and of any monotone function. We then give a detailed account of the complexity of NAND circuits for several important functions and pursue an analysis of the important set of symmetric functions. A number of gap theorems for symmetric functions are exhibited and these are contrasted with uniform hierarchies for several large sets of functions.

Finally, we describe several short formulae for threshold functions.

1. INTRODUCTION

Let $X_n = \langle x_o, x_1, \ldots, x_{n-1} \rangle$ be an n-tuple of formal arguments. A partial function $f : D^n \rightarrow D$ with finite domain D^n is called a finite function of n arguments and may be written as $f\ (x_o, x_1, \ldots, x_{n-1})$. If $D = \{0,1\}$, then the functions $f : D^n \rightarrow D$ are known as Boolean functions. There are many different computation procedures for any given finite function and each of these uses a certain amount of resources e.g. time, space. The time complexity of some finite function f can thus be defined as the minimal amount of time required by any computation of f. In a similar way, complexity can be defined with respect to other measures.

We shall consider computations of Boolean functions by acyclic circuits of binary gates where each gate corresponds to some binary Boolean function. Two fundamental complexity measures for a Boolean circuit are size and depth. Another measure closely related to circuit depth is formula size. Our primary concern here will be with circuit depth. Some of the results to be presented have appeared in a preliminary report on the depth of Boolean functions, see McColl (1976).

1.1 SOME MOTIVATIONAL REMARKS.

The study of circuit complexity is important for both practical and theoretical reasons. The practical motivation is that many of the tasks for which digital hardware must be designed can be represented as the computation of Boolean functions. The two fundamental measures of circuit complexity are closely related to the cost and delay associated with such hardware. This practical significance provided the original stimulus for research in this area. However, until quite recently few mathematicians outside the Soviet Union recognized circuit complexity as a legitimate branch of mathematics. Birkhoff (1971) remarks that pure theorists working on Boolean algebra have tended to overlook the natural but extremely difficult problem of estimating the complexity of Boolean functions.

Recently there has been considerable interest in the computational complexity of algebraic and combinatorial problems. It is now recognized that the development of methods for estimating the complexity of finite functions is of vital importance if we are to reach a complete understanding of many familiar problems. This theoretical motivation has provided additional stimulus and the study of Boolean function complexity is now one of the most active areas in theoretical computer science. Despite considerable research effort, only modest progress has been made in this area and for many familiar functions the best known lower bounds appear to be very weak. Much of the theoretical interest in circuits as a model of computation for Boolean functions stems from the fact that Turing machine computations may be efficiently simulated by circuits.

In complexity studies, Turing machines are the classical model of computation and it is known that Turing machine complexity closely reflects the difficulty which is experienced in computing finite functions. Therefore, results on circuit complexity are of relevance to practical computations.

At the present time the cost of digital hardware is diminishing rapidly. Therefore from the point of view of hardware design it seems more important to minimise the depth of a circuit than to minimise circuit size. Another motivation for studying circuit depth stems from the capability of parallel processing on modern computers. This raises the problem of designing efficient algorithms which minimise delay. In practice we might only be interested in those algorithms which require only a fixed number of processors. However, some of the techniques developed in designing circuits with small depth may be of use in designing such algorithms, even although the circuits (which use unbounded parallelism) are not of practical value.

We have given some practical reasons for studying circuit depth. However, our main aim is to reach a deeper understanding of the inherent difficulty involved in computing Boolean functions and of the reasons for this difficulty.

1.2 DEFINITIONS. Let $B_n = \{ f \mid f: \{0,1\}^n \to \{0,1\} \}$.

We note that $|B_n| = 2^{2^n}$ and thus $|B_2| = 16$.

To introduce our notations for these 16 basic functions we list them in the following table with definitions in terms of GF(2), the two-element field.

Symbol for f	Name for f	$f(x_0,x_1)$
0	constant	0
1	"	1
π_0	projection	x_0
π_1	"	x_1
$\overline{\pi}_0$	"	$1 + x_0$
$\overline{\pi}_1$	"	$1 + x_1$
$\wedge$	conjunction	$x_0 \cdot x_1$
NAND	nand	$1 + x_0 \cdot x_1$
$\vee$	disjunction	$x_0 + x_1 + x_0 \cdot x_1$
NOR	nor	$(1+x_0).(1+x_1)$
$\rightarrow$	implication	$1 + x_0 + x_0 \cdot x_1$
$\leftarrow$	"	$1 + x_1 + x_0 \cdot x_1$
$\overline{\rightarrow}$	"	$x_0 \cdot (1+x_1)$
$\overline{\leftarrow}$	"	$x_1 .(1+x_0)$
$\oplus$	sum (modulo 2)	$x_0 + x_1$
$\equiv$	equivalence	$1 + x_0 + x_1$

The 16 functions of B_2 with GF(2) equivalents

Table 1

Functions in B_n are to be computed by circuits over some basis Ω , where $\Omega \subseteq B_2$. A circuit is a connected acyclic directed graph in which nodes have either in-degree 2 (gates) in which case the pair of incoming arcs are ordered, or else in-degree 0 (input nodes) in which case an input from some set is associated with the node. A formula is a circuit in which all gates have out-degree at most one. Each gate is labelled with a binary Boolean function from the basis Ω .

Let $I_n = \langle x_0, x_1, \ldots, x_{n-1}, \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_{n-1}, 0, 1 \rangle$ be the set of possible inputs in formulae and circuits, where $\bar{x}_i$ denotes the complement of Boolean variable x_i.

Let $X_n = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ be the set of formal arguments. A Boolean function $f \in B_n$ will be written as $f(x_0, x_1, \ldots, x_{n-1})$ or as $f(X_n)$.

In a circuit β , an input node associated with $\tilde{x}_i$, where $\tilde{x}_i \in I_n$, is said to compute the function $\tilde{U}_i\ (X_n) = \tilde{x}_i$. Proceeding inductively, a gate ν labelled with a binary Boolean function h is said to compute the function

$$f_\nu(X_n) = h(\ f_{\nu_1}(X_n),\ f_{\nu_2}(X_n)\ )$$

where ν_1 , ν_2 and f_{ν_1}, f_{ν_2} are the nodes on the first and second arcs entering ν and the functions they compute. A circuit β computes f if there is a node in β which computes f.

The size $C(\beta)$ of a circuit β is the total number of gates. The <u>depth</u> $D(\beta)$ of a circuit β is the maximum number of gates in any path. The <u>size</u> $F(\beta)$ of a formula β is the total number of input nodes and this is one more than the number of gates. Each of these circuit parameters induces a corresponding complexity measure over B_n in a natural way.

For any f in B_n,

$$C_\Omega(f) = \min\left\{ C(\beta) \;\middle|\; \beta \text{ is a circuit over } \Omega \text{ which computes } f \right\}$$

$$D_\Omega(f) = \min\left\{ D(\beta) \;\middle|\; \beta \text{ is a circuit over } \Omega \text{ which computes } f \right\}$$

$$F_\Omega(f) = \min\left\{ F(\beta) \;\middle|\; \beta \text{ is a formula over } \Omega \text{ which computes } f \right\}$$

A basis Ω <u>covers</u> $f \in B_n$ iff f can be computed by a circuit over Ω with inputs from the set X_n. If f is not covered by Ω, then $C_\Omega(f)$, $D_\Omega(f)$ and $F_\Omega(f)$ are defined to be $+\infty$. If each f in B_2 is covered by Ω, then Ω is said to be <u>complete</u>. For example, B_2, $\{\wedge, \vee, \bar{\Pi}_0\}$ are complete bases.

Let $M_n = \left\{ f \in B_n \;\middle|\; \underline{X} \leqslant \underline{Y} \rightarrow f(\underline{X}) \leqslant f(\underline{Y}) \right\}$

where $\underline{X},\underline{Y}$ are n-tuples of Boolean variables x_i, y_i, $0 \leqslant i < n$. We write $\underline{X} \leqslant \underline{Y}$ if for all i, $x_i \leqslant y_i$, where $0 \leqslant 0$, $0 \leqslant 1$, $1 \leqslant 1$. M_n is the set of <u>monotone increasing</u> Boolean functions of n arguments. It is well-known that

M_n is precisely the set of n argument Boolean functions which are covered by the incomplete basis

$$M_2 = \{ \wedge, \vee, \Pi_0, \Pi_1, 0, 1 \} .$$

In all subsequent considerations of circuits over the basis M_2 we shall let X_n (and not I_n) be the set of possible inputs..

1.3 MACHINES $\Rightarrow$ CIRCUITS. When considering the complexity of Boolean functions, two commonly used models of computation are Turing machines and circuits. Recently there has been considerable interest in the relations among complexity measures for these two models. Several results have been derived which show that Turing machine computations may be efficiently simulated by circuits.

Let M be a Turing machine accepting or rejecting an input string $W \in \{0,1\}^n$ within time bound $T(n)$. A result of N. Pippenger and M. J. Fischer shows that the computation of M may be simulated by a Boolean circuit over the basis B_2 which computes some $f \in B_n$ and which has $O(T(n)\log T(n))$* gates. Therefore lower bounds on the circuit size of such a function yield corresponding lower bounds on the running time of the Turing machine. Pratt and Stockmeyer (1976),

* All logarithms are taken to base 2 unless otherwise stated.

$$O(f(n)) = \left\{ g(n) \;\middle|\; \begin{array}{l} \text{There are positive constants } C, n_0, \\ |g(n)| \leqslant C.\ f(n) \quad \text{for all } n \geqslant n_0 \end{array} \right\}$$

Borodin (1975) show that a nondeterministic L(n) tape bounded Turing machine can be simulated on n bits of input by a Boolean circuit of depth $O(L(n)^2)$. We now consider some consequences of these relations.

Machine-based complexity theory is concerned with relations among complexity measures in different models of computation. As Turing machine complexity is closely related to circuit complexity we can pose many of the open problems concerning machines in terms of the size and depth of circuits. For example, a conjecture of Cook (1974) concerning the relative power of time and space could be proved by demonstrating a function $f \in B_n$, where

$$C_{B_2}(f) = O(n^k) \text{ for some fixed } k$$

<u>and</u>

$$D_{B_2}(f) > (\log n)^k \text{ for any fixed } k.$$

Likewise, the $P = NP?$* question could be resolved by establishing a nonpolynomial lower bound on $C_{B_2}(f)$ for some $f \in B_n$ whose corresponding language recognition problem is in NP. For example, the function which is true iff there is a clique of size $\lceil n/2 \rceil$** in a graph with n nodes.

* P(NP) is the class of languages recognizable by deterministic (nondeterministic) Turing machines within time polynomial in the length of the input.

** $\lceil i \rceil$ denotes the least integer greater than or equal to i ; $\lfloor i \rfloor$ will denote the greatest integer less than or equal to i .

The relationship between Turing machine space and circuit depth is based on a simulation of space bounded machines. This simulation relies heavily on the transitive closure problem for binary relations on finite sets. An upper bound of $O((\log n)^k)$ on the circuit depth of transitive closure would immediately yield an upper bound of $O(L(n)^k)$ on the circuit depth required to simulate a nondeterministic $L(n)$ tape bounded Turing machine. This raises the following

<u>Open problem</u>

$$D_{B_2}(TC(n)) = o((\log n)^2)^{*} \quad ?$$

where $TC(n)$ is the transitive closure problem for sets of size n. Any nontrivial lower bound on the depth of transitive closure over the monotone basis M_2 would also be of interest.

---

* $f(n) = o(g(n))$ denotes the fact that $f(n)$ <u>grows more slowly</u> than $g(n)$; i.e. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

2. RELATIONS AMONG MEASURES

Fortunately the various measures of circuit complexity are not entirely independent. In this chapter we investigate the relationships among these measures. We also note the effect of different binary bases on the complexity of Boolean functions.

2.1 GLOBAL RELATIONS FOR B_n.

In this section we consider circuits over complete bases and note a number of relationships which hold for all Boolean functions. Two of these are immediate.

<u>Lemma 2.1</u>

For all f in B_n and all complete binary bases Ω,

$$C_\Omega(f) < F_\Omega(f) \leqslant 2^{D_\Omega(f)}$$

<u>Proof</u>

The first inequality follows from the fact that a formula is a restricted form of circuit. The second follows from the observation that for any circuit an equivalent formula <u>with the same depth</u> can be constructed by replicating nodes of the circuit until the unit fan-out restriction is satisfied. Furthermore any binary tree with depth d has at most 2^d external nodes

□

These inequalities are the best possible of their type. This can be seen by considering an appropriate function for the basis in question. For example, if Ω is the full binary basis B_2 then we need only consider the function f in B_n which takes the value 1 iff all its arguments are 1, i.e.

$$f(X_n) = \bigwedge_{i=0}^{n-1} x_i$$

It is evident that for $n=2^p$,

$$C_{B_2}(f) + 1 = F_{B_2}(f) = 2^{D_{B_2}(f)} = n$$

Therefore the above inequalities cannot be improved for the basis B_2. By choosing appropriate functions we can show this to be true for all complete bases. As a consequence there is no nontrivial lower bound on depth in terms of formula size or circuit size which holds for all Boolean functions.

For inequalities in the reverse directions we have no such complete results.

<u>Notation</u>

Where no ambiguity can arise we shall henceforth refer to the basis $\{NAND\}$ simply as NAND.

<u>Theorem 2.2</u>

For all f in B_n,

$D_{NAND}(f) \leqslant k.\log F_{B_2}(f) + O(1)$

where $k = 2 \log_{\phi} 2 \simeq 2.88$

and ϕ is the (unique) real positive root of

$z^2 = z + 1$. ϕ is known as the <u>golden</u> <u>ratio</u>.

Let $|F|$ denote the size of formula F. According to a well known lemma by Brent et al. (1973, Lemma 2) for any number $1 < m \leqslant |F|$ a subformula $L \theta R$ of F can be found, for $\theta \in B_2$, such that $|L \theta R| \geqslant m$, $|R| \leqslant |L| < m$. This affords a partition of F into three subformulae L,R and A, where A is formula F with $L \theta R$ replaced by a new indeterminate a.

Let L,R and A compute $L(X_n)$, $R(X_n)$ and $A(X_n,a)$ respectively. Then the formula

$(L(X_n)\ \theta\ R(X_n))\text{NAND}\ A(X_n,1).\text{NAND}.\ (L(X_n)\ \bar{\theta}\ R(X_n))\text{NAND}\ A(X_n,0)$

computes the same function as F. Every binary Boolean function has an associated NAND circuit of depth not more than 2 when variables are available as inputs in both complemented and uncomplemented form. Therefore an arbitrary Boolean formula can be expressed in the alternative form of Fig.1 where each gate computes the NAND function and each Z_i, $1 \leqslant i \leqslant 8$, computes either $L(X_n)$, $R(X_n)$ or one of their complements.

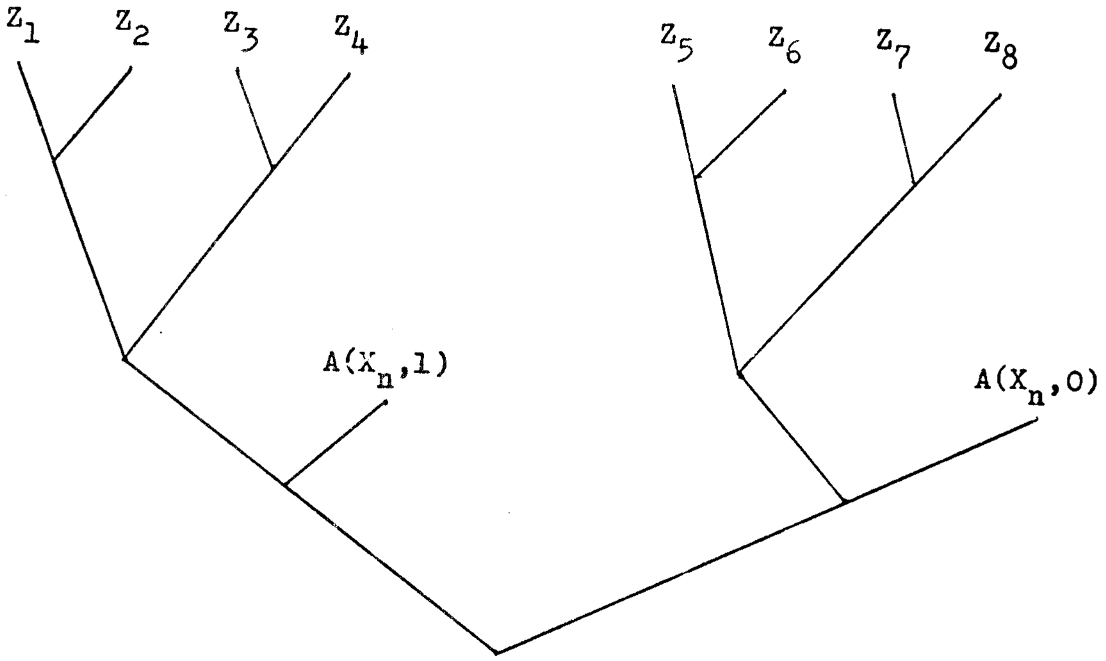


Fig.1 An alternative NAND formula

Notice that the formula size of both $A(X_n,1)$, $A(X_n,0)$ over the basis B_2 is not more than $|A|$. Also note that the formula size of $L(X_n)$ over this basis is equal to the formula size of its complement and that this is not more than $|L|$. Similarly for $R(X_n)$.

Let $d(\alpha) = \max\left\{D_{NAND}(f) \,\middle|\, F_{B_2}(f) \leqslant \alpha\right\}$

Lemma 2.3

Let $r_0 = r_1 = 1$, $r_2 = r_3 = 2$, and

$$r_{k+4} = r_{k+2} + r_k \tag{2.1}$$

for $k \geqslant 0$. Then $d(r_k) \leqslant k$.

Proof

As an inductive hypothesis suppose that

$d(r_0) \leqslant 0$, $d(r_1) \leqslant 1$, ..., $d(r_{k+3}) \leqslant k+3$

(By inspection, this is true for $k = 0$). We shall show that

$d(r_{k+4}) \leqslant k+4$.

Let F be any formula over the basis B_2 where $|F| = r_{k+4}$. Using the lemma of Brent et al. find L and R (joined by θ) with $m = r_k$. Restructure the formula according to the expansion described above. Since $m = r_k$ we have

$|R| \leqslant |L| < r_k$.

By the inductive hypothesis, the functions $L(X_n)$, $R(X_n)$ and their complements can be computed by NAND circuits within depth k. Also, $|L \theta R| \geqslant r_k$ and so $|A| \leqslant r_{k+4} - r_k = r_{k+2}$. Again, by the inductive hypothesis $A(X_n,1)$, $A(X_n,0)$ have depth at most k+2 over the basis NAND and the result follows by induction on k

$\square$

We are now ready to give

<u>Proof of Theorem 2.2</u>

From the linear recurrence relation (2.1) we have for all $n \geqslant 0$,

$$r_{2n} = r_{2n+1} = a_{n+1}$$

where a_n is the n^{th} number in the Fibonacci sequence

1,2,3,5,8,13,21,34,55,.....

given by the boundary conditions $a_1 = 1$, $a_2 = 2$ and the recurrence relation $a_n = a_{n-1} + a_{n-2}$ $(n \geqslant 3)$.

An explicit formula for a_n as a function of n is now given without proof.

$$a_n = \frac{\phi^{n+1} - \beta^{n+1}}{\phi - \beta} \quad \text{for } n = 1,2,3,4,\ldots.$$

where $\phi = \frac{1}{2}(1+\sqrt{5})$, $\beta = \frac{1}{2}(1-\sqrt{5})$ are roots of the equation

$$Z^2 = Z + 1.$$

Let $k \geqslant 0$ be such that $r_{2k} < n \leqslant r_{2k+2}$.

Then $n \geqslant r_{2k} + 1 > a_{k+1}$ and using the explicit formula given above we obtain

$$\phi^k \leqslant C.n \quad \text{for some constant } C$$

giving

$$k \leqslant \log_\phi (C.n).$$

From Lemma 2.3,

$$d(n) \leqslant d(r_{2k+2}) \leqslant 2k+2$$

so

$$d(n) \leqslant 2 \log_\phi n + O(1)$$

and the result follows

$\square$

Remarks

When the construction described above is used to restructure formulae over the restricted basis NAND it appears to be inefficient in many ways. However, despite considerable study we have not yet obtained an improved strategy for this special case. This raises the following

Open problem

Improve the coefficient $k \simeq 2.88$ in the global relation for all f in B_n,

$$D_{NAND}(f) \leqslant k . \log F_{NAND}(f) + O(1)$$

It seems likely that this coefficient can be improved although Theorem 4.4 demonstrates functions f in B_n for which $F_{NAND}(f) = O(n)$ and $D_{NAND}(f) = 2 \lceil \log n \rceil$. In view of this result the coefficient cannot be reduced to less than 2.

An essentially similar technique to that of Theorem 2.2 can be used to show that for all f in B_n,

$$D_{B_2}(f) \leqslant k . \log F_{B_2}(f) + O(1)$$

where $k \simeq 2.465$.

Definition

$U_2 = B_2 - \{ \oplus , \equiv \}$ is the set of mixed-monotone or unate binary Boolean functions. This unate basis is of considerable interest since it is known that for all f in B_n,

$$F_{U_2}(f) = \Pi(f)$$

where $\Pi(f)$ is the minimum number of contacts in any Π - circuit (series-parallel contact circuit) which realises f.

Preparata and Muller (1976) prove that for all Boolean functions $f \in B_n$,

$$D_{U_2}(f) \leqslant 1.81 \log F_{U_2}(f) + O(1)$$

These results raise the problem of proving lower bounds on the best possible coefficient. We have already noted that there is a lower bound of 2 for the problem of restructuring arbitrary NAND formulae so as to minimize depth. For the bases B_2, U_2 no such result is known and we have only the trivial lower bound of 1. We have already noted that there is no nontrivial lower bound on depth in terms of formula size which holds for all Boolean functions. Therefore in order to get nontrivial lower bounds on coefficient size we should tackle the following question about specific functions:

<u>Open problem</u>

Establish a lower bound on depth over B_2, U_2 which is not derivable from a corresponding bound on formula size. $\square$

We thus have satisfactory, although not complete, answers to questions about the relations between circuit depth and formula size over various bases. Much less is known about the relationship between circuit size and depth. Indeed, only recently has it been established that for all Boolean functions, circuit size is nonlinear in depth. Paterson and Valiant (1976) prove that for all f in B_n,

$$D_{B_2}(f) = O(C_{B_2}(f) / \log C_{B_2}(f))$$

Noting that for all complete binary bases Ω,

$D_{\Omega}(f) \geqslant \log F_{\Omega}(f)$, we can obtain from the above result a relation between the circuit size and formula size of all Boolean functions.

Finally, we consider the expressive power of different bases with respect to some complexity measure. Two results are presented which indicate the maximum disparity between NAND, B_2 w.r.t. depth and U_2, B_2 w.r.t. formula size. In subsequent sections of this chapter we pursue further some problems concerned with the relative power of bases.

<u>Lemma 2.4</u>

For all f in B_n,

$D_{NAND}(f) \leqslant 2.D_{B_2}(f)$

when $I_n = \langle x_0, x_1, \ldots, x_{n-1}, \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_{n-1}, 0, 1\rangle$ is the set of possible inputs.

<u>Proof</u>

Consider any circuit of depth D which computes f and which has gates drawn from the basis B_2.

By applying the identities:

$(x_0 \wedge x_1) \equiv (x_0 \text{NAND } x_1 . \text{ NAND}. \ x_0 \text{ NAND } x_1)$

$(x_0 \vee x_1) \equiv (x_0 \text{NAND } x_0 . \text{ NAND}. \ x_1 \text{ NAND } x_1)$

$(x_0 \oplus x_1) \equiv (x_0 \text{NAND } \bar{x}_1 . \text{ NAND}. \ \bar{x}_0 \text{ NAND } x_1)$

and complementing subformulae as necessary we obtain a NAND circuit of depth 2D which computes f. $\square$

Some immediate consequences of this result now follow.

Corollary 2.5

For all f in B_n,

$D_{U_2}(f) \leqslant 2.D_{B_2}(f)$

$D_{NAND}(f) \leqslant 2.D_{U_2}(f)$

Proof

$NAND \subset U_2 \subset B_2$

$\square$

Subsequent results will show that in each case the coefficient of 2 is best possible for any such relation which holds for all Boolean functions. When the set of possible inputs is restricted to X_n each of these upper bounds neei be increased by only 1. For example, we have

$$D_{NAND}(f) \leqslant 2.D_{B_2}(f) + 1$$

for all Boolean functions f, when X_n is the allowable set of inputs. To see that this upper bound is best possible of those which hold for all $n \geqslant 2$ we note that the binary functions $0, \oplus$ both require depth 1 over B_2 and depth 3 over NAND when X_n is the set of inputs.

Pratt (1975) considers the effect of basis on formula size and establishes that for all f in B_n,

$$F_{U_2}(f) = O((F_{B_2}(f))^k)$$

where $k = \log_3 10 \simeq 2{\cdot}095$.

2.2 EQUIVALENT BASES.

Definition

Two binary bases Ω_1, Ω_2 are equivalent with respect

<u>to depth</u> iff they both cover the same subset S of functions in B_n and for all f in S, $D_{\Omega_1}(f)$, $D_{\Omega_2}(f)$ are separated by at most an additive constant. Likewise, two binary bases Ω_1, Ω_2 are <u>equivalent w.r.t. circuit size</u> (<u>formula size</u>) iff they cover the same set S of n-argument Boolean functions and for each f in S, $C_{\Omega_1}(f)$, $C_{\Omega_2}(f)$ ($F_{\Omega_1}(f)$, $F_{\Omega_2}(f)$) are separated by at most a constant factor. $D[\Omega]$ will denote the <u>complexity class</u> of binary bases which are equivalent to Ω w.r.t. depth. Similarly for $C[\Omega]$, $F[\Omega]$.

It is known that all pairs of complete binary bases Ω_1, Ω_2 are equivalent with respect to circuit size since each complete basis Ω_1 can be replaced with another complete set of basic functions Ω_2 by building each element of Ω_1 with some fixed number of elements from Ω_2.

It is not difficult to see that the set of complete binary bases can be partitioned into $F[U_2]$ and $F[B_2]$. In the next section we note the disparity between these two complexity classes.

Determining the complexity classes of complete binary bases w.r.t. depth turns out to be considerably more difficult. In defining functional completeness we noted that B_2, $\{\wedge, \vee, \overline{\Pi}_0\}$ are two examples of complete bases while $M_2 = \{\wedge, \vee, \Pi_0, \Pi_1, 0, 1\}$ is an incomplete basis. The definition of completeness implies that, in particular, the constant functions $f(X_n) = 1$ and $f(X_n) = 0$ must be realizable with inputs from the set X_n. If this requirement is removed and it can be assumed that the constants 0 and 1 are available as inputs where necessary, then the basis is said to be <u>weak complete</u>. Note that a complete basis is weak complete, but as we shall see the converse is not necessarily true.

An example of a weak complete basis is provided by the well-known complement-free ring sum expansion due to Zhegalkin (1927)

$$f(X_n) = \bigoplus a_i \wedge x_{j1} \wedge x_{j2} \wedge \ldots\ldots \wedge x_{jk} \qquad (2.2)$$

where a_i is 0 or 1, $0 \leqslant i \leqslant 2^n - 1$, $0 \leqslant k \leqslant n$, $j_i < j_{i+1}$ and the set of variables $\langle x_{j1}, x_{j2}, \ldots, x_{jk} \rangle$ denotes a subset of k variables from X_n and $\bigoplus$ denotes the extended sum (modulo 2) operation. This expansion implies that $\{\wedge, \oplus\}$ is a weak complete basis but not a complete basis since the constant function $f(X_n) = 1$ cannot be realised using these operations. Another binary basis which is weak complete but not complete is $\{\rightarrow\}$.

Post (1941) has established necessary and sufficient conditions for a basis to be complete and Glushkov (1966) has used this result to formulate a similar criterion for weak completeness. We now give, without proof, an abbreviated account of these results. Some of the properties of basic functions which shall be used are:

<u>Property 1</u> (Monotonicity)

M_2 is precisely the set of <u>monotone</u> functions in B_2.

Property 2 (Linearity)

A function $f(X_2)$ in B_2 is said to be linear if its canonical expansion given by (2.2) has the form

$$f(X_2) = a_o \oplus a_1x_o \oplus a_2x_1$$

where a_i ($i = 0,1$ and 2) is 0 or 1.

Property 3 (Self-duality)

A function $f(X_2)$ is said to be self-dual if complementing its arguments results in the complementary function i.e.

$$f(X_2) = \overline{f}\,(\overline{x}_o, \overline{x}_1)$$

Property 4 (Zero preservation)

A function $f(X_2)$ is said to be a function preserving zero if

$$f(0,0) = 0$$

Property 5 (One preservation)

A function $f(X_2)$ is said to be a function preserving one if

$$f(1,1) = 1$$

These properties of functions in B_2 are summarized in the following table:

FUNCTION	1	2	3	4	5
0	1	1	0	1	0
1	1	1	0	0	1
π_0	1	1	1	1	1
π_1	1	1	1	1	1
$\overline{\pi}_0$	0	1	1	0	0
$\overline{\pi}_1$	0	1	1	0	0
$\wedge$	1	0	0	1	1
NAND	0	0	0	0	0
$\vee$	1	0	0	1	1
NOR	0	0	0	0	0
$\rightarrow$	0	0	0	0	1
$\leftarrow$	0	0	0	0	1
$\overline{\rightarrow}$	0	0	0	1	0
$\overline{\leftarrow}$	0	0	0	1	0
$\oplus$	0	1	0	1	0
$\equiv$	0	1	0	0	1

Classification of functions in B_2

Table 2

In this table each function occupies a row and if it has property i, 1 is entered in column i of the row associated with the function; otherwise the entry is 0.

<u>Theorem</u> (Post (1941))

A basis is complete if and only if for all i ($1 \leqslant i \leqslant 5$) it contains at least one function which does not have property i.

$\square$

<u>Theorem</u> (Glushkov (1966))

A basis is weak complete if and only if it contains at least one nonmonotone function and at least one nonlinear function.

$\square$

These results establish computable criteria for determining completeness and weak completeness. An immediate consequence of the former result is that the only binary functions which form complete bases on their own are NAND , NOR. This can be checked by consulting Table 2.

<u>Lemma 2.6</u>

Every nonmonotone function in B_2 satisfies at most one of properties 3, 4 and 5.

<u>Proof</u>

Immediate from definitions of the properties. Can be verified by inspection of Table 2.

$\square$

Definition

A (weak) complete basis is minimal if no proper subset of it forms a (weak) complete basis.

The result of Glushkov (1966) implies that a minimal weak complete basis can have at most two functions. An example of such a basis is provided by $\{\wedge, \oplus\}$. By combining Post's (1941) theorem and Lemma 2.6 we can show that a minimal complete basis has at most 3 functions. There are 32 minimal complete bases which we now tabulate.

$$\begin{array}{l}
\{\text{NAND}\} \\
\{\text{NOR}\} \\
\{0\} \cup \{\rightarrow \mid \leftarrow\} \\
\{1\} \cup \{\overline{\rightarrow} \mid \overline{\leftarrow}\} \\
\{\overline{\pi}_0\} \cup \{\wedge \mid \vee \mid \rightarrow \mid \leftarrow \mid \overline{\rightarrow} \mid \overline{\leftarrow}\} \\
\{\overline{\pi}_1\} \cup \{\wedge \mid \vee \mid \rightarrow \mid \leftarrow \mid \overline{\rightarrow} \mid \overline{\leftarrow}\} \\
\{\rightarrow\} \cup \{\overline{\rightarrow} \mid \overline{\leftarrow} \mid \oplus\} \\
\{\leftarrow\} \cup \{\overline{\rightarrow} \mid \overline{\leftarrow} \mid \oplus\} \\
\{\equiv\} \cup \{\overline{\rightarrow} \mid \overline{\leftarrow}\} \\
\{\wedge\} \cup \{\{0, \equiv\} \mid \{1, \oplus\} \mid \{\oplus, \equiv\}\} \\
\{\vee\} \cup \{\{0, \equiv\} \mid \{1, \oplus\} \mid \{\oplus, \equiv\}\}
\end{array}$$

where $\{1\} \cup \{\overline{\rightarrow} \mid \overline{\leftarrow}\}$ denotes the two bases $\{1, \overline{\rightarrow}\}$, $\{1, \overline{\leftarrow}\}$ and so on . Thus, there are two minimal complete bases of size 1, twenty-four of size 2 and six of size 3.

Theorem 2.7

Every minimal complete basis is in at least one of the following eight complexity classes:

$$D\left[\text{NAND}\right]$$
$$D\left[\{\rightarrow, 0\}\right]$$
$$D\left[\{\wedge, \overline{\Pi}_0\}\right]$$
$$D\left[\{\rightarrow, \overline{\rightarrow}\}\right]$$
$$D\left[\{\rightarrow, \oplus\}\right]$$
$$D\left[\{\wedge, \oplus, 1\}\right]$$
$$D\left[\{\wedge, \equiv, 0\}\right]$$
$$D\left[\{\wedge, \oplus, \equiv\}\right]$$

To establish this result, a number of simple facts will be needed. "Trivial" proofs are omitted and only the necessary result is given.

Definition

The dual $\hat{f}(X_2)$ of some function $f(X_2)$ in L_2 is defined by $\hat{f}(X_2) = \overline{f}(\overline{x}_0, \overline{x}_1)$ where $\overline{f}$ is the complement of f. The dual of a projection function (e.g. Π_0, $\overline{\Pi}_1$) is itself.

Fact 2.8

If Ω_1, Ω_2 are complete bases and by replacement of each function in Ω_1 with its dual we can obtain Ω_2, then

$$D\left[\Omega_1\right] = D\left[\Omega_2\right].$$

$\square$

For example, $\{\rightarrow, \oplus\}$ and $\{\overline{\leftarrow}, \equiv\}$ are in the same complexity class w.r.t. depth. Similarly, NAND is the dual of NOR and so NAND $\in D[\{\text{NOR}\}]$.

Fact 2.9

If replacement of some implication function in Ω_1 by the complement of its dual yields Ω_2, then

$$D[\Omega_1] = D[\Omega_2]$$

$\square$

For example, $\{\overline{\rightarrow}, 0\}$ and $\{\overline{\leftarrow}, 0\}$ are equivalent with respect to depth. Likewise, $\{\rightarrow, \overline{\rightarrow}\}$ and $\{\leftarrow, \overline{\rightarrow}\}$ are equivalent.

Fact 2.10

Replacement of one complemented projection function (i.e. $\overline{\Pi}_0$ or $\overline{\Pi}_1$) by another yields an equivalent basis w.r.t. depth.

$\square$

Note

Facts 2.9, 2.10 apply to all binary bases whereas Fact 2.8 does not apply to some incomplete bases. For example, $\wedge$ is the dual of $\vee$ but $\{\wedge\} \notin D[\{\vee\}]$.

Proof of Theorem 2.7

Noting Facts 2.8, 2.9 and 2.10, it remains to show that $D[\{\rightarrow, 0\}] = D[\{\rightarrow, \overline{\Pi}_0\}]$. It is evident that any circuit of depth Z over $\{\rightarrow, 0\}$ can be simulated by a circuit of depth $Z + O(1)$ over $\{\rightarrow, \overline{\Pi}_0\}$. In view of the identity $\overline{\Pi}_0(X_2) = \rightarrow(x_0, 0)$, there is a similar translation in the opposite direction.

$\square$

This result establishes an upper bound of 8 on the number of complexity classes of minimal complete bases w.r.t. depth. In the next section we prove a corresponding lower bound of 4 and consider the disparity between non-equivalent bases. We have not considered the problem of determining the number of complexity classes of complete bases (minimal and non-minimal) w.r.t. depth. However, it is conceivable that this number will be substantially larger than our upper bound of 8 for minimal bases.

We have already mentioned two important non-minimal bases, B_2 and U_2. Some simple propositions about $D\left[B_2\right]$, $D\left[U_2\right]$ are now given.

<u>Definition</u>

Every binary Boolean function which depends on both arguments is one of the following three types:

$\wedge$ - <u>type</u> $\quad x_0^* \wedge x_1^*$

$\vee$ - <u>type</u> $\quad x_0^* \vee x_1^*$

$\oplus$ - <u>type</u> $\quad x_0^* \oplus x_1$

where a starred argument represents either the argument or its complement.

<u>Proposition 2.11</u>

Every complete binary basis which contains at least one function from each of the above three types, is equivalent to B_2 w.r.t. depth.

Proof

For such a basis Ω, any circuit over B_2 can be simulated with a circuit over Ω by complementing subformulae as necessary. This simulation need not increase the circuit depth by more than an additive constant.

Any binary basis is a subset of B_2 and so simulation in the opposite direction is trivial.

$\square$

Proposition 2.12

Every complete binary basis which contains at least one $\wedge$-type and one $\vee$-type function, but no $\oplus$-type function, is equivalent to U_2 w.r.t. depth.

Proof

As in Prop. 2.11, noting the fact that any binary basis with no $\oplus$-type function is a subset of U_2.

$\square$

2.3 ORDERINGS ON COMPLETE BASES. We have already remarked that all complete bases are equivalent with respect to circuit size and have noted that the set of complete bases can be partitioned into $F[U_2]$ and $F[B_2]$. Pratt (1975) has established that the maximum disparity between these classes is at most $O(n^k)$ where $k=\log_3 10$, i.e. if $F_{B_2}(f) = n$ then $F_{U_2}(f) = O(n^k)$. The following result shows that this maximum disparity is at least order n^2.

Theorem 2.13 (Khrapchenko (1971))

$$F_{U_2}\left(\bigoplus_{i=0}^{n-1} x_i \right) \geq n^2$$

$$\square$$

The following theorem provides a functional characterization of shallow circuits over the (non-minimal) complete basis $\{\text{NAND}, \rightarrow\}$. A corollary of this result will be combined with Theorems 2.13 and 4.4 in order to prove a lower bound of 4 on the number of complexity classes of minimal complete bases w.r.t. depth.

Theorem 2.14

For all $k>0$, let a_k be the k^{th} Fibonacci number and and f be some function with circuit depth not more than k over the basis $\{\text{NAND}, \rightarrow\}$. Then f can be expressed as

$$\bigvee_j \bigwedge_{i \in Q_j} z_i$$

where each z_i is some input, and for all j, $|Q_j| \leq a_k$.

Furthermore, the complement of f can be expressed in this form, where each conjunction of inputs is of length not more than a_{k+1}.

Proof

By induction on k. As the basis of our induction, we note that the theorem is true for $k=1$. For $k>1$, assume it is true for $k=n$ and consider those functions f with circuit depth not more than $n+1$ over the basis $\{NAND, \rightarrow\}$. Any such f can be expressed either as $\bar{g} \vee h$ or as $\bar{g} \vee \bar{h}$, where g,h both have circuits of depth n over the basis $\{NAND, \rightarrow\}$. An application of the inductive hypothesis shows that any such f can be expressed as

$$\bigvee_j \bigwedge_{i \in Q_j} z_i$$

where each conjunction of inputs has length at most $\max\{a_n, a_{n+1}\} = a_{n+1}$.

Similarly, the complement of f may be expressed as $g \wedge \bar{h}$ or as $g \wedge h$. In this case the inductive hypothesis implies that $\bar{f}$ can be represented in the above form, with conjunctions of inputs whose length is not more than $\max\{a_n + a_{n+1}, 2a_n\} = a_{n+2}$.

Therefore, the theorem is true for $k=n+1$ and thus for all $k>0$.

□

<u>Corollary 2.15</u>

$$D_{\{NAND, \rightarrow\}}\left(\bigwedge_{i=0}^{n-1} x_i \right) \geq k.\log n - O(1)$$

where $k = \log_{\phi} 2 \simeq 1.44$ and ϕ is the golden ratio.

<u>Proof</u>

Immediate from the explicit formula for a_n given in the proof of Theorem 2.2

□

An immediate consequence of this result is the following lower bound for the minimal complete basis $\{\rightarrow, 0\}$.

<u>Corollary 2.16</u>

$$D_{\{\rightarrow,0\}}\left(\bigwedge_{i=0}^{n-1} x_i\right) \geq \log_{\phi} 2.\log n - O(1)$$
$$\square$$

It should be noted that the function $\mathrm{CONJ}^{(n)} = \bigwedge_{i=0}^{n-1} x_i$ requires only linear formula size over the basis $\{\rightarrow, 0\}$. Therefore, Corollaries 2.15 and 2.16 provide examples of lower bounds on circuit depth which could not be derived from corresponding bounds on formula size.

We now combine a number of results in order to prove

<u>Theorem 2.17</u>

There is a lower bound of 4 on the number of distinct complexity classes of minimal complete bases w.r.t. depth.

<u>Proof</u>

Note firstly that the lower bound of Corollary 2.16 is achievable to within an additive constant. This precise result on the depth of $\mathrm{CONJ}^{(n)}$ is now combined with other results for this function and for $\mathrm{SUM}^{(n)} = \bigoplus_{i=0}^{n-1} x_i$, and these are summarized in the following table :

Minimal Complete Basis	$\mathrm{CONJ}^{(n)}$	$\mathrm{SUM}^{(n)}$
$\{\mathrm{NAND}\}$	2 log n	2 log n
$\{\rightarrow, 0\}$	1.44 log n	$d \geq 2$ log n
$\{\wedge, \overline{\pi}_0\}$	log n	$d \geq 2$ log n
$\{\rightarrow, \overline{\rightarrow}\}$	log n	2 log n
$\{\rightarrow, \oplus\}$	$d \leq 1.44$ log n	log n
$\{\wedge, \oplus, 1\}$	log n	log n
$\{\wedge, \equiv, 0\}$	log n	log n
$\{\wedge, \oplus, \equiv\}$	log n	log n

where each entry denotes the depth (d) of the function over the corresponding minimal complete basis. Precise results are stated where these are known, otherwise a lower or upper bound is given. Additive constants are ignored throughout. Some of these results are obtained from lower bounds in 2.13, 2.16 and 4.4.

Two bases Ω_1, Ω_2 are inequivalent w.r.t. depth if there is some function f such that $D_{\Omega_1}(f)$ and $D_{\Omega_2}(f)$ differ by more than an additive constant. The above table shows that both $D[\text{NAND}]$ and $D[\{\rightarrow, 0\}]$ are inequivalent to all other complexity classes and inequivalent to each other. This yields a lower bound of 3. Finally we note that over $\{\wedge, \overline{\Pi}_0\}$ and $\{\rightarrow, \overline{\rightarrow}\}$, the depth of $SUM^{(n)}$ is at least $2 \log n - O(1)$ while over any basis which contains $\oplus$ or $\equiv$ the depth is at most $\log n + O(1)$. Therefore, both $\{\wedge, \overline{\Pi}_0\}$ and $\{\rightarrow, \overline{\rightarrow}\}$ are inequivalent to any of the bases $\{\rightarrow, \oplus\}$, $\{\wedge, \oplus, 1\}$, $\{\wedge, \equiv, 0\}$ and $\{\wedge, \oplus, \equiv\}$ and we have a lower bound of 4.

$\square$

We now consider the disparity among non-equivalent complete bases and investigate a natural ordering on the complexity classes of complete binary bases with respect to depth.

Definition

Let $D = \left\{ D\left[\Omega\right] \right\}$ where Ω ranges over all complete binary bases. We define a binary relation $\leqslant$ on the set D as follows.

For each pair $D\left[\Omega_1\right]$, $D\left[\Omega_2\right]$ in D, $D\left[\Omega_1\right] \leqslant D\left[\Omega_2\right]$ if for all f in B_n, $D_{\Omega_2}(f) \leqslant D_{\Omega_1}(f) + O(1)$.

E.g. $D\left[U_2\right] \leqslant D\left[B_2\right]$. To see this we merely note that $U_2 \subset B_2$. Theorem 2.13 shows that $D_{U_2}(SUM^{(n)}) \geqslant 2 \log n - O(1)$ and so the two complexity classes are not equivalent. Another example is $D\left[\left\{\rightarrow, \not\rightarrow\right\}\right] \leqslant D\left[U_2\right]$. However in this case the two complexity classes are identical (see Proposition 2.12). As a complexity class can be represented in a number of distinct ways, it will be convenient to have the following

Definition

The relation $<$ on the set of complete binary bases is defined as follows. If $D\left[\Omega_1\right] \leqslant D\left[\Omega_2\right]$ and Ω_1 is not equivalent to Ω_2 w.r.t. depth, then $\Omega_1 < \Omega_2$

E.g. NAND $< U_2$ as NAND $\subset U_2$ and Theorem 4.4 shows that $D_{NAND}(CONJ^{(n)}) = 2\left\lceil \log n \right\rceil$.

Definition

A partial order on a set Z is a binary relation R such that for each x,y and z in Z:

1. x R x is true (R is reflexive)

2. x R y and y R z imply x R z (R is transitive) and

3. x R y and y R x imply x = y (R is antisymmetric).

The relation $\leqslant$ on integers and the inclusion relation ($\subseteq$) on sets are two examples of partial orders. However, the relation $<$ on the set of integers is not a partial order.

Definition

A linear (or total) order on a set Z is a partial order R on Z such that for every pair of elements x,y in Z either x R y or y R x.

The relation $\leqslant$ on the set D induces an order on the set of complexity classes D. We can informally view this ordering in the following way. If $D[\Omega_1] \leqslant D[\Omega_2]$ then the "expressive power" of Ω_2 is greater than or equal to that of Ω_1 w.r.t. depth.

Lemma 2.18

The relation $\leqslant$ on the set D is a partial order.

Proof

The reflexive and transitive properties are easily verified.

If $D[\Omega_1] \neq D[\Omega_2]$ then without loss of generality we may assume that there is some function f in B_n such that

$$D_{\Omega_1}(f) > D_{\Omega_2}(f) + C$$

for any fixed constant C. This implies that the relation

$$D_{\Omega_1}(f) \leqslant D_{\Omega_2}(f) + O(1)$$

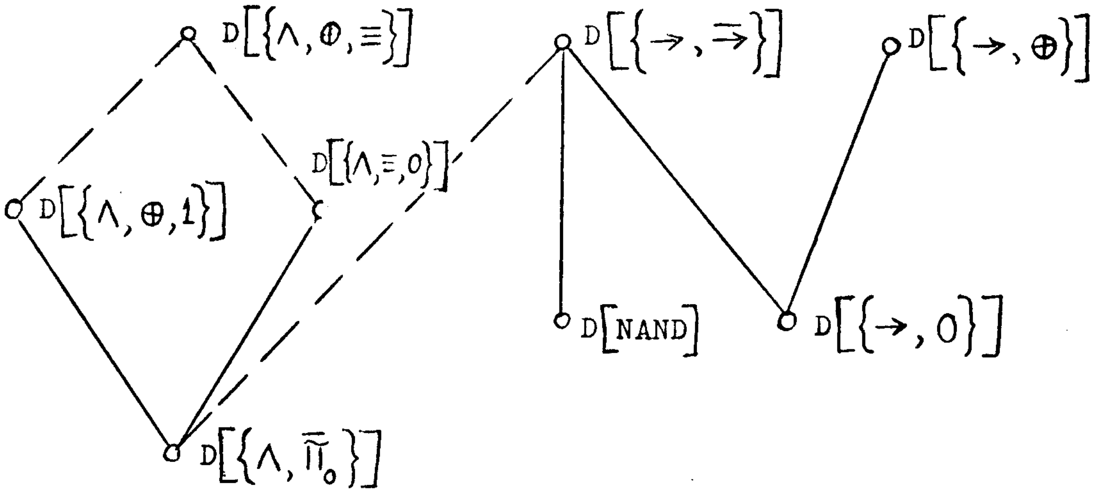
is false and consequently $D[\Omega_2] \leqslant D[\Omega_1]$ is false. This establishes that $\leqslant$ on the set D is an antisymmetric relation and proves the lemma.

$\square$

Conjecture

The set D is not totally ordered by the relation $\leqslant$. $\square$

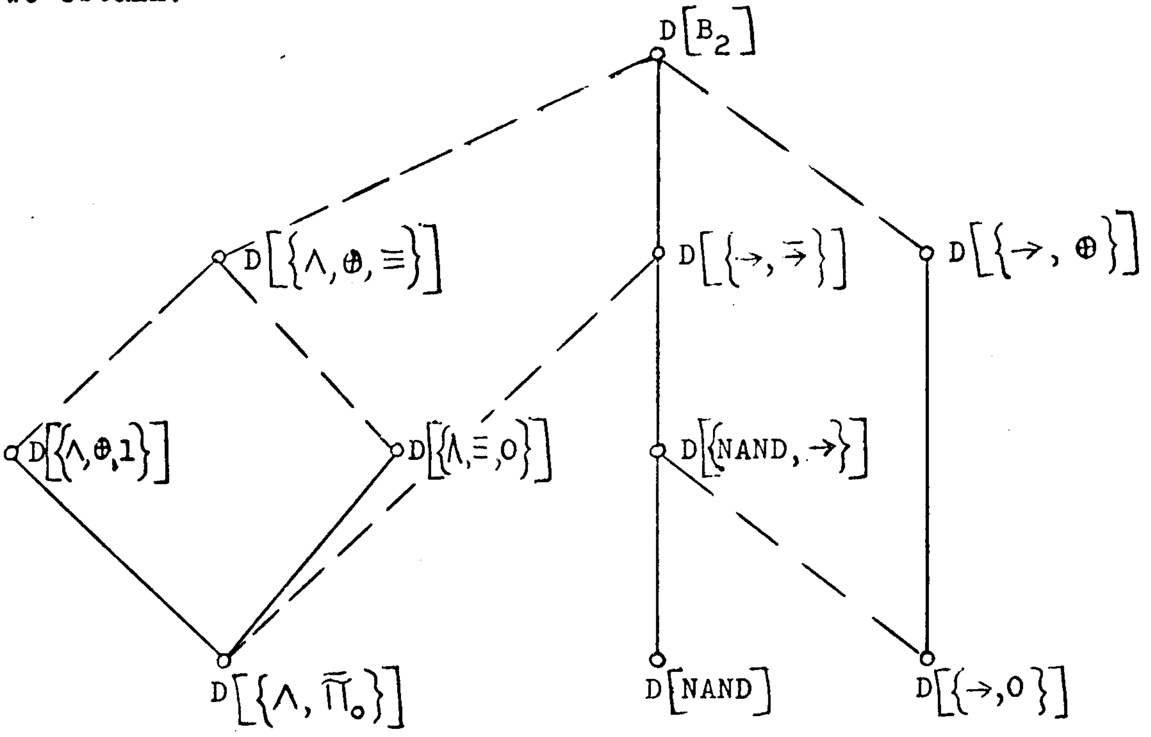
The partial order $\leqslant$ on the subset $\{D[\Omega]\}$ where Ω ranges over all minimal complete bases, may be conveniently depicted by the following diagram:


$D[\{\wedge, \oplus, \equiv\}]$
$D[\{\rightarrow, \Rightarrow\}]$
$D[\{\rightarrow, \oplus\}]$
$D[\{\wedge, \equiv, 0\}]$
$D[\{\wedge, \oplus, 1\}]$
$D[\text{NAND}]$
$D[\{\rightarrow, 0\}]$
$D[\{\wedge, \overline{\Pi}_0\}]$

where two classes $D[\Omega_1]$, $D[\Omega_2]$ are joined by a broken line if $D[\Omega_1] \leqslant D[\Omega_2]$ and by a full line if $\Omega_1 < \Omega_2$. E.g. the diagram shows that $\{\wedge, \overline{\Pi}_0\} < \{\wedge, \oplus, \equiv\}$ and $D[\{\wedge, \overline{\Pi}_0\}] \leqslant D[\{\rightarrow, \Rightarrow\}]$. The figure is not a

Hasse diagram since some pairs of classes may be equivalent. E.g. it is not known whether $D\left[\left\{\wedge, \overline{\Pi}_0\right\}\right]$ and $D\left[\left\{\rightarrow, \overline{\rightarrow}\right\}\right]$ are equivalent or not. The validity of the order depicted is easily verified from results given previously and from trivial simulations of one basis by another.

Extending this figure to include $D\left[B_2\right]$ and $D\left[\left\{\mathrm{NAND}, \rightarrow\right\}\right]$ we obtain:


$D[B_2]$
$D[\{\wedge, \oplus, \equiv\}]$
$D[\{\rightarrow, \overline{\rightarrow}\}]$
$D[\{\rightarrow, \oplus\}]$
$D[\{\wedge, \oplus, 1\}]$
$D[\{\wedge, \equiv, 0\}]$
$D[\{\mathrm{NAND}, \rightarrow\}]$
$D[\{\wedge, \overline{\Pi}_0\}]$
$D[\mathrm{NAND}]$
$D[\{\rightarrow, 0\}]$

and again the order depicted is easily verified from previous results and simple simulations. Note that the relation $<$ on the set of complete binary bases is transitive and so e.g. $\mathrm{NAND} < B_2$.

We now consider the maximum disparity between some non-equivalent complexity classes.

Lemma 2.19

For each Ω_1 in $\{NAND, \{\rightarrow, \nrightarrow\}\}$ and Ω_2 in $\{\{\rightarrow, \nrightarrow\}, B_2\}$

where $\Omega_1 \neq \Omega_2$, we have

i) For all f in B_n,

$$D_{\Omega_1}(f) \leqslant 2.D_{\Omega_2}(f) + O(1)$$

and

ii) There is some f in B_n such that

$$D_{\Omega_1}(f) \geqslant 2.D_{\Omega_2}(f) - O(1)$$

Proof

Noting that $\{\rightarrow, \nrightarrow\} \in D[U_2]$ by Proposition 2.12, the lemma follows from results 2.4, 2.5, 2.13 and 4.4.

$\square$

Comment

This result establishes precise bounds on the maximum disparity between any pair of bases from B_2, U_2 and NAND. It also raises the question of how disparate are any two complexity classes $D[\Omega_1]$, $D[\Omega_2]$ when $D[\Omega_1] \leqslant D[\Omega_2]$. It is not difficult to see that this question is essentially concerned with the maximum disparity between B_2 and any minimal complete basis. Taking into account the partial order on complexity classes depicted above, the problem then reduces to a consideration of the maximum disparity between B_2 and the bases $\{\wedge, \overline{\Pi}_0\}$, $\{\rightarrow, 0\}$ and NAND. It seems likely that this maximum disparity is greater than the maximum disparity between B_2 and NAND which was determined precisely in Lemma 2.19. However, at present this is an open problem.

2.4 RELATIONS FOR MONOTONE FUNCTIONS.

Let X_n be the set of possible inputs. The result of Preparata and Muller (1976) establishes that for all monotone functions f in M_n,

$$D_{M_2}(f) \leqslant 1.81 \log F_{M_2}(f) + O(1)$$

However, as in the case of arbitrary Boolean functions, much less is known about the relationship between circuit size and depth. For monotone circuits the technique of Paterson and Valiant (1976) can be used with only a minor modification. The technique employs the identity

$$f = \bigvee_{\underset{\sim}{c}} \delta_{\underset{\sim}{c}} \wedge f_{\underset{\sim}{c}}$$

to produce an alternative circuit for some f in B_n, where $\underset{\sim}{c}$ is an m-tuple of Boolean values and $\delta_{\underset{\sim}{c}}$ is a conjunction of functions associated with the values of $\underset{\sim}{c}$. In the case of monotone circuits we choose $\delta_{\underset{\sim}{c}}$ to be a conjunction of those functions associated with the value 1 in $\underset{\sim}{c}$. With this modification the technique yields monotone circuits with depth not greater than those obtained by the general method. Thus we have the following relation for all monotone functions f in M_n,

$$D_{M_2}(f) = O(\, C_{M_2}(f) \,/\, \log C_{M_2}(f)\,)$$

A relation between the monotone circuit size and monotone formula size of all f in M_n can be obtained from this result by noting that $D_{M_2}(f) \geqslant \log F_{M_2}(f)$.

An interesting question is whether monotone functions can be realized more economically if non-monotone basis functions are used. The present state of our knowledge about the effect of not using negations can best be appreciated by considering the following

<u>Open problem</u>

Demonstrate some f in M_n such that,

(i) $C_{U_2}(f) = o(C_{M_2}(f))$

(ii) $F_{U_2}(f) = o(F_{M_2}(f))$

or

(iii) $(D_{M_2}(f) - D_{U_2}(f)) \rightarrow \infty$ as $n \rightarrow \infty$.

Several such results have been established for families of Boolean functions. For example, Paterson (1975) shows that any computation of the product of two n x n Boolean matrices by a circuit over the basis $\{\wedge, \vee\}$ requires at least n^3 $\wedge$-gates and $n^3 - n^2$ $\vee$-gates. In contrast to this lower bound, it is known that

$$O(n^{\log 7} (\log n)^{1+\epsilon}) \quad \text{for any} \quad \epsilon > 0$$

is an upper bound on the circuit size of n x n x n Boolean matrix product over any complete binary basis.

3. BOUNDS FOR "ALMOST ALL" FUNCTIONS

Convention

A Boolean function f is explicitly defined if and only if the truth-table of f can be generated by a multitape Turing machine in time polynomial in the length of the truth-table.

For example, the function associated with the clique problem is explicitly defined. And in fact, most familiar Boolean functions have this property. However, despite considerable effort no one has yet established a strong lower bound on the complexity of any particular function which is explicitly defined. To date, only lower bounds which are linear in n, where n is the size of input, have been proved for circuit size. Likewise, only lower bounds of the form $C.\log n$ where C is some constant have been established on the circuit depth of explicitly defined functions.

These apparently poor lower bounds raise the question of whether there are Boolean functions whose complexity is high. In this chapter we consider the maximum complexity of any Boolean function and note that this is not much higher than the inherent complexity of 'almost all' Boolean functions. Similar problems for the subset of monotone functions are also considered.

3.1 SCHEMES. Several constructions to be described in this chapter have the property of being 'uniform'; The same directed graph with the same assignment of arguments to inputs is used for all the functions concerned, the necessary variation being only in the assignment of base functions to the gates. We formalise this notion of uniformity in our definition of 'circuit scheme'.

<u>Definition</u>

A <u>circuit</u> <u>scheme</u> (<u>formula</u> <u>scheme</u>) is a circuit (formula) in which the gates are left unspecified i.e. there are no basis functions associated with the gates. X_n is the set of possible inputs to schemes.

Let $C_n \subseteq B_n$ and $b \subseteq B_2$. A circuit scheme β <u>covers C_n over basis b</u> if for each f in C_n, there is an assignment of functions from b to the gates of β such that the resulting circuit computes f. Figure 2 shows a formula scheme which covers B_3 over the basis B_2.

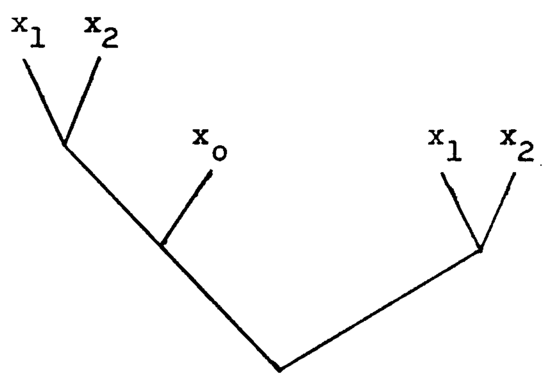


Fig. 2. A scheme for B_3

This follows from the expansion

$$f(X_3) = (x_o \wedge f_1 (x_1,x_2)) \oplus f_o (x_1,x_2)$$

where $f_o (x_1,x_2) = f(0,x_1,x_2)$ and

$f_1(x_1,x_2) = f(1,x_1,x_2) \oplus f(0,x_1,x_2)$. It has been verified that this is the unique formula scheme (to within obvious symmetries) with fewer than five gates that covers B_3. Its depth of 3 is therefore optimal.

Our interest in this specialized model of computation stems from the fact that we can obtain lower bounds on scheme complexity using simple counting arguments.

We must distinguish the notions of complet: bases for circuits and for schemes. For example, NAND is complete for circuits but no single element basis can be complete for schemes. This difficulty can be resolved by adding a projection function ,e.g. $\{ \Pi_o, \text{NAND} \}$ is complete for schemes.

3.2 MAXIMAL BOUNDS FOR B_n. In this section we consider the maximum complexity of any Boolean function and note the complexity of 'almost all' functions.

If we write $C_\Omega(S)$ for $\max \{ C_\Omega(f) \mid f \in S \}$ and similarly for $D_\Omega(S)$ and $F_\Omega(S)$ then we have the following classical results for all complete binary bases Ω,

<u>Theorem</u> (Lupanov (1958), Lupanov (1962))

$$C_\Omega(B_n) = O(2^n/n)$$

$$F_\Omega(B_n) = O(2^n/\log n)$$

$\square$

Note The constructions which establish these upper bounds are not uniform for all Boolean functions (i.e. they are not schemes). In fact, a simple counting argument yields a lower bound of 2^{n-2} on the number of gates in any circuit scheme which covers B_n over the basis B_2. We merely note that any circuit scheme with q gates can cover a set of at most $|B_2|^q$ different functions. As $|B_n| = 2^{2^n}$, any scheme of size q which covers B_n must satisfy $16^q \geqslant 2^{2^n}$, $q \geqslant 2^{n-2}$.

A counting argument due to Shannon (1949) can be used to show that the fraction of functions in B_n with circuit size not at least proportional to $2^n/n$, tends to 0 with increasing n. Likewise, a counting argument of Riordan and Shannon (1942) can be used to show that 'almost all' Boolean functions require formulae of size at least proportional to $2^n/\log n$. Each of these lower bounds holds over any complete binary basis. An immediate consequence of the lower bound on formula size is that $D_{B_2}(B_n) \geqslant n - \log\log n - O(1)$. This lower bound also holds for 'almost all' functions in B_n, i.e. there is a constant C such that,

$$\left|\left\{ f \in B_n \;\middle|\; D_{B_2}(f) \leq n - \log\log n - C\right\}\right| = o(|B_n|)$$

Theorem 3.1

Any circuit scheme which covers B_n over any basis $b \subseteq B_2$ has depth at least n-1. Furthermore if $|b| \leqslant 4$ or $|b| = 2$ the depth is at least n or n + 1 respectively.

Proof

A scheme of depth D has at most 2^D-1 gates, and so by varying the assignment to gates from b it can cover a set of at most $|b|^{2^D-1}$ different functions. Since $|B_n| = 2^{2^n}$ we have

$$|b|^{2^D-1} \geqslant 2^{2^n}$$

which yields the stated bounds

$\square$

These lower bound results raise the problem of finding an upper bound on the depth of all Boolean functions. Preparata and Muller (1971) give the following upper bounds on $D_{B_2}(B_n)$ for specific values of n,

$$\begin{array}{lll} n & \text{for} & n \leqslant 8 \\ n+1 & \text{for} & n \leqslant 2^8 + 8 = 264 \\ n+2 & \text{for} & n \leqslant 2^{264} + 264 \\ \text{etc.} & & \end{array}$$

while Spira (1971) shows that over the basis U_2 any function in B_n has a circuit of depth $n + \log^* n$ where

$$\log^* n = (\text{if } n \leqslant 1 \text{ then } 0 \text{ else } \log^*(\log n) + 1)$$

Thus we have

$$n\text{-}\log\log n - O(1) \leqslant D_{B_2}(B_n) \leqslant n + \log^* n$$

We now describe a construction which improves the upper bound to n + 1 and gives an upper bound of n + 3 for the restricted basis U_2. Furthermore, the construction is a scheme. Theorem 3.1 shows that for this specialized model of computation it achieves the optimal depth to within an additive constant.

Theorem 3.2

For all $n > 0$, there is a formula scheme with depth $n + 1$ which covers B_n over B_2.

$\square$

We shall give here an informal account of the construction. (A full proof of the theorem is given in McColl and Paterson (1975)). Our starting point is a pair of familiar dual expansions for Boolean functions. Let $Y = \langle y_0, \ldots, y_{k-1} \rangle$ and $Z = \langle z_0, \ldots, z_{m-1} \rangle$ be sets of binary variables. Any function $f(Y,Z)$ in B_{k+m} may be expressed as a disjunctive expansion about Z by

$$f\,(Y,Z) = \bigvee_{c \in \{0,1\}^m} \delta_c(Z) \wedge f\,(Y,c)$$

where $\delta_c(Z) = 1$ iff $Z = c$.

The dual conjunctive expansion about Z is

$$f\,(Y,Z) = \bigwedge_{c} \bar{\delta}_c(Z) \vee f\,(Y,c)$$

where $\bar{\delta}_c$ is the complement of δ_c.

Each δ or $\bar{\delta}$ term requires a formula of depth only $\lceil \log m \rceil$ and in each case the total depth used exceeds the maximum for δ_c, $\bar{\delta}_c$ and $f\,(Y,c)$ by $m + 1$. The outer disjunctions or conjunctions over 2^m subformulae need depth m and one extra level is used for the single conjunction or disjunction used to attach the δ's or $\bar{\delta}$'s. It is the accumulation of these extra single levels in a recursive expansion about successive subsets of arguments which accounts for the $\log^* n$ term in Spira's bound. We plan to avoid these increments.

Consider one term $\delta_c(Z) \wedge f\ (Y,c)$ of the disjunctive expansion. We may ensure that $f\ (Y,c)$ is expressed as a conjunction of many small terms by using the conjunctive expansion for the next subset of arguments. Using the associativity of conjunction we might attempt to reassociate δ_c into $f\ (Y,c)$ but unfortunately the number of subterms of $f\ (Y,c)$ will be exactly a power of two. Our seemingly reckless solution is to discard one of these terms to make room for δ_c, and to be content with an "approximation" to the original function. To accomplish this ruse for each expansion we alternate disjunctive and conjunctive expansions about successive subsets of variables. The result of this first construction will be a formula of depth only n, but it will represent merely an approximation to the required function.

Rather surprisingly we are able to show that the required function can be derived as the sum (modulo 2) of this "approximation" and a second function which we can generate using the whole construction recursively in depth n also. The result is therefore of depth n + 1.

We shall describe our construction in terms of formulae rather than more abstractly as schemes. It will be clear throughout however that the formulae are uniform for all Boolean functions.

To define the subsets of arguments for the expansion, let R_o, R_1, ..., R_p be a partition of X_n with $|R_i| = r_i$ for all i. We shall use the simple sequence $\langle r_o, \ldots, r_p \rangle$ defined by

$$r_o = 2$$
$$r_i = i + 1 \quad \text{for} \quad o < i < p$$
$$r_p = n - S_{p-1} \text{ where } S_m = \sum_{i=o}^{m} r_i$$

and where p is maximal such that

$$\frac{p(p+1)}{2} + 1 < n.$$

For example, if n = 17 we have $\langle 2,2,3,4,5,1 \rangle$.

The following definition allows us to describe the kind of function which will be used as an 'approximation' to the required function.

<u>Definition</u>

Given $S = \{R_1, \ldots, R_k\}$ where $R_j \subseteq X_n$ for all $1 \leqslant j \leqslant k$, we define $g(X_n)$ to be <u>S-simple</u> if

$g(X_n) = 0$ whenever $R_j = \mathbf{O}$ for some $R_j \in S$,

where $\mathbf{O} = \langle o, o, \ldots, o \rangle$.

Having given the rationale for our construction, we shall merely state

<u>Lemma 3.3</u>

For $n > 4$, every $\{R_1, \ldots, R_{p-1}\}$ - simple function $g(X_n)$ has a formula of depth n.

$\square$

For details of the proof, see McColl and Paterson (1975). Some attention must be paid to the sequence of cardinalities of the expansion subsets, and the way in which a term is omitted from the expansions is not quite straightforward.

Given Lemma 3.3, it remains to be shown how formulae for arbitrary functions can be derived from the construction for simple functions.

<u>Lemma 3.4</u>

Suppose $R_1, \ldots, R_k$ are disjoint subsets of X_n. For all $f\ (X_n)$, there exists $f_1(X_n - R_1), \ldots, f_k(X_n - R_k)$ such that

$$g_k\ (X_n) = f \oplus \bigoplus_{i=1}^{k} f_i \quad \text{is } \{R_1, \ldots, R_k\}\text{-simple.}$$

<u>Proof</u>

This is by induction on k. The lemma holds trivially for $k = 0$. Let $k > 0$, and suppose the result is true for $k - 1$. Then, there exists $f_1\ (X_n - R_1), \ldots, f_{k-1}\ (X_n - R_{k-1})$ such that for all j, $1 \leqslant j < k$,

$$R_j = O \Rightarrow g_{k-1}\ (X_n) = f \oplus \bigoplus_{i=1}^{k-1} f_i = 0$$

We define $f_k\ (X_n - R_k) = 0$ if $\exists\, i,\ 1 \leqslant i < k,\ R_i = O$

$= g_{k-1}$ with arguments R_k set to O, otherwise.

It is evident that g_k has the required property.

$\square$

The main result has now been prepared for.

Proof of Theorem 3.2

Schemes for B_1, B_2 are obvious, while for B_3, B_4 expansions can be made about 1 and 2 arguments respectively to yield schemes of depth 3 and 4.

By the previous lemma and properties of $\oplus$, any function $f\ (X_n)$ may be expressed as

$$g(X_n) \oplus \bigoplus_{i=1}^{p-1} f_i(X_n - R_i)$$

where $g(X_n)$ is $\left\{R_1, \ldots, R_{p-1}\right\}$-simple, and the $\left|R_i\right| = r_i$ are defined as above. i.e. $\langle r_0, r_1, \ldots\rangle = \langle 2,2,3,4,\ldots\rangle$.

For $n > 4$, Lemma 3.3 yields a formula of depth n for $g(X_n)$, to which we must "add" appropriate functions $f_1, \ldots, f_{p-1}$ where f_i has $n_i = n - i - 1$ arguments. Whenever $n_i \leqslant 4$, a formula for f_i is constructed directly, otherwise the whole construction is used recursively to yield a formula of depth $n_i + 1 = n-i$.

Thus f is expressible as

$$g(X_n) \oplus \bigoplus_{i=1}^{p-1} f_i(X_n - R_i)$$

or, after reassociation, as

$$g \oplus (f_1 \oplus (f_2 \oplus \ldots. \oplus f_{p-1}))\ \ldots.)$$

Since f_i has depth n-i for i=1,..., p-1, the latter represents a formula of depth n + 1. It is evident that the construction is uniform for all Boolean functions and thus yields a scheme. $\square$

In the previous chapter we defined three types of basis functions: $\wedge$-type, $\vee$-type and $\oplus$-type. Provided the basis b permits a scheme to cover B_2 and contains at least one function from each of these three types, the construction can be followed more or less as before, complementing subformulae as necessary to achieve an upper bound of n + (depth of a scheme to cover B_2).

For the unate basis U_2 we may replace $\oplus$ by

$$x_0 \oplus x_1 = (x_0 \wedge \bar{x}_1) \vee (\bar{x}_0 \wedge x_1)$$

In order to fit in the correcting functions efficiently we choose a new sequence

$$\langle r_0, r_1, r_2, \ldots \rangle = \langle 2,2,4,6,8,10, \ldots \rangle$$

so that each f_i contains $\underline{2}$ fewer arguments than the previous one. The result is a scheme of depth $n + 3$ which covers B_n over U_2.

<u>Remark</u>

In outlining the construction we used a sequence $\langle 2,2,3,4,5, \ldots . \rangle$, but a sequence which grows much faster could be used instead. The effect of the choice of sequence on formula size has not been considered. However, as it is a scheme the possible size is within 2^{n-2} and 2^{n+1}. The construction of Lupanov (1962) yields formulae of size $O(2^n/\log n)$ for all Boolean functions, though not of course using schemes. This raises the following question.

Open problem

$$D_{B_2}(B_n) \geqslant n - O(1) \text{ ?}$$

i.e. does a lower bound of n-O(1) hold for formulae as well as schemes?

Another interesting question is whether an upper bound of n+O(1) holds for all complete bases. It has been shown that such a bound holds for every basis in $D[B_2] \cup D[U_2]$. However, for schemes over the basis $\{NAND, \Pi_0\}$ we have, at present, achieved no better than $n + O(\log^* n)$. We now describe the construction which yields this upper bound. As before, we shall describe it in terms of formulae although it will be evident that the formulae are uniform for all f in B_n.

Lemma 3.5

$$D_{NAND}(B_{n+2k}) \leqslant 2k + 2 + \max\left\{ D_{NAND}(B_n), 2.88 \log(2k) + O(1) \right\}$$

Proof

Any $f(X_{n+2k}) \in B_{n+2k}$ may be expressed as

$$(x_0 \wedge (x_1 \vee f^{10})) \text{ NAND } (\bar{x}_1 \vee f^{11}) \text{ .NAND. } (\bar{x}_0 \wedge (x_1 \vee f^{00})), \text{ NAND } (\bar{x}_1 \vee f^{01})$$

where $f^{pq} = f\ (p, q, x_2, x_3, \ldots, x_{n+2k-1})$

Applying this identity recursively to every f^{pq}, we get an identity which shows that any $f\ (X_{n+2k})$ can be computed by a NAND formula of depth 2k where each input is a formula which computes some function g_i.

Each such g_i may be expressed as a formula β_i over the basis B_2 in which $x_0, x_1, \ldots, x_{2k-2}$ and x_{2k-1} appear at most once and which contains exactly one occurrence of a subformula for some function $A_i\ (x_{2k}, x_{2k+1}, \ldots, x_{n+2k-1}) \in B_n$.

By defining $g_i\big|_{A_i=c}$ to be the function computed by β_i with the subformula for A_i replaced by the constant c, we have the identity

$$g_i = A_i \text{ NAND } g_i\big|_{A_i = 1} \text{ .NAND. } \bar{A}_i \text{ NAND } g_i\big|_{A_i = 0}$$

which yields the recurrence

$$D_{NAND}(B_{n+2k}) \leqslant 2k+2+\max_i \Big\{ D_{NAND}(A_i), D_{NAND}(\bar{A}_i), D_{NAND}(g_i\big|_{A_i=1}), D_{NAND}(g_i\big|_{A_i=0}) \Big\}$$

The result then follows by noting Theorem 2.2

$\square$

<u>Theorem 3.6</u>

For all $n > 1$, $D_{NAND}(B_n) \leqslant n + O(\log^* n)$

<u>Proof</u>

Let $2k = \lfloor c^n - n \rfloor$ in the recurrence of the previous lemma. Then, we have

$$D_{NAND}(B_{C^n}) \leqslant C^n - n + 2 + \max\left\{ D_{NAND}(B_n),\ 2.88\ n \log C + O(1) \right\}$$

If we choose the constant C such that $2.88 \log C < 1$, then

$$D_{NAND}(B_{C^n}) - C^n \leqslant 2 + D_{NAND}(B_n) - n$$

Let $d(n) = D_{NAND}(B_n) - n$, then

$$d(C^n) \leqslant 2 + d(n)$$

$$d\left(\left. C^{C^{C^{\cdot^{\cdot^{\cdot^{C}}}}}} \right\} r \right) \leqslant 2r + k \quad \text{for some constant } k.$$

If $n = \left. C^{C^{C^{\cdot^{\cdot^{\cdot^{C}}}}}} \right\} r$ for some $C > 1$, then $r = O(\log^* n)$

$\square$

<u>Conjecture</u>

$$D_{NAND}(B_n) \leqslant n + O(1)$$

<u>Definition</u>

Let $\Delta = U_2 - \{\wedge, \vee, \text{NAND}, \text{NOR}\}$ and B^{Δ} be the set of binary bases $\{B\}$ where $B \subset \Delta$, $B \not\leq D[U_2]$ and B is complete. E.g. $\{\rightarrow, \leftarrow, 0, 1, \overline{\Pi}_0\} \in B^{\Delta}$ whereas $\{\rightarrow, \not\Rightarrow\}$ is not.

Theorem 3.7

For all complete binary bases excluding those in B^{Δ}, there is an upper bound of $n + O(\log^* n)$ on the depth of all Boolean functions.

Proof

The basic technique for deriving efficient upper bounds on the depth of all Boolean functions is expansion about a subset of the variables, together with recursive use of the method for the remaining variables.

Disjunctive expansion

Each binary function in B_2 can be computed over any complete basis within constant depth. Thus, by choosing the subset to have about $n - \log n$ variables, one immediately gets an upper bound of $n + O(\log^* n)$ as a corollary of Spira's (1971) result, for all complete bases Ω where

$D_{\Omega}(\bigwedge_{i=0}^{n-1} x_i) = \log n + O(1)$, since the recursion goes to depth $\log^* n$.

Ring-sum expansion

Let $Y = \langle y_0, \ldots, y_{k-1}\rangle$ and $Z = \langle z_0, \ldots, z_{m-1}\rangle$ be sets of binary variables. Any function $f(Y,Z)$ in B_{k+m} may be expressed as a ring-sum expansion about Z by

$$f(Y,Z) = \bigoplus_{C \in \{0,1\}^m} \delta_C(Z) \wedge f(Y,C)$$

where $\delta_C(Z) = \bigwedge_{C_i=1} Z_i$

Using this expansion recursively and choosing the subset to have about n-log n variables we can achieve an upper bound of $n + O(\log^* n)$ for any complete basis Ω, where

$$D_\Omega(\bigoplus_{i=0}^{n-1} x_i) = \log n + O(1) \text{ and}$$

$$D_\Omega(\bigwedge_{i=0}^{n-1} x_i) = C.\log n$$ for some constant C. Note that this upper bound holds for bases such as $\{\rightarrow, \oplus\}$. The analysis of the recurrence in this case is similar to that in the proof of Theorem 3.6.

The theorem then follows by noting the result of Theorem 3.6 which establishes an upper bound of $n + O(\log^* n)$ for the basis NAND and by considering the complexity classes of minimal complete bases which were derived in Theorem 2.7.

□

In section 2.3 we derived precise bounds on the depth of $\bigwedge_{i=0}^{n-1} x_i$ over all bases in B^Δ. These can be used to show that there is an upper bound of $1.44\, n + o(n)$ on the depth of all Boolean functions, either by using a similar recursive construction or by simulating disjunctive normal form. The low order term will be $O(\log^* n)$ and $O(\log n)$ respectively. This gives an upper bound on the maximal depth of any Boolean function over any complete basis.

We have seen that for each of the three important complexity measures there are surprisingly precise results on the complexity of "almost all" Boolean functions.

The original motivation for studies of circuit complexity was to obtain a satisfactory solution to the so-called minimization problem , i.e. given a Boolean function, find a minimal circuit which represents it. For this problem we have the trivial solution in which we order circuits according to complexity, and then search all circuits up to complexity C until we find one which represents the function concerned. In this way we can always find a minimal circuit, but since there are 2^{2^n} functions in B_n this approach is not feasible as an impossibly large number of circuits might have to be compared.

In fact, there is reason to believe that no feasible solution (i.e. one which takes at most polynomial time) exists for the minimization problem. Cook (1971) has given strong evidence which suggests that a simpler problem requires nonpolynomial time. The problem is that of recognizing whether a certain disjunctive normal form (for a Boolean function) represents the constant 1. Note that a fast algorithm for the minimization problem would give us also a fast constant recognizer. Thus it seems likely that any exact procedure for the minimization problem will be comparable (in terms of computational complexity) to an exhaustive search among circuits.

As no satisfactory solution to the minimization problem seems likely, present research is directed towards establishing bounds on the complexity of Boolean functions. In subsequent chapters we pursue this line of investigation and prove a number of small lower bounds on the depth of explicitly defined functions. The fact that we have, at present, only small lower bounds for explicitly defined functions raises the question of whether there are functions of intermediate complexity, e.g. is there a function f in B_n for which $C_{B_2}(f) = k.n^2$ for some constant k ? In chapter 6 we note that several large sets of Boolean functions form reasonably uniform hierarchies with respect to the three important measures of circuit complexity.

But first we show that similar results to those obtained for "almost all" functions in B_n can be obtained for the important subset of monotone functions.

3.3 "ALMOST ALL" MONOTONE FUNCTIONS.

In this section we focus our attention on the maximum circuit depth of any monotone function and on the depth of "almost all" functions in M_n. For an excellent account of the corresponding formula size and circuit size problems, see Pippenger (1976).

Gilbert (1954) established a lower bound of 2^{C_n} on the number of monotone increasing Boolean functions of n arguments, where

$$C_n = \binom{n}{\lfloor n/2 \rfloor} = k.\frac{2^n}{\sqrt{n}} + o\left(\frac{2^n}{\sqrt{n}}\right)$$

and k is some constant.

<u>Lemma 3.8</u>

There is a constant C such that,

$$\left|\left\{ f \in M_n \;\middle|\; D_{B_2}(f) \leqslant n - \tfrac{1}{2}\log n - \log\log n - C \right\}\right| = o\left(\left|M_n\right|\right)$$

<u>Proof</u>

$$\frac{\left|\left\{ f \in M_n \;\middle|\; D_{B_2}(f) \leqslant n\,(1-\epsilon) \right\}\right|}{\left|M_n\right|}$$

$$\leqslant \frac{\left|B_2\right|^{2^{n(1-\epsilon)}} {}^{-1}.\; n^{2^{n(1-\epsilon)}}}{\left|M_n\right|} = \frac{(16.n)^{2^{n(1-\epsilon)}}}{16.\left|M_n\right|}$$

Therefore, it suffices to show that

$$\frac{2^{n(1-\epsilon)}\log(16.n)}{\log\left|M_n\right| + 4} \rightarrow 0 \quad \text{as } n \rightarrow \infty$$

Gilbert's lower bound shows that there is a constant C such that the fraction of monotone functions f in M_n, for which

$$D_{B_2}(f) \leqslant n\,(1-\epsilon)$$

tends to 0 with increasing n, whenever

$$\epsilon . n \geq \tfrac{1}{2}\log n + \log\log n + C$$

$\square$

In Lemma 3.8 we proved a lower bound on the depth of 'almost all' monotone functions by a simple counting argument. The size of lower bound obtained by such a counting argument depends solely on the size of the subset of functions considered.

Kleitman (1969) has shown that Gilbert's lower bound cannot be substantially improved, therefore we must use a different approach in order to improve the lower bound of Lemma 3.8.

<u>Lemma 3.9</u>

Any circuit scheme which covers M_n over the basis B_2 has depth at least $n - \frac{1}{2}\log n - O(1)$.

<u>Proof</u>

As in Theorem 3.1, noting the lower bound on the number of monotone functions.

$\square$

We now describe a scheme of depth n which covers M_n over the basis M_2.

<u>Theorem 3.10</u>

For all $n > 0$, there is a circuit scheme of depth n which covers M_n over the basis M_2.

The construction shall be described in terms of formulae rather than as a scheme. However, it will be evident that the formulae are uniform for all monotone functions. Since we shall consider formulae rather than schemes, it will be convenient to prove the result as the following

<u>Theorem 3.11</u>

For all $n > 0$, every $f(X_n) \in M_n$ can be expressed as a disjunction of $n - 1$ monotone subformulae of depth $1,2,3,\ldots,n-1$. Alternatively, f may be expressed as a conjunction of $n - 1$ such subformulae.

<u>Proof</u>

We proceed by induction on n. For any $f(X_n)$, let

$f^{x_i = C}$ denote $f(x_0, x_1, \ldots, x_{i-1}, C, x_{i+1}, \ldots, x_{n-1})$, where $C \in \{0,1\}$.

This definition implies that if $f(X_n) \in M_n$, then

$f^{x_i = 0}$, $f^{x_i = 1} \in M_{n-1}$.

Any $f(X_n) \in M_n$ can be expanded about an argument x_i using

either of the identities $f(X_n) = (x_i \wedge f^{x_i=1}) \vee f^{x_i=0}$ (3.1)

$f(X_n) = (x_i \vee f^{x_i=0}) \wedge f^{x_i=1}$ (3.2)

For $n < 3$ the theorem is obvious, while for $n = 3$, it follows from (3.1), (3.2).

For $n = k$, $k > 3$, assume it is true for $n = k - 1$.

Then by (3.1), any $f(X_k) \in M_k$ can be expressed as

$(x_i \wedge f^{x_i=1}) \vee f^{x_i=0}$. By the induction hypothesis,

$f^{x_i=1}$ $(f^{x_i=0})$ can be expressed as a conjunction (disjunction) of $k - 2$ subformulae. Therefore, the x_i may be reassociated, using the associativity of conjunction, to yield a formula for $f(X_k)$ which is a disjunction of $k - 1$ subformulae of depth $1,2,3,\ldots, k-1$.

Alternatively, using (3.2) and choosing the same expressions for $f^{x_i=1}$, $f^{x_i=0}$, after reassociating the x_i, this time using the associativity of disjunction, we get a conjunctive formula for $f(X_k)$

$\square$

We have proved that

$n-\frac{1}{2} \log n - \log\log n - O(1) \leqslant D_{B_2}(M_n) \leqslant D_{M_2}(M_n) \leqslant n$ and have matched the upper bound with a lower bound of $n-\frac{1}{2} \log n - O(1)$ under the restriction of uniformity.

Reznik (1962) gives a construction (though not a scheme) which proves

$$C_{M_2}(M_n) = O(2^n(\log n)^2/n^{3/2})$$

In fact, any scheme with q gates which covers M_n over some basis $b \subseteq B_2$ must satisfy $16^q \geqslant 2^{k.2^n/\sqrt{n}}$, $q \geqslant k.2^{n-2}/\sqrt{n}$ where k is some constant.

These results raise a myriad of open questions about the depth of "almost all" monotone functions. Two problems of particular relevance are:

(i) $D_{B_2}(M_n) \geqslant n-\frac{1}{2}\log n - O(1)$? i.e. does a lower bound of $n-\frac{1}{2}\log n - O(1)$ hold for formulae as well as schemes?

(ii) $D_{B_2}(M_n) < D_{M_2}(M_n)$ when X_n is the set of inputs?

4. THE BASIS NAND

Studies of circuit complexity draw some practical motivation from the fact that many of the tasks for which digital hardware must be designed can be represented as the computation of Boolean functions. The actual and potential efficiency (delay) of such hardware can be usefully investigated in terms of circuit depth. Likewise, hardware costs are closely related to circuit size.

In view of this practical justification, it is appropriate to consider some of the problems which face the logic designer. One of these is choice of basis. With current technologies, the choice of basis is of crucial importance in determining the overall cost and performance of a logic circuit. Some factors to be weighed when evaluating the utility of some basis are:

(1) Feasibility and economy of producing the necessary gates with physical components.

(2) Useful algebraic properties of the basic functions such as commutativity and associativity.

(3) Functional completeness i.e. the ability of the basis to implement arbitrary Boolean functions.

(4) Size of basis. Bases with a minimal number of functions have the advantages of standardization of building blocks, ease of replacement of parts etc. This uniformity may be crucial in determining manufacture and maintenance costs.

The basis consisting of binary $\wedge$, $\vee$ and unary negation is the classical set of primitives. However, this basis is rather poor with respect to our first criterion (1). This stems from the fact that $\wedge$-gates and $\vee$-gates are rather expensive to produce. Also, from the hardware point of view their performance is poor because they fail to maintain the signal value without loss of amplitude. Consequently, $\wedge$, $\vee$ -circuits sometimes require amplitude restoration after the signal has travelled through a few levels of gates.

In view of these drawbacks to the classical basis, the possibility of constructing logic circuits from gates for the functions $\overset{-}{\rightarrow}$, $\rightarrow$, $\oplus$, $\equiv$, NOR, NAND is of practical interest. The four implication functions (in logic design terminology, the functions $\overset{-}{\rightarrow}$,$\overset{-}{\leftarrow}$ are computed by INHIBIT gates) are non-commutative and thus impractical for use as standard logic gates. The two non-unate binary functions $\oplus$, $\equiv$ have many excellent characteristics as candidates for standard logic gates but are expensive to construct with physical components. They are normally available as standard logic gates in integrated circuit packages but are usually constructed internally with other standard gates. For example, with NAND gates

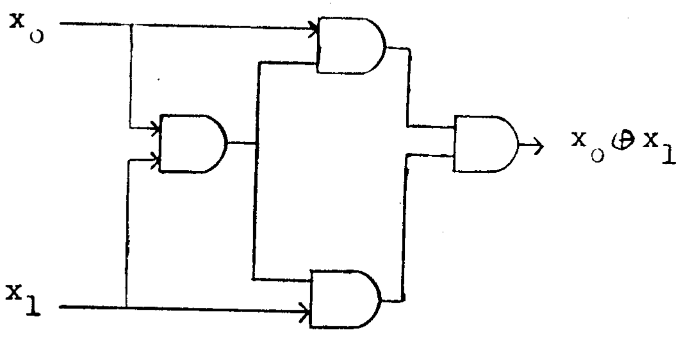


Fig. 3. A NAND circuit for $x_0 \oplus x_1$

Only a limited number of Boolean functions can be implemented using only $\oplus$, $\equiv$ gates. This lack of functional completeness is another reason for not using $\oplus$ and $\equiv$ as standard logic gates.

The functions NAND and NOR are extensively used as standard logic gates in designing digital hardware. In fact, logic circuits are more frequently constructed from NAND or NOR gates than from $\wedge$-gates, $\vee$-gates and inverters. From the hardware point of view the big advantage is that they supply outputs which maintain the signal value without loss of amplitude. This is due to the presence of transistors in circuits for NAND,NOR. Because of this, there is a gain associated with these gates which regenerates the signal upon deterioration. Diode $\wedge$-gates and $\vee$-gates do not have this property. These problems of signal deterioration must be faced when an actual digital system is designed. To the logic designer, they are usually reflected in the need to observe loading restrictions. For example, the number of output terminals on a gate (the fan-out) may have to be limited. There might also have to be a restriction on the permissible number of levels appearing in the system.

NAND and NOR gates can be easily and cheaply constructed with transistor circuits. Their associated functions satisfy the commutativity property and both form a functionally complete basis on their own. Therefore, they satisfy all of our criteria with the exception of the associativity property. For these reasons, they serve as the major components presently used in logic design.

In this chapter we consider the realization of some explicitly defined Boolean functions by circuits over the basis NAND. By duality, each of the results can be translated into a corresponding result about NOR circuits. We give results which show that for many familiar functions, an insistence on using NAND gates only for purposes of uniformity, cheapness etc., must be paid for by a substantial increase in circuit depth.

4.1 FUNCTIONAL PROPERTIES OF SHALLOW CIRCUITS

<u>Lemma 4.1</u>

For all $D > 0$,

$$\left\{ f \,\middle|\, D_{NAND}(f) \leqslant 2.D-1 \right\} \subseteq \left\{ f \,\middle|\, \begin{array}{l} f = \bigvee_{j} \bigwedge_{i \in Q_j} z_i \\ \text{where } \forall j, |Q_j| \leqslant 2^{D-1} \text{ and each} \\ z_i \text{ is some input.} \end{array} \right\}$$

<u>Proof</u>

The lemma is true for D = 1. To prove inductively that it is true for all $D > 0$, we assume it is true for $D = n$ and consider those functions f where $D_{NAND}(f) \leqslant 2n + 1$.

By the induction hypothesis, any such f may be expressed as

$$f_1 \text{ NAND } f_2 \text{ .NAND. } f_3 \text{ NAND } f_4$$

(or equivalently as $f_1 \wedge f_2 \text{ .V. } f_3 \wedge f_4$) where each f_k, $1 \leqslant k \leqslant 4$, can be expressed in the form

$$f_k = \bigvee_j \bigwedge_{i \in Q_j} z_i$$

where $|Q_j| \leqslant 2^{n-1}$ and each z_i is some input.

Thus any such f may be expressed as

$$\bigvee_j \bigwedge_{i \in Q_j} z_i$$

where $|Q_j| \leqslant 2^{n-1} + 2^{n-1}$. So the lemma is true for $D = n+1$ and for all $D > 0$

$\square$

<u>Lemma 4.2</u>

For all $D > 0$,

$$\{f \mid D_{NAND}(f) \leqslant 2.D\} \subseteq \left\{ f \;\middle|\; \begin{array}{l} f = \bigvee_{i \in Q} t_i \\ \text{where } |Q| \leqslant 2^{2^D} - 1 \text{ and} \\ \text{each } t_i \text{ is a conjunction} \\ \text{of inputs} \end{array} \right\}$$

Proof

As in Lemma 4.1, by induction on D. The result holds for D = 1. Assume it is true for D = n and consider any function f where $D_{NAND}(f) \leqslant 2\ (n+1)$. By the induction hypothesis, f may be expressed as

$$f_1 \text{ NAND } f_2 \text{ .NAND. } f_3 \text{ NAND } f_4$$

where each f_i, $1 \leqslant i \leqslant 4$, can be expressed as a disjunction of

$2^{2^n - 1}$ terms.

Therefore, f may be expressed as a disjunction of

$2.(2^{2^n - 1})^2 = 2^{2^{n+1}-1}$ terms and so the lemma is true for D = n+1

$\square$

Using rate-of-growth arguments we have derived two properties of the set of Boolean functions computable by NAND circuits of limited depth.

Definitions

Let $B_n' = B_n - \{0,1\}$ be the set of nonconstant Boolean functions of n arguments. For all f in B_n', IM(f) will denote the set of nonconstant implicants of $f(X_n)$, i.e. excluding 0. A function $t(X_n)$ is an implicant of $f(X_n)$ iff $t \rightarrow f$ and t may be expressed as a conjunction of variables in X_n or their complements.

$L(t)$ will denote the <u>length</u> of nonconstant implicant t, i.e. the (unique) number of distinct variables which appear when t is expressed as a conjunction.

For example, let $t = x_0\bar{x}_1x_2 = x_0\bar{x}_1x_2\bar{x}_3 \vee x_0\bar{x}_1x_2x_3$ be an implicant of some $f(X_n)$ in B_n. Then $t, x_0\bar{x}_1x_2\bar{x}_3$ and $x_0\bar{x}_1x_2x_3$ are all in $IM(f)$ but $L(t) = 3$ whereas $L(x_0\bar{x}_1x_2\bar{x}_3) = L(x_0\bar{x}_1x_2x_3) = 4$.

We have given two functional properties of limited depth NAND circuits in terms of the maximum length of implicants and the necessary number of implicants. These two results give general criteria which imply lower bounds on the depth of NAND circuits for certain Boolean functions. We now derive a special property for the important subset <u>symmetric</u> functions.

<u>Definition</u>

A Boolean function $f(X_n)$ in B_n is said to be <u>symmetric</u> when the value of f is unchanged by any permutation of its arguments X_n, or equivalently, f is symmetric iff there is a function g such that

$$f(X_n) = g\left(\sum_{i=0}^{n-1} x_i\right)$$

There are precisely 2^{n+1} symmetric functions in B_n since $\sum_{i=0}^{n-1} x_i$ can take $n+1$ different values. We denote the set of symmetric functions in B_n by S_n. Any function f in S_n may be defined by an n+1-tuple of Boolean values which denote the value of f when the arithmetic sum of its arguments is $0,1,2,3,\ldots,$ n-1 and n. This "defining vector" will be denoted by

$$\langle f_o, f_1, \ldots, f_n \rangle \in \{0,1\}^{n+1}$$

$$\text{where} \quad (\sum_{i=0}^{n-1} x_i = k) \Rightarrow (f(X_n) = f_k)$$

For example, $\langle 0,0,1,1,1,\ldots,1 \rangle$ denotes the function 'threshold 2', $\langle 1,0,0,1,0,0,1,0,\ldots. \rangle$ denotes 'congruent to 0(mod 3)'.

<u>Definitions</u>

Let $S_n' = S_n - \{0,1\}$ be the set of nonconstant symmetric functions of n arguments.

For $0 < k \leqslant n+1$, let

$$P_k^{(n)} = \left\{ f \in S_n \;\middle|\; \forall t,\; 0 \leqslant t \leqslant n-k+1, \left(\bigwedge_{i=t}^{t+k-1} f_i \right) = 0 \right\}$$

be the set of symmetric functions f for which $\langle f_o, f_1, \ldots, f_n \rangle$ has a 0 value in each consecutive block of length k.

For example, $P_1^{(n)}$ contains only the constant function 0, while $P_{n+1}^{(n)}$ consists of every symmetric function except the constant function 1.

Theorem 4.3

For all f in $P^{(n)}_{\lfloor n+1 - 2^{D-1} \rfloor} \cap S'_n$, $D_{NAND}(f) \geqslant 2.D$

Proof

Consider any f in S'_n such that $D_{NAND}(f) < 2D$.

By Lemma 4.1, f may be expressed as

$$f = \bigvee_j \bigwedge_{i \in Q_j} z_i$$

where $|Q_j| \leqslant 2^{D-1}$ and each z_i is some input. Let t be some implicant in IM(f) of length k, $k \leqslant 2^{D-1}$. Such an implicant may be extended to give implicants of length n, in at least $2^{n-2^{D-1}}$ different ways. For example, with n=6, D=3, $x_o\bar{x}_1x_2\bar{x}_3$ can be extended to

$$x_o\bar{x}_1x_2\bar{x}_3x_4x_5$$
$$x_o\bar{x}_1x_2\bar{x}_3x_4\bar{x}_5$$
$$x_o\bar{x}_1x_2\bar{x}_3\bar{x}_4x_5$$
$$x_o\bar{x}_1x_2\bar{x}_3\bar{x}_4\bar{x}_5$$

Thus in the defining vector of any such f there is a consecutive block of length $n+1 - 2^{D-1}$ which contains only 1 values.

So for all f in S'_n,

$$(D_{NAND}(f) < 2.D) \rightarrow (f \notin P^{(n)}_{n+1 - 2^{D-1}})$$

and the result follows

$\square$

This property of the symmetric functions computable by limited depth NAND circuits will prove useful in deriving a number of lower bounds for specific functions.

4.2 DECODERS AND ENCODERS. Discrete elements of information are normally represented in digital systems by <u>binary codes</u>. For example, the integer 45 might be represented by the 8-bit binary code 00101101. Two useful tasks for which digital hardware might be designed are decoding and encoding. A <u>decoder</u> takes an n-bit binary <u>address</u> code and on the basis of this, sets precisely one of 2^n outputs to 1 and all others to 0. Decoding circuits find applications in computer memories for the selection of a particular item of data addressed by a binary code. An <u>encoder</u> can be thought of as a positional to binary transformer. It takes a binary string of length n, exactly one element of which is 1, and produces a corresponding $\lceil \log n \rceil$ -bit binary code for the position of this element.

The operation of such hardware can be represented as the computation of Boolean functions. Let $D(n) = \left\{ d_i \middle| 1 \leqslant i \leqslant 2^n \right\}$ and $E(n) = \left\{ e_i \middle| 1 \leqslant i \leqslant \lceil \log n \rceil \right\}$ be the outputs of a decoder and encoder respectively. Then each $d_i(X_n)$ can be expressed as $\bigwedge_{i=0}^{n-1} \widetilde{x}_i$ where $\widetilde{x}_i$ denotes the variable x_i or its complement. Similarly, each $e_i(X_n)$ can be expressed as $\bigvee_{j \in Z_i} x_j$ where $Z_i \subset X_n$, $|Z_i| \leqslant \lfloor n/2 \rfloor$.

If variables and their complements are available as inputs and decoders are constructed using $\wedge$-gates, then the delay in decoding need only be $\lceil \log n \rceil$. As each output $d_i(X_n)$ depends on all its arguments, this delay is best possible. Encoders can similarly be constructed with delay $\lceil \log \lfloor n/2 \rfloor \rceil \leqslant \lfloor \log n \rfloor$ using $\vee$-gates. We will now show that if decoders or encoders are constructed with NAND gates (or dually with NOR gates) then these delays must be doubled. Thus the advantages of using NAND gates for the construction of such circuits must be paid for by a substantial loss in efficiency.

Definition

Let $E_n = \left\{ E_k^{(n)} \middle| 0 \leqslant k \leqslant n \right\}$ where $E_k^{(n)}(X_n)$ is the symmetric Boolean function which takes the value 1 iff

$\sum_{i=0}^{n-1} x_i = k$. $E_k^{(n)}$ is the <u>k^{th}</u> <u>elementary</u> <u>symmetric</u> <u>function</u>.

<u>Theorem 4.4</u>

For all f in E_n,

$$D_{NAND}(f) \geqslant 2 \lceil \log n \rceil$$

<u>Proof</u>

From the definition of $E_k^{(n)}$, it follows that

$E_n \subseteq P_2^{(n)} \cap S_n'$. The result is then immediate from Theorem 4.3 since for all $n > 1$,

$$n+1 - 2^{\lceil \log n \rceil - 1} \geqslant 2$$

$\square$

The definition of $E_n^{(n)} = \bigwedge_{i=0}^{n-1} x_i$ suggests a NAND circuit of depth $2 \lceil \log n \rceil$. The previous result shows that this depth bound is optimal for the basis NAND.

<u>Lemma 4.5</u>

$$D_{NAND}(\bigvee_{i=0}^{n-1} x_i) \geqslant 2 \lceil \log n \rceil - 1$$

<u>Proof</u>

By complementing the output and variable inputs of a

NAND circuit which computes $\bigvee_{i=0}^{n-1} x_i$, we obtain a circuit for $E_n^{(n)}$. Inputs may be complemented without increasing depth and any output can be complemented by a single NAND gate yielding

$$D_{NAND}(f) + 1 \geqslant D_{NAND}(\overline{f})$$

for all f in B_n. Thus

$$D_{NAND}(\bigvee_{i=0}^{n-1} x_i) + 1 \geqslant D_{NAND}(E_n^{(n)})$$

and the result follows from Theorem 4.4

$\square$

These lower bounds hold even when arguments can appear as inputs in either complemented or uncomplemented form. The lower bound for the function $E_n^{(n)}$ is tight as it has a NAND circuit of depth $2\lceil \log n \rceil$ with inputs from the set X_n. However, although $\bigvee_{i=0}^{n-1} x_i$ can be computed in depth $2\lceil \log n \rceil -1$ when $I_n = \langle x_0, x_1, \ldots, x_{n-1}, \overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{n-1}, 0, 1\rangle$ is the set o: inputs, the best known upper bound when complemented arguments are forbidden is $2\lceil \log n \rceil$. We now show that this upper bound is best possible in this case.

<u>Theorem 4.6</u>

$$D_{NAND}(\bigvee_{i=0}^{n-1} x_i) \geqslant 2\lceil \log n \rceil$$

when $\langle x_0, x_1, \ldots, x_{n-1}, 0, 1\rangle$ is the set of possible inputs.

The following property of NAND formulae will be used in the proof of Theorem 4.6. It is given here without proof.

Fact 4.7

Let β be any NAND formula and β' be the formula β with all inputs on paths of length 2D+1, for some integer $D \geqslant 0$, replaced by the input 0. If β computes f and β' computes f', then $f \rightarrow f'$

$\square$

Proof of Theorem 4.6

Let $h = \bigvee_{i=0}^{n-1} x_i$. From this definition it follows that

$$\left\{ f \,\middle|\, (h \not\equiv f) \wedge (h \rightarrow f) \right\} = \left\{ 1 \right\} \qquad (4.1)$$

Assume $D_{NAND}(h) = 2D + 1$ for some integer $D \geqslant 0$ and consider an optimal depth formula β which computes h.

In formula β , replace all inputs from the subset $\langle x_0, x_1, \ldots, x_{n-1} \rangle$ which are on paths of length 2.D+1 by the input 0 and remove redundant gates. In view of Fact 4.7, we obtain a formula β' of depth 2D which computes h', where $h \rightarrow h'$.

By definition of the replacements, if $h(X_n), h'(X_n)$ are not equal then there is some x_i in X_n which takes the value 1. But for all $x_i \in X_n$, $h|_{x_i=1} \equiv h'|_{x_i=1}$ as $h' \in \left\{ h, 1 \right\}$ which in turn follows from (4.1) and the fact that $h \rightarrow h'$.

This contradiction proves that h, h' are equivalent and we have obtained a circuit of reduced depth which still computes h. This contradicts the assumption of optimal depth and so we must have $D_{NAND}(h) = 2D$ for some integer $D \geqslant 0$. The result now follows from the fact that $D_{NAND}(h) \geqslant 2\lceil \log n \rceil - 1$ when I_n is the set of inputs.

$\square$

The proof of Theorem 4.6 is particularly interesting as it gives the first application of the "specific refinement" technique to non-monotone circuits, albeit only to circuits over a very restricted complete basis. This technique has already been applied to monotone circuits by Paterson (1975).

Soprunenko (1965) has considered the circuit <u>size</u> of Boolean functions over the basis NAND and has established that

$$C_{NAND}(E_n^{(n)}) \geqslant 2.n - O(1)$$

$$C_{NAND}\left(\bigvee_{i=0}^{n-1} x_i\right) \geqslant 3.n - O(1)$$

Therefore, the circuit size and depth of $E_n^{(n)}$ are both increased by a factor of two in going from the basis U_2 to the basis NAND, while the size and depth of $\bigvee_{i=0}^{n-1} x_i$ are increased by factors of 3,2 respectively.

A result of section 2.3 shows that decoders and encoders require depth 1.44 log n (to within an additive constant) over the complete basis $\{\Rightarrow, 1\}$. Thus "INHIBIT" gates are more efficient w.r.t. depth than NAND gates and less efficient than $\wedge$-gates and $\vee$-gates for the purpose of building decoders and encoders.

4.3 COUNTING (MODULO K). We have considered two useful tasks for which digital hardware might be designed, decoding and encoding, and have investigated the inherent delay associated with these problems. Another important task is that of counting. Binary counters of one form or another are basic subsystems in a modern computer. If we are interested in combinational (rather than sequential) circuits, then the operation of a standard binary counter can be thought of as unary-to-binary conversion. Pippenger (1974) describes a construction which, taken in conjunction with Theorem 2.2, shows that a standard binary counter with delay of about 10·2.log n can be constructed from NAND gates. This upper bound is very rough and can probably be substantially improved by considering the details of the actual construction. We shall not pursue this further here. Instead, another type of counter is considered.

Often a circuit is required which will count with respect to some specified modulus, i.e. when two numbers which differ by a multiple of some number are to be considered equivalent.

In the case where this number is 2, two such numbers are said to have the same parity. Circuits for parity checking find important applications in the transmission of information where they are used to check for any inaccuracy in the reproduction. In this section we establish a number of bounds on the delay required by modulo k counters, when constructed from NAND gates.

Definition

Let $C^{(n)}_{k,r}$ (X_n) be the symmetric Boolean function which is 1 iff

$$\sum_{i=0}^{n-1} x_i \equiv r \pmod{k}.$$

Lemma 4.8

For all integers k and r, $1 < k \leqslant 2^D$, $0 \leqslant r \leqslant k-1$,

$$D_{NAND}(C^{(2^D)}_{k,r}) \geqslant 2.D$$

Proof

By definition, each such $C^{(2^D)}_{k,r}$ is in $P_2^{(2^D)} \cap S_{2^D}$, thus the result follows from Theorem 4.3 since for all $D \geqslant 1$,

$$2^D + 1 - 2^{D-1} \geqslant 2$$

$\square$

This lower bound may be compared with two known results which apply to a smaller class of functions, but which hold over more expressive bases. Fischer, Meyer and Paterson (1975) have shown that for all <u>fixed k</u>, <u>$k > 2$</u>, $D_{B_2}(C^{(n)}_{k,r})$ is not less than log n + loglog n - o(loglog n)*, while Khrapchenko (1972a) has established that for all <u>fixed k</u>, <u>$k > 1$</u>,

$$D_{U_2}(C^{(n)}_{k,r}) \geqslant 2 \log n - O(1).$$

Recently, Stockmeyer (1976) has established a general lower bound on the <u>circuit size</u> of "congruence" functions. The main result allows one to infer that

$$C_{B_2}(C^{(n)}_{k,0}) \geqslant \tfrac{5}{2}n - k/2 - 4$$

for all integers k and n with $3 \leqslant k < n$. It is also shown that

$C_{B_2}(C^{(n)}_{4,0}) \leqslant \frac{5}{2}n$ for all n and so the lower bound is optimal to within an additive constant in this case. Lower bounds of the form $\frac{5}{2}n - c$, where c is a constant independent of n, are the best presently known on the circuit size of any explicitly defined function over the basis B_2.

We now derive certain upper bounds on

$D_{NAND}(C^{(2^D)}_{k,r})$ for k = 2,3 and 4. The result for k = 2 establishes that the growth rates in Lemmas 4.1, 4.2 are achievable. We also show that the lower bound of Lemma 4.8 is <u>not</u> achievable for k = 3.

* Paterson (private communication) has improved this lower bound to log n + loglog n - O(1).

Lemma 4.9

For all $D \geqslant 0$, $\quad D_{NAND}(C_{2,0}^{(2^D)}) \quad = \quad D_{NAND}(C_{2,1}^{(2^D)}) \quad = \quad 2.D$

Proof

The lower bounds are derived in Lemma 4.8 and the upper bounds are now proved by induction on D.

For $D = 0$,

$$C_{2,0}^{(1)}\ (X_1) = \bar{x}_o$$

$$C_{2,1}^{(1)}\ (X_1) = x_o$$

and so the lemma is true in this case. To prove inductively that it is true for all $D \geqslant 0$, we assume it is true for $D = n$.

Let $X^1_{2^n} = \langle x_o, x_1, \ldots, x_{2^n-1} \rangle$ and $X^2_{2^n} = \langle x_{2^n}, x_{2^n+1}, \ldots, x_{2^{n+1}-1} \rangle$.

$C_{2,0}^{(2^{n+1})}\ (X_{2^{n+1}})$ may be expressed as

$$C_{2,0}^{(2^n)}\ (X^1_{2^n}) \text{ NAND } C_{2,0}^{(2^n)}\ (X^2_{2^n}) \text{ .NAND. } C_{2,1}^{(2^n)}\ (X^1_{2^n}) \text{ NAND } C_{2,1}^{(2^n)}(X^2_{2^n})$$

while $C_{2,1}^{(2^{n+1})}\ (X_{2^{n+1}})$ may be expressed as

$$C_{2,0}^{(2^n)}(X^1_{2^n}) \text{ NAND } C_{2,1}^{(2^n)}(X^2_{2^n})\text{.NAND. } C_{2,1}^{(2^n)}(X^1_{2^n}) \text{ NAND } C_{2,0}^{(2^n)}(X^2_{2^n})$$

and so the lemma is true for $D = n+1$ and thus for all $D \geqslant 0$

$\square$

The following two facts about $C_{k,r}^{(2^D)}$, for all $1 < k \leqslant 2^D$, are given without proof.

<u>Fact 4.10</u>

For all implicants t in $IM(C_{k,r}^{(2^D)})$, $L(t) = 2^D$

$\square$

This implies that if $C_{k,r}^{(2^D)}$ is expressed as a disjunction of implicants, then each implicant in $IM(C_{k,r}^{(2^D)})$ must appear at least once. Noting chis fact we can also obtain the following explicit formula for the number of implicants in $IM(C_{k,r}^{(2^D)})$.

<u>Fact 4.11</u>

$$\left| IM(C_{k,r}^{(2^D)}) \right| = \binom{2^D}{r} + \binom{2^D}{r+k} + \binom{2^D}{r+2k} + \binom{2^D}{r+3k} + \ldots\ldots.$$

$\square$

<u>Corollary 4.12</u>

$$\left| IM(C_{2,0}^{(2^D)}) \right| = \left| IM(C_{2,1}^{(2^D)}) \right| = 2^{2^D - 1}$$

<u>Proof</u>

Immediate from 4.11

$\square$

Combining Lemma 4.9 and Corollary 4.12, we have the following

Proposition 4.13

The upper bound on growth rate in Lemma 4.2 is achievable.

$\square$

While combining Lemma 4.9 and Fact 4.10 gives

Proposition 4.14

The upper bound on growth rate in Lemma 4.1 is achievable.

$\square$

Furthermore, both these upper bounds are simultaneously achieved by an optimal depth circuit for $C_{2,0}^{(2^D)}$.

In this chapter we have derived various lower bounds on the depth of Boolean functions over the basis NAND. These have been obtained from rate-of-growth arguments which characterized the functions computable by limited depth NAND circuits. We have also shown that the growth rates which were used to obtain our lower bounds can be simultaneously achieved. Therefore, in order to prove lower bounds larger than 2D for any f in B_{2^D}, we must consider properties other than just the minimum length of implicants or the necessary number of implicants.

In order to prove that the lower bound of Lemma 4.8 is not achievable for k = 3 we require the following result which is easily obtained from Fact 4.11.

Fact 4.15

For all $D \geqslant 0$,

$$\left\lfloor \frac{2^{2^D}}{3} \right\rfloor \leqslant \left| IM(C_{3,r}^{(2^D)}) \right| \leqslant \left\lceil \frac{2^{2^D}}{3} \right\rceil$$

and whether it equals the upper bound or the lower bound depends on r.

$\square$

Theorem 4.16

For $r = 0,1$ and 2,

$$D_{NAND}\ (C_{3,r}^{(2^D)}) \geqslant 2.D + 1$$

Proof

If $D_{NAND}\ (C_{3,r}^{(2^D)}) \leqslant 2.D$ then there are functions f_1, f_2, f_3 and f_4 such that $C_{3,r}^{(2^D)} = f_1 \wedge f_2 . \vee . f_3 \wedge f_4$ and $D_{NAND}(f_i) \leqslant 2.D - 2$ for all $1 \leqslant i \leqslant 4$.

Noting Lemmas 4.1, 4.2, each f_i $(1 \leqslant i \leqslant 4)$ can be represented as :

$$\bigvee_{j \in N_i} t_{ij} \quad \text{where} \quad t_{ij} = \bigwedge_{k \in Q_{ij}} Z_k$$

and each Z_k is some input.

Also,

$$\text{for all } i,j, \quad |Q_{ij}| \leqslant 2^{D-1} \qquad (4.2)$$

$$\text{for all } i, \quad |N_i| \leqslant 2^{2^{D-1}-1} \qquad (4.3).$$

This representation of f_i will be denoted by the set of terms

$$F_i = \{t_{i1}, t_{i2}, t_{i3}, \ldots, t_{in_i}\} \quad \text{where } n_i = |N_i|$$

and for all i,j, t_{ij} is in $IM(f_i)$ (i.e. $t_{ij} \neq 0$).

Also, each term is assumed to be distinct, i.e.

$$j \neq k \rightarrow t_{ij} \neq t_{ik} .$$

In view of Fact 4.10, $f_1 \wedge f_2$ can be represented by the set

$$F_{12} = \left\{ t_{1j} \wedge t_{2k} \;\middle|\; \begin{array}{l} t_{1j} \in F_1 \\ t_{2k} \in F_2 \end{array} \right\} \cap IM(C_{3,r}^{(2^D)}(X_{2^D})).$$

Each term in F_{12} is nonconstant and we assume that each term is distinct. F_{34} can be similarly defined.

Taken together, Facts 4.10 and 4.15 show that at least one of F_{12}, F_{34} contains not less than $\lfloor 2^{2^D}/3 \rfloor /2$ implicants. Without loss of generality, we assume that $|F_{12}| \geq \lfloor 2^{2^D}/6 \rfloor$. With this assumption, we now establish the theorem by a rather delicate analysis of combined growth rates. The following property of implicants is essential to the proof.

If $t_a \in F_1$, $t_b, t_c \in F_2$ and $t_a \wedge t_b, t_a \wedge t_c \in F_{12}$, then corresponding to t_a there is a unique residue r_a, $0 \leq r_a < 3$, Then in view of (4.3),

$$\left| \text{Extra } (t_{11}) \right| \leqslant 2^{2^{D-1}}/2 - M$$
$$\leqslant \left\lfloor 2^{2^{D-1}}/6 \right\rfloor .$$

If we now let

$$T_{11} = \left\{ t_{1i} \;\middle|\; t_{11} \wedge t_{2j},\ t_{1i} \wedge t_{2j} \in F_{12} \right\}$$

then from the upper bound of M on Fanout (t) for any term t, we have

and a domain $D_a \subset X_{2^D}$, $\left| D_a \right| = 2^{D-1}$ such that

$$t_b, t_c \in IM(C^{(2^{D-1})}_{3,r_b} (X_{2^D} - D_a)) \qquad (4.4)$$

where $r_a + r_b \equiv r \pmod 3$. The validity of this observation can be checked by noting (4.2) and Fact 4.10. By symmetry, the property holds when $t_a \in F_2$ and $t_b, t_c \in F_1$.

We now define

$$\left| T_{11} \right| \geqslant \frac{\left| F_{12} \right|}{M} - \left| \mathrm{Extra}(t_{11}) \right| \geq M - O(1).$$

Property (4.4) shows that if t_{1i}, t_{1j} are both in T_{11}, then they are both in $IM(C^{(2^{D-1})}_{3,r_{11}}(D_{11}))$ and so the maximum size of $\left| F_{12} \right|$ is not more than

$$\left| IM(C^{(2^{D-1})}_{3,r}) \right|^2 + (2^{2^{D-1}} - 1 - \left| T_{11} \right|). \left| \mathrm{Extra}\ (t_{11}) \right|$$

$$\leqslant \frac{2^{2^D}}{9} + (\frac{2^{2^{D-1}}}{6}).(\frac{2^{2^{D-1}}}{6}) + o(2^{2^D})$$

which is less than $\lfloor 2^{2^D}/6 \rfloor$ and we have a contradiction.

$\square$

The proof of Theorem 4.16 combines results on the maximum number of implicants and on maximum implicant size in order to prove a lower bound on $D_{NAND}(C^{(2^D)}_{3,r})$. The difference between this lower bound and the result of Lemma 4.8 is only 1 for all values of D. However, the improved result is worthy of consideration as it provides the largest lower bound on circuit depth which has yet been proved for any explicitly defined function. Also, the rather complex proof of this result indicates limitations in our rate-of-growth approach when aiming for lower bounds larger than 2D.

The problem of whether the lower bound of Lemma 4.8 is achievable for some fixed $k > 3$ remains unresolved. The following bounds on the number of distinct implicants of $C_{4,r}$ are easily obtained from Fact 4.11.
For all $D > 1$,

$$2^{2^D-2} - 2^{2^{D-1}-1} \leqslant \left| IM\left(C_{4,r}^{(2^D)}\right) \right| \leqslant 2^{2^D-2} + 2^{2^{D-1}-1}$$

and the precise value depends on r.

However, the proof technique of Theorem 4.16 seems to apply only to $C_{3,r}^{(2^D)}$, and the best lower bound we have on $D_{NAND}(C_{k,r}^{(2^D)})$, for $k \neq 3$, is Lemma 4.8.

We conclude this section by establishing upper bounds on the depth of $C_{k,r}^{(2^D)}$ for $k = 3,4$.

<u>Theorem 4.17</u>

For $r = 0,1$ and 2,

$D_{B_2}(C_{3,r}^{(2^D)}) \leqslant 2.D$

<u>Proof</u>

The theorem is true for $D = 0$. To show inductively that it is true for all $D \geqslant 0$, we assume it is true for $D = n$.

Let $X^1_{2^n} = \langle x_0, x_1, \ldots, x_{2^n-1} \rangle$ and $X^2_{2^n} = \langle x_{2^n}, \ldots, x_{2^{n+1}-1} \rangle$

$C^{(2^{n+1})}_{3,r}$ $(X_{2^{n+1}})$ may be expressed as

$$C^{(2^n)}_{3,\ 2r \ (\mathrm{mod}\ 3)}(X^1_{2^n}) \equiv C^{(2^n)}_{3,2r \ (\mathrm{mod}\ 3)}(X^2_{2^n})$$

$$.\wedge.\ C^{(2^n)}_{3,2r+1 \ (\mathrm{mod}\ 3)}(X^1_{2^n}) \equiv C^{(2^n)}_{3,2r+2 \ (\mathrm{mod}\ 3)}(X^2_{2^n})$$

and so the theorem is true for $D = n+1$ and thus for all $D \geqslant 0$

$\square$

<u>Corollary 4.18</u>

For $r = 0,1$ and 2,

$$D_{NAND}(C^{(2^D)}_{3,r}) \leqslant 4.D$$

<u>Proof</u>

By Lemma 2.4

$\square$

We now show that the upper bound of 4.18 can be improved.

<u>Theorem 4.19</u>

For all $D \geqslant 0$ and for $r = 0,1$ and 2,

$$D_{NAND}(C^{(4^D)}_{3,r}) \leqslant 6.D$$

<u>and</u>

$$D_{NAND}(\overline{C}^{(4^D)}_{3,r}) \leqslant 6.D$$

<u>Proof</u>

As in Theorem 4.17, the proof is by induction on D. However, the inductive step is slightly more complicated. For simplicity, we require the following :

<u>Notation</u>

$\langle C, R, \underline{X} \rangle$ will denote the function

$$C^{(|\underline{X}|)}_{3,R \ (\mathrm{mod}\ 3)}(\underline{X})$$

and $\langle \overline{C}, R, \underline{X} \rangle$ will denote its complement.

<u>E 1</u>. $C_{3,r}(\underline{X},\underline{Y})$ can be expressed as

$(f_1 \vee f_2)$ NAND $(f_3 \vee f_4)$.NAND. f_5 NAND f_6

where $f_1 = \langle C,2r,\underline{X} \rangle$

$f_2 = \langle C,2r+1,\underline{Y} \rangle$

$f_3 = \langle C,2r+2,\underline{X} \rangle$

$f_4 = \langle C,2r,\underline{Y} \rangle$

$f_5 = \langle C,2r+1,\underline{X} \rangle$

$f_6 = \langle C,2r+2,\underline{Y} \rangle$

<u>E 2</u>. $\overline{C}_{3,r}(\underline{X},\underline{Y})$ can be expressed as

$(\overline{f}_1$ NAND $\overline{f}_2$.NAND. $\overline{f}_3$ NAND $\overline{f}_4) \wedge (\overline{f}_5 \vee \overline{f}_6)$

E 3. $\overline{C}_{3,r}(\underline{X},\underline{Y})$ can be expressed as

$$(g_1 \vee g_2)\text{NAND}(\overline{g}_1 \vee \overline{g}_2).\text{NAND}.(g_3 \vee g_4)\text{NAND}(\overline{g}_3 \vee \overline{g}_4)$$

where
$$\begin{aligned} g_1 &= \langle C, 2r, \underline{X} \rangle \\ g_2 &= \langle C, 2r, \underline{Y} \rangle \\ g_3 &= \langle C, 2r+1, \underline{X} \rangle \\ g_4 &= \langle C, 2r+2, \underline{Y} \rangle \end{aligned}$$

E 4. $C_{3,r}(\underline{X},\underline{Y})$ can be expressed as

$$(\overline{g}_1 \text{ NAND } \overline{g}_2 \text{ .NAND.} g_1 \text{ NAND } g_2) \wedge (\overline{g}_3 \text{ NAND } \overline{g}_4 \text{ .NAND. } g_3 \text{ NAND } g_4)$$

<u>Inductive step</u>

Let $X^1_{4^D}$, $X^2_{4^D}$, $X^3_{4^D}$, $X^4_{4^D}$ be a partition of $X_{4^{D+1}}$.

We give the inductive step as an algorithm for constructing NAND circuits for $C_{3,r}(X_{4^{D+1}})$ and its complement.

<u>$C_{3,r}$</u>

1) Express $C_{3,r}$ using E 1.

2) Express the resulting $f_1(X^1,X^2)$, $f_2(X^3,X^4)$, $f_3(X^1,X^2)$ and $f_4(X^3,X^4)$ using E 4.

3) Express $f_5(X^1,X^2)$ and $f_6(X^3,X^4)$ using E 1.

$\overline{C}_{3,r}$

1) Express $\overline{C}_{3,r}$ using E 3.

2) Express the resulting $g_1(X^1,X^2)$, $g_2(X^3,X^4)$, $g_3(X^1,X^2)$ and $g_4(X^3,X^4)$ using E 4.

3) Express $\overline{g}_1$, $\overline{g}_2$, $\overline{g}_3$ and $\overline{g}_4$ using E 2.

In each case we then transform subcircuits according to the identity

$(x_0 \text{ NAND } x_1 \text{ .NAND. } x_2 \text{ NAND } x_3) \equiv (x_0 \wedge x_1 \text{ .}\vee\text{. } x_2 \wedge x_3)$

and simulate any remaining $\vee$-gates using

$(x_0 \text{ NAND } x_0 \text{ .NAND. } x_1 \text{ NAND } x_1) \equiv (x_0 \vee x_1)$

This completes the proof of the inductive step and the result follows by noting that the theorem is true for D=0. $\square$

Corollary 4.20

$$2D + 1 \leqslant D_{\text{NAND}}(C_{3,r}^{(2^D)}) \leqslant 3D + O(1)$$

Proof

By Theorems 4.16, 4.19 as $4^{\lceil D/2 \rceil} \geq 2^D$ $\square$

Vilfan (1972) describes a short formula for $C_{4,0}^{(n)}$ which was suggested by A. Meyer. We show how this can be used to obtain an upper bound on $D_{NAND}(C^{(n)}_{4,0})$ which is within $o(\log n)$ of the lower bound in Lemma 4.8.

For $n = 2^k$,

$$C^{(n)}_{4,0}(X_n) = \overline{H}_k(X_n) \wedge \overline{L}_k(X_n)$$

where $H_k(X_n)$ and $L_k(X_n)$ are the functions which compute the high and low order digits of the binary representation of

$$\sum_{i=0}^{n-1} x_i \pmod 4$$

Thus, $H_0(x_0) \equiv 0$, $L_0(x_0) \equiv x_0$

By binary addition (mod 4),

$$L_k(X_n) = \bigoplus_{i=0}^{n-1} x_i$$

and $H_k(X_n) = H_{k-1}(X^1_{2^{k-1}}) \oplus H_{k-1}(X^2_{2^{k-1}}) \oplus (L_{k-1}(X^1_{2^{k-1}}) \wedge L_{k-1}(X^2_{2^{k-1}}))$

where $X^1_{2^{k-1}} = \langle x_0, x_1, \ldots, x_{2^{k-1}-1} \rangle$

and $X^2_{2^{k-1}} = \langle x_{2^{k-1}}, x_{2^{k-1}+1}, \ldots, x_{2^k-1} \rangle$

By recursively applying this identity we obtain a formula for $H_k(X_n)$ which, after reassociation, may be expressed as

$$\bigoplus_{i=1}^{k} F_i$$

where F_i is a formula of size n which contains every $x_i \in X_n$.

Hence, $D_{B_2}(L_k(X_n)) = \log n$

$$D_{B_2}(H_k(X_n)) \leqslant \log n + \lceil \log k \rceil$$

and thus $D_{B_2}(C_{4,0}^{(n)}) \leqslant \log n + \log\log n + O(1)$

If n is not a power of two, we can obtain a formula for $C_{4,0}^{(n)}$ from one for $C_{4,0}^{(m)}$, where $n < m < 2.n$ and m is a power of two. Therefore, the above upper bound holds for all $n \geqslant 1$.

Combining this construction with the result of Lemma 2.4, yields

$$D_{NAND}(C_{4,0}^{(n)}) \leqslant 2 \log n + o(\log n)$$

Formulae for $C_{4,1}^{(n)}$, $C_{4,2}^{(n)}$ and $C_{4,3}^{(n)}$ can all be obtained from formulae for $C_{4,0}^{(n+3)}$ and so these upper bounds on depth over B_2 and NAND hold for all r.

5. SYMMETRIC FUNCTIONS

Symmetric functions arise in many familiar computational problems such as sorting and counting. Therefore, the inherent complexity of symmetric functions is closely related to the potential efficiency of algorithms for many practical problems. The properties of symmetric functions make them interesting from a theoretical point of view and it is perhaps this, more than any practical significance, which has stimulated much of the research into their computational complexity.

There is a long history of improvements to the best known upper bound on the formula size of all symmetric Boolean functions. Korobkov (1956) seems to have been first to investigate this problem. The first polynomial upper bound was derived by Khrapchenko (1972b) who showed that

$$F_{U_2}(S_n) \leqslant n^{4.93}$$

Some recent advances have been made for the full basis B_2. The best upper bound published to date is due to Pippenger (1974).

$$F_{B_2}(S_n) = O(n^{3.55})$$

A useful theorem for deriving lower bounds on the formula size of Boolean functions over the basis B_2 is due to Neciporuk (1966). For any $f(X_n) \in B_n$, suppose the arguments X_n are partitioned into <u>blocks</u> $R_1, \ldots, R_p$. If for some i the arguments in all the blocks R_j, $j \neq i$, are fixed to 0 or 1 in some way, the result is a <u>restriction</u> of f, a function $f'(R_i)$. Let m_i be the number of different such restrictions f' for all possible fixations of the other variables. Then we have the following

<u>Theorem</u> (Neciporuk (1966))

There exists $a > 0$ such that for all f,

$$F_{B_2}(f) \geqslant a. \sum_{i=1}^{p} \log m_i$$

where the m_i's are as defined above

$\square$

Using this theorem, Neciporuk (1966) has derived lower bounds of $n^2/\log n$ for some rather artificial functions, each of which involve some notion of "indirect addressing". Harper and Savage (1972) have also applied this theorem to a practical combinatorial problem and obtained a lower bound of $a.n^{3/2}$.

For symmetric functions, the maximum number of distinct restrictions of a block of size r is limited to $\min\left\{2^{r+1}, n-r+1\right\}$ and thus only linear lower bounds can be obtained by this technique. The best lower bound which has been proved for any such function over the basis B_2 is due to

Fischer, Meyer and Paterson (1975) who have shown that many symmetric functions require formulae of size at least $n \log n / \log\log n$. (M.S.Paterson has improved this lower bound to $n \log n$.)

5.1 A COROLLARY OF A THEOREM OF SPECKER.

In this section we consider the complexity of symmetric functions over the full basis B_2.

For circuit size, the results of Schnorr (1974), (1976) imply that for each $n > 2$ all but eight of the 2^{n+1} functions in S_n have circuit size which is at least $2n - 3$. The eight exceptions consist of two constant functions and six with circuit size $n-1$. Stockmeyer (1976) shows that at least one half of the functions in S_n have circuit size which is at least $(5/2)n - 5$. He also states that

$$C_{B_2}(S_n) \leqslant 6.n$$

For formula size we have the important result of Hodes and Specker (1968) which gives non-linear lower bounds for a number of interesting Boolean functions. Paterson (1976) points out that when the theorem of Specker is restricted to symmetric functions it can be restated as

<u>Theorem 5.1</u> (Hodes and Specker (1968))

For all f in $S_n - \triangle_n$, where $\triangle_n$ is defined overleaf,

$$F_{B_2}(f) \geqslant n.t(n)$$

for some (slowly growing) function $t(n)$ with $t \to \infty$ as $n \to \infty$, where $t(n) < \log^* n$

$\square$

<u>Definitions</u>

Let $EQ(X_n) = \bigwedge_{i=0}^{n-2} \left[x_i \equiv x_{i+1} \right]$.

Using our notation for the "defining vector" of a symmetric function, we define

$$\Delta_n = \left\{ f \in S_n \;\middle|\; EQ(f_1, f_3, f_5, \ldots, f_{n-1}) \wedge EQ(f_2, f_4, f_6, \ldots, f_{n-2}) \right\}$$

for n even, and

$$\Delta_n = \left\{ f \in S_n \;\middle|\; EQ(f_1, f_3, f_5, \ldots, f_{n-2}) \wedge EQ(f_2, f_4, f_6, \ldots, f_{n-1}) \right\}$$

for n odd.

<u>Note</u>

There are precisely 16 functions in Δ_n and the defining vector of each such function has one of the forms:

$$\langle ?,0,0,\ldots\ldots\ldots,0,? \rangle$$
$$\langle ?,1,1,\ldots\ldots\ldots,1,? \rangle$$
$$\langle ?,0,1,0,1,\ldots\ldots\ldots,? \rangle$$
$$\langle ?,1,0,1,0,\ldots\ldots\ldots,? \rangle$$

<u>Corollary 5.2</u>

For all f in S_n, either

$$D_{B_2}(f) \leqslant \lceil \log n \rceil + 2$$

or

$$D_{B_2}(f) \geqslant \log n + \log t(n) - O(1)$$

for some t(n) with $t \to \infty$ as $n \to \infty$

Proof

Lower bound is immediate from Theorem 5.1 while the upper bound can be verified by considering formulae for functions in Δ_n

$\square$

It seems likely that the lower bound in Corollary 5.2 can be improved to log n + loglog n - O(1). However, at present this is an open problem.

5.2 LOWER BOUNDS OVER UNATE BASES. We now prove similar "gap" theorems for the depth of symmetric functions over various unate bases and for the depth of monotone symmetric (threshold) functions over the basis M_2.

Some simple facts about the defining vector of symmetric functions are given without proof.

Fact 5.3

$\overline{f}$, the complement of some f in S_n, is defined by $\langle \overline{f}_0, \overline{f}_1, \overline{f}_2, \ldots, \overline{f}_n \rangle$

$\square$

Fact 5.4

Let $\hat{f}$ be the function computed by a circuit for some f in S_n with all nonconstant inputs complemented. Then $\hat{f}$ is defined by $\langle f_n, f_{n-1}, \ldots, f_2, f_1, f_0 \rangle$

$\square$

Definitions

$$K_n = \left\{ f \in S_n \;\middle|\; \overline{EQ}(f_{\lceil n/4 \rceil}, f_{\lceil n/4 \rceil + 1}, \ldots, f_{\lfloor 3n/4 \rfloor}) \right\}$$

$$\Phi_n = \left\{ f \in S_n \;\middle|\; EQ(f_1, f_2, \ldots, f_{n-1}) \right\}$$

K_n is the subset of n argument symmetric functions whose defining vector is either

$$\langle ?,\ldots\ldots \,\Big|\, \ldots,?,0,1,?,\ldots \,\Big|\, \ldots\ldots,? \rangle$$

or $\qquad \lceil n/4 \rceil \qquad\qquad \lfloor 3n/4 \rfloor$

$$\langle ?,\ldots\ldots \,\Big|\, \ldots\ldots,?,1,0,?, \,\Big|\, \ldots\ldots,? \rangle$$

while the defining vector of any function in Φ_n has one of the forms:

$$\langle ?,0,0,\ldots\ldots,0,? \rangle$$
$$\langle ?,1,1,\ldots\ldots,1,? \rangle$$

Thus, there are precisely eight functions in Φ_n.

Examples. $SUM^{(n)}$ is in K_n.

$E_n^{(n)}$, the nth elementary symmetric function is in Φ_n.

Theorem 5.5

For all $f \in S_n$, either

$$D_{U_2}(f) \leqslant \lceil \log n \rceil + 1$$

or

$$D_{U_2}(f) \geqslant \log n + \log\log n - O(1)$$

Proof

A result of Khrapchenko (1972a) can be paraphrased as the assertion that for all f in K_n,

$$D_{U_2}(f) \geqslant 2.\log n - O(1)$$

The upper bound can be verified by considering formulae for functions f in Φ_n. Thus, the result follows from the following lemma

$\square$

Lemma 5.6

For all f in $S_n - \left\{K_n \cup \Phi_n\right\}$,

$$D_{U_2}(f) \geqslant \log n + \log\log n - O(1)$$

Proof

In the defining vector of each function $f(X_n)$ in $S_n - \left\{K_n \cup \Phi_n\right\}$ there is a consecutive block of length p, $\lfloor n/2 \rfloor + 2 \leqslant p \leqslant n+1$, which has one of the following forms:

0,1,0,0,.....,0	1,0,1,1,.....,1
1,1,0,0,.....,0	0,0,1,1,.....,1
0,0,...,0,0,1,0	1,1,...,1,1,0,1
0,0,...,0,0,1,1	1,1,...,1,1,0,0

By setting arguments to constants and simplifying a circuit which computes f, we can obtain a circuit which is not deeper and which computes some f' in S_{p-1}, where f' is defined by a vector in one of the above forms.

Over the basis U_2, the output and variable inputs of a circuit may be complemented without increasing depth. In view of this and Facts 5.3, 5.4, we need only consider those functions which are defined by the vectors

$$\langle 0,0,1,1,\ldots\ldots,1\rangle$$
$$\text{and } \langle 1,0,1,1,\ldots\ldots,1\rangle .$$

Krichevskii (1964) has shown that on p-1 arguments, each of these symmetric functions requires formula size of order at least p log p over the basis U_2.

For $\lfloor n/2 \rfloor + 2 \leqslant p \leqslant n + 1$ and any such f' in S_{p-1}, this yields

$$D_{U_2}(f') \geqslant \log n + \log\log n - O(1)$$

$\square$

<u>Lemma 5.7</u>

For all $\lceil n/2 \rceil < k \leqslant n + 1$,

$$P_k^{(n)} \cup \left\{ \bar{f} \,\middle|\, f \in P_k^{(n)} \right\} \supseteq S_n$$

<u>Proof</u>

Immediate from the definition of $P_k^{(n)}$

$\square$

<u>Theorem 5.8</u>

For all f in S_n',

$$D_{NAND}(f) \geqslant 2 \lceil \log n \rceil - 3$$

Proof

S_n' is the set of nonconstant functions in S_n.

Let f be some function in S_n'. Then from Theorem 4.3, we have

$$(f \in P_{\lceil n/2 \rceil + 1} \cap S_n') \rightarrow (D_{NAND}(f) \geqslant 2\lceil \log n \rceil - 2)$$

The result then follows from Lemma 5.7 if we note that for all f in B_n,

$$D_{NAND}(f) + 1 \geqslant D_{NAND}(\bar{f})$$

$\square$

Using identical proof techniques to those used for Theorem 5.8 we can prove that for all f in S_n',

$$D_{\{NAND, \rightarrow\}}(f) \geq \log_{\phi} 2.\log n - O(1)$$

where ϕ is the golden ratio, by using Theorem 2.14.

We have established that over various unate bases there is an important gap in the depths of symmetric functions. It is interesting to note that for each of the bases $\Omega = \text{NAND}, \{\text{NAND}, \rightarrow\}$ and $\{\rightarrow, 0\}$ there is no function f in S_n' such that

$$D_{\Omega}(f) = \log n + o(\log n)$$

although there are functions in S_n' which have linear formula size over Ω.

Finally, we give an easy corollary of Theorem 5.5 which establishes a significant gap in the depths of threshold functions over the monotone basis M_2.

Definition

Let $T_k^{(n)}(X_n)$ be the symmetric Boolean function which is 1 iff $\sum_{i=0}^{n-1} x_i \geqslant k$.

$T_n = \left\{ T_k^{(n)} \middle| 0 \leqslant k \leqslant n \right\} = \left\{ M_n \cap S_n \right\}$ is the set of threshold functions.

Noting the fact that $M_2 \subset U_2$, we have the following

Corollary 5.9

For all f in T_n, either

$$D_{M_2}(f) \leqslant \lceil \log n \rceil$$

or

$$D_{M_2}(f) \geqslant \log n + \log\log n - O(1)$$

$\square$

6. HIERARCHIES

In a previous chapter we noted that 'almost all' Boolean functions require large amounts of depth. However, at present we have only small lower bounds on the depth of explicitly defined functions and it would appear to be a difficult problem to substantially improve upon these lower bounds.

Given this situation, it is natural to ask whether there exist functions f in B_n of depth, say, $\log^2 n$ or $n^{\frac{1}{2}}$. In the absence of closely matching bounds for specific functions, we can answer such questions by demonstrating that the depths of Boolean functions form a reasonably uniform hierarchy. Hierarchies often yield valuable insight when exact bounds for specific functions are difficult to derive.

In this chapter we exhibit various hierarchies for sets of Boolean functions. These are obtained from bounds for 'almost all' functions by defining a sequence of functions in terms of subcircuits for some function of nearly maximal complexity and in some cases by employing a "padding" technique.

<u>Definitions</u>

$$\text{Depth}_{\Omega}(S,z) = \left\{ f \in S \,\middle|\, D_{\Omega}(f) \leqslant z \right\}$$

$$\text{C.Size}_{\Omega}(S,z) = \left\{ f \in S \,\middle|\, C_{\Omega}(f) \leqslant z \right\}$$

$$\text{F.Size}_{\Omega}(S,z) = \left\{ f \in S \,\middle|\, F_{\Omega}(f) \leqslant z \right\}$$

Lemma 6.1

For all complete binary bases Ω and all sufficiently large n,

$$\mathrm{Depth}_{\Omega}(B_n, i) \subsetneq \mathrm{Depth}_{\Omega}(B_n, i+1)$$

whenever $0 \leqslant i \leqslant n - \log\log n - O(1)$

Proof

In Chapter 3 we noted that for all such Ω and for all $n \geqslant 0$, there is some f in B_n such that

$$D_{\Omega}(f) = Z \geqslant n - \log\log n - O(1)$$

Let β be a circuit of depth Z over the basis Ω in which there is a gate ν which computes f.

If f_{ν_1}, f_{ν_2} are the functions computed by the pair of arcs entering ν, then we have either

$$D_{\Omega}(f_{\nu_1}) = Z-1$$

or

$$D_{\Omega}(f_{\nu_2}) = Z-1$$

Without loss of generality, assume

$$D_{\Omega}(f_{\nu_1}) = Z-1$$

Then by applying this argument inductively to f_{ν_1} and so on, we can derive a sequence of Boolean functions $\{f_i\}$ for $0 \leqslant i \leqslant Z$, where $D_{\Omega}(f_i) = i$. As each of these functions f_i depends on n or fewer arguments, we can define f_i to be some function in B_n

$\square$

Corollary 6.2

$$\mathrm{Depth}_{M_2}(M_n,i) \subsetneq \mathrm{Depth}_{M_2}(M_n,i+1)$$

whenever $0 \leqslant i \leqslant n - \frac{1}{2}\log n - \log\log n - O(1)$

Proof

Identical to Lemma 6.1, using Lemma 3.8

$\square$

Definition

Let B_n'' be the set of n argument Boolean functions which depend on all n arguments and let M_n'' be the corresponding set of monotone functions.

Lemma 6.3

For all complete binary bases Ω and all sufficiently large n,

$$\mathrm{Depth}_{\Omega}(B_n'',i) \subsetneq \mathrm{Depth}_{\Omega}(B_n'',i+2)$$

whenever $\lceil \log n \rceil \leqslant i \leqslant n - \log\log n - O(1)$

Proof

Using the method of Lemma 6.1 we can derive a sequence of Boolean functions $f_1, f_2, f_3, \ldots\ldots, f_Z$ where

$$D_{\Omega}(f_i) = i$$

$$Z \geqslant n - \log\log n - O(1)$$

and each f_i depends on α_i arguments, $1 \leq \alpha_i \leq n$ for all i.

For all complete binary bases Ω,

$$\Omega \cap B_2'' \neq \emptyset$$

Let $\odot$ be some function in $\Omega \cap B_2''$. Since $\odot$ may not be associative, we inductively define $\bigodot_{i=0}^{n-1} x_i$ to be $(\bigodot_{i=0}^{\lceil n/2\rceil -1} x_i) \odot (\bigodot_{i=\lceil n/2\rceil}^{n-1} x_i)$ for all $n \geq 2$, and note that $\bigodot_{i=0}^{n-1} x_i$ is in B_n''.

Let $f_i'(X_n) = \tilde{f}_i(X_{\alpha_i}) \odot \bigodot_{j=\alpha_i}^{n-1} x_j$ where $\tilde{f}_i = \overline{f}_i$ if $\odot \in \{NAND, NOR, \rightarrow, \leftarrow\}$ and $D_\Omega(\overline{f}_i) = i-1$, $\tilde{f}_i = f_i$ otherwise.

Then $f_1'(X_n), f_2'(X_n), \ldots, f_Z'(X_n)$ have the following properties .

i) $\forall i,\ 1 \leqslant i \leqslant Z,\ f_i' \in B_n''$ and thus $\forall i,\ 1 \leqslant i \leqslant Z,\ D_\Omega(f_i') \geqslant \lceil \log n \rceil$

ii) $\forall i,\ \lceil \log n \rceil \leqslant i \leqslant Z, \quad i \leqslant D_\Omega(f_i') \leqslant i+1$

The lower bound follows from the following two facts :

a) By setting $x_{\alpha_i}, \ldots, x_{n-1}$ to appropriate constants,in a circuit which computes $f_i'(X_n)$,we can obtain a circuit which computes $f_i(X_{\alpha_i})$ (or $\overline{f}_i$ if $\odot \in \{NAND, NOR, \rightarrow, \leftarrow\}$ and $D_\Omega(\overline{f}_i) \geq i$.)

b) $D_\Omega(f_i) = i$.

The upper bound follows from the fact that

$$D_\Omega(\bigodot_{j=\alpha_i}^{n-1} x_j) \leqslant \lceil \log n \rceil . \quad \text{Thus,}$$

$$D_\Omega(f_i') \leqslant \max \left\{ \lceil \log n \rceil , i \right\} + 1$$

The hierarchy then follows from these properties of the sequence $\{f_i'\}$

$\square$

Corollary 6.4

$$\mathrm{Depth}_{M_2}(M_n'', i) \subsetneq \mathrm{Depth}_{M_2}(M_n'', i+2)$$

whenever $\lceil \log n \rceil \leqslant i \leqslant n - \frac{1}{2} \log n - \log\log n - O(1)$

$\square$

Similar hierarchies can be obtained for formula size and circuit size.

Lemma 6.5

For all complete binary bases Ω and all sufficiently large n,

$$\mathrm{F.Size}_{\Omega}(B_n, i) \subsetneq \mathrm{F.Size}_{\Omega}(B_n, 2.i)$$

whenever $1 \leqslant i \leqslant 2^{n-1}/\log n$

Proof

The counting argument of Riordan and Shannon (1942) shows that for all such Ω and for all $n \geqslant 0$, there is some f in B_n such that $F_{\Omega}(f) = F \geqslant c.2^n/\log n$ for some constant c.

Let β be a formula of size F over the basis Ω in which there is a gate ν which computes f.

If f_{ν_1}, f_{ν_2} are the functions computed by the pair of arcs entering ν, then we have either

$$\lceil F/2 \rceil \leqslant F_{\Omega}(f_{\nu_1}) < F$$

or

$$\lceil F/2 \rceil \leqslant F_{\Omega}(f_{\nu_2}) < F$$

Without loss of generality, assume that

$$\lceil F/2 \rceil \leqslant F_{\Omega}(f_{\sim 1}) < F$$

By applying this argument inductively to $f_{\sim 1}$ and so on, we can derive a sequence of Boolean functions $\{f_i\}$ for $1 \leqslant i \leqslant p$, where

i) $\lceil \log F \rceil \leqslant p \leqslant F-1$

ii) $F_{\Omega}(f_1) = 2$, $F_{\Omega}(f_p) = F$

iii) $\forall i,\ 1 \leqslant i < p,\ F_{\Omega}(f_i) < F_{\Omega}(f_{i+1}) \leqslant 2.F_{\Omega}(f_i)$

As each of these functions f_i depends on not more than n arguments, we can define f_i to be some function in B_n.

Having defined the sequence of functions in this way, the lemma follows immediately

$\square$

Lemma 6.6

For all complete binary bases Ω and all sufficiently large n,

$C.Size_{\Omega}(B_n,i) \subsetneqq C.Size_{\Omega}(B_n,2i)$

whenever $1 \leqslant i \leqslant 2^{n-1}/n$.

Proof

Similar to Lemma 6.5, using the fact that for all such Ω and for all $n \geqslant 0$, there is some f in B_n such that $C_{\Omega}(f) \geqslant c.2^n/n$ for some constant c.

$\square$

7. CONCLUSION

We have presented a number of results on the circuit complexity of Boolean functions. For many of the problems considered there remain important gaps between the best known lower and upper bounds. The uniform hierarchies which we exhibited in the last chapter are particularly interesting in juxtaposition with the gap theorems for symmetric functions in Chapter 5. Taken together, these results show a number of cases where there is a Boolean function of a certain complexity but no symmetric function of that complexity. For example, consider Theorem 5.5 and Lemma 6.3. An example in the case of monotone complexity is provided by Corollaries 5.9 and 6.4.

Every nonconstant symmetric function depends on all its arguments. It seems that this property and others of the functions in S_n preclude the possibility of uniform complexity hierarchies for S_n. Our results give some formal justification for this intuition. Although, from the two examples explicitly mentioned above, we see that dependence on all arguments does not alone explain the sharp distinction between the complexity hierarchies for symmetric functions and those for sets such as B_n, M_n.

It has often been remarked that good theories rarely develop outside the context of a background of well understood real problems and special cases. Therefore, in order to build a realistic theory of computational complexity we ought to concentrate on acquiring a deeper understanding of particular problems and hope that from this we will be able to guess and prove more general principles. For this reason, the main aim of the research reported here was to acquire a deeper understanding of some particular problems concerning the circuit depth of Boolean functions.

This conservative attitude to the development of a theory of complexity is supported by the fact that there are already several instances where studies of a particular problem in computational complexity have shown intuition to be wrong, e.g. in the problems of integer multiplication, matrix multiplication and finding the median. Classically these problems take time n^2, n^3 and n log n respectively and intuition might suggest that these upper bounds are optimal. However, we now have procedures for these problems which only require time n log n loglog n, $n^{2\cdot 81}$ and 3n respectively. This shows that many of our beliefs which seem to be common sense may turn out to be false.

If we focus our attention on the complexity of finite functions, then there are several open questions of recognized importance. These include the problems of verifying Cook's conjecture on time versus space using results on circuit complexity, and of verifying that $P \neq NP$ by a circuit theoretic approach. Besides these major open problems there are a myriad of questions which remain unresolved. Many of these have been suggested in the preceding chapters.

Many problems of practical and theoretical interest can be conveniently formulated in terms of the circuit complexity of Boolean functions. Much work has recently been done in this area as can be seen from the extensive list of references. However, the large number of open problems, conjectures etc., indicate the embarrassingly large gaps which remain in our knowledge. This is primarily due to our inability to prove large lower bounds on the complexity of many familiar explicitly defined functions.

Although much has already been attained, even more remains to be done before we can achieve our ultimate goal, a realistic theory of computational complexity.

Appendix. SHORT FORMULAE FOR THRESHOLD FUNCTIONS.

Many sorting algorithms can be modelled by networks of comparator gates. A comparator network is a (non feedback) switching network, composed of 2-input 2-output gates, whose inputs are drawn from some totally ordered set (e.g. the non-negative integers). One output of each gate corresponds to the maximum of the two inputs and the other corresponds to the minimum. Such a network can only represent a subset of the sorting algorithms which could be carried out by a general purpose computer. It cannot, for example, model algorithms where the comparison tree is altered and pruned as more information becomes available about the ordering of the inputs.

We may use the Boolean notation $a \vee b$ and $a \wedge b$ for the maximum and minimum, respectively, of two numbers a and b. The interpretation of these expressions, however, depends upon the domain of the variables. This notation is convenient for the analysis of sorting networks since it permits the outputs of such a network to be described by monotone Boolean formulae.

It is well known that a network of comparator gates sorts n numbers if and only if, when interpreted as a network of $\wedge$ and $\vee$ gates, it realizes the set of threshold functions $T_k^{(n)}$, for $1 \leqslant k \leqslant n$. This can be easily seen by considering a network which sorts inputs from $\{0,1\}^n$.

Therefore, the inherent monotone complexity of threshold functions is closely related to the potential efficiency of sorting networks. In particular, the delay required by a network of comparator gates which sorts n numbers need be no more than $D_{M_2}(T_n)$.

In previous sections we have established a number of bounds on the depth of threshold functions over nonassociative bases such as NAND , $\{\text{NAND}, \rightarrow\}$. E.g. we have shown that over the basis NAND there is a lower bound of $2\lceil \log n \rceil - 3$ on the depth of $T_k^{(n)}$ for all $0 < k \leqslant n$. We now consider the depth and formula size of $T_k^{(n)}$, for $1 < k < n$, over bases such as B_2, U_2 and M_2. First we note two simple relations between threshold functions and elementary symmetric functions.

(i) For all $0 \leqslant k < n$, $E_k^{(n)} = T_k^{(n)} \wedge \overline{T}_{k+1}^{(n)}$.

(ii) For all $0 < k \leqslant n$, $T_k^{(n)} = \bigwedge_{i=0}^{k-1} \overline{E}_i^{(n)}$.

These identities show that over complete bases such as B_2, U_2, there is a close correlation between the complexities of functions in these two sets.

Hodes and Specker (1968) show that for all $1 < k < n$, $F_{B_2}(T_k^{(n)}) \geqslant n.t(n)$ where $t(n) \to \infty$ as $n \to \infty$, while M. S. Paterson has derived a lower bound of order $n \log n$ on the formula size of $T^{(n)}_{\lceil n/2 \rceil}$ over this basis. (A weaker version of the result which proves this lower bound appears in Fischer, Meyer and Paterson (1975)).

The result of Krichevskii (1964) can be used to show that for all $1 < k < n$, $F_{U_2}(T_k^{(n)}) \geqslant C.n \log n$ for some constant C while Khrapchenko (1972a) has proved that

$$F_{U_2}(T^{(n)}_{\lceil n/2 \rceil +1}) \geqslant \frac{(n+1)^2}{4}$$

As $M_2 \subset U_2 \subset B_2$, these lower bounds also hold for the monotone basis M_2.

We have already noted that $O(n^{3.55})$ is an upper bound on the formula size of all symmetric functions over the basis B_2 and thus on the formula size of $T^{(n)}_{\lceil n/2 \rceil}$. Similarly, Khrapchenko (1972b) describes a construction which shows that

$$F_{U_2}(T^{(n)}_{\lceil n/2 \rceil}) = O(n^{4.62})$$

If we consider the basis M_2, we find that the best known upper bound on the monotone depth of $T^{(n)}_{\lceil n/2 \rceil}$ is $\frac{1}{4}(\log n)^2 + O(\log n)$. This upper bound follows from a probabilistic argument derived independently by Khasin (1970) and Pippenger (1975). Pippenger refers to results obtained in this way as "existential propositions". In both cases a similar argument is used to show that $\log n + \log\log n + O(1)$ is an upper bound on the monotone depth of all threshold functions $T^{(n)}_k$ with fixed threshold k.

Erdös and Spencer (1974) have demonstrated the power of the probabilistic, or nonconstructive, method of proving theorems, i.e. proving that some member of a class has a certain property without actually constructing that member. Once a result has been established by nonconstructive methods, it often becomes interesting to obtain a proof by construction, i.e. to replace existential propositions by algorithms which construct mathematical objects.

Following this approach, we now describe several short formulae for $T^{(n)}_k$ where k is some fixed number independent of n. Explicit constructions are given for all formulae, some of which are non-monotone. However, only for $k = 2$ have we obtained an explicit formula of depth $\log n + \log\log n + O(1)$ or of length $O(n \log n)$ although these bounds should be possible for all fixed k.

1. THRESHOLD K, K < 7

Threshold 2

For $n = 2^k$, $T_2^{(n)}(X_n)$ may be expressed as

$$T_2^{(n/2)}(X^1_{n/2}) \vee T_2^{(n/2)}(X^2_{n/2}) \vee T_1^{(n/2)}(X^1_{n/2}) \wedge T_1^{(n/2)}(X^2_{n/2})$$

where $X^1_{n/2} = \langle x_0, x_1, \ldots, x_{n/2 -1} \rangle$ and $X^2_{n/2} = \langle x_{n/2}, x_{n/2+1}, \ldots, x_{n-1} \rangle$

By recursively applying this identity we obtain a monotone formula for $T_2^{(n)}(X_n)$ which, after reassociation, may be expressed as

$$\bigvee_{i=1}^{k} F_i$$

where F_i is a monotone formula of size n which contains every $x_i \in X_n$.

Hence, we have given an explicit construction which shows that

$$D_{M_2}(T_2^{(n)}) \leqslant \log n + \log\log n \text{ for } n = 2^k$$

If n is not a power of two, we can obtain a monotone formula for $T_2^{(n)}$ from one for $T_2^{(m)}$, where $n < m < 2n$ and m is a power of two. Therefore, we have an upper bound of $\log n + \log\log n + O(1)$ for all $n \geqslant 1$.

$\square$

We shall informally describe our short formulae for $k > 2$, using some unusual notation.

Definition

We shall assume a partition of X_n into $n^{\frac{1}{2}}$ blocks :

$X^1, X^2, X^3, \ldots, X^{n^{\frac{1}{2}}}$

each of size $n^{\frac{1}{2}}$.

Notation

$T_p(T_q)$ will denote

$$T_p^{(n^{\frac{1}{2}})}(T_q^{(n^{\frac{1}{2}})}(X^1), T_q^{(n^{\frac{1}{2}})}(X^2), \ldots, T_q^{(n^{\frac{1}{2}})}(X^{n^{\frac{1}{2}}}))$$

Threshold 3

$T_3^{(n)}$ may be expressed using the identity

$$T_3^{(n)} = T_1(T_3) \vee T_3(T_1) \vee (T_1(T_2) \wedge T_2(T_1))$$

From this identity we obtain the following recurrence relation for the formula size of $T_3^{(n)}$.

$$F_{M_2}(T_3^{(n)}) \leqslant 2n^{\frac{1}{2}}\, F_{M_2}(T_3^{(n^{\frac{1}{2}})}) + 2n^{\frac{1}{2}}\, F_{M_2}(T_2^{(n^{\frac{1}{2}})})$$

We have already noted that $F_{M_2}(T_2^{(n)}) = O(n \log n)$

$$\therefore\ F_{M_2}(T_3^{(n)}) \leqslant 2n^{\frac{1}{2}}\, F_{M_2}(T_3^{(n^{\frac{1}{2}})}) + O(n \log n)$$

Now if we let $F_{M_2}(T_3^{(n)}) = n.f(\log n)$, we have

$$n.f(\log n) \leqslant 2n^{\frac{1}{2}}n^{\frac{1}{2}}\ f(\tfrac{1}{2}\log n) + O(n \log n)$$

$$\therefore\ f(\log n) \leqslant 2f(\tfrac{1}{2}\log n) + O(\log n)$$

$$f(\log n) = O(\log n \operatorname{loglog} n)$$

and thus

$$F_{M_2}(T_3^{(n)}) = O(n \log n \operatorname{loglog} n)$$

$\square$

Open problem

Give an explicit monotone formula of size $O(n \log n)$ for $T_3^{(n)}$.

Threshold 4

$$\begin{aligned} T_4^{(n)} = {} & T_1(T_4) \vee T_4(T_1) \\ & \vee (T_3(T_1) \wedge T_1(T_2)) \\ & \vee (T_2(T_1) \wedge T_1(T_3)) \\ & \vee (T_2(T_1) \wedge \bigwedge_{i=1}^{n^{\frac{1}{2}}} \left[T_1(X^i) \rightarrow T_2(X^i)\right]\ ) \end{aligned}$$

and this identity yields the recurrence

$$F_{U_2}(T_4^{(n)}) \leqslant 2n^{\frac{1}{2}}\, F_{U_2}(T_4^{(n^{\frac{1}{2}})}) + O(n \log n \operatorname{loglog} n)$$

Therefore, from such a construction we obtain

$$F_{U_2}(T_4^{(n)}) = O(n \log n (\operatorname{loglog} n)^2)$$

$\square$

Threshold 5

$$\begin{aligned} T_5^{(n)} = {} & T_1(T_5) \vee T_5(T_1) \\ & \vee(T_2(T_1) \wedge T_1(T_4)) \\ & \vee(T_3(T_1) \wedge T_1(T_3)) \\ & \vee(T_4(T_1) \wedge T_1(T_2)) \\ & \vee\Big(\Big[\Big[\bigoplus_{i=1}^{n^{\frac{1}{2}}} T_2(X^i)\Big]=0\Big] \wedge (T_1(T_3).\mathrm{V}.T_3(T_1) \wedge T_1(T_2))\Big) \\ & \vee\Big(T_3(T_1) \wedge \bigwedge_{i=1}^{n^{\frac{1}{2}}} \big[T_1(X^i) \rightarrow T_2(X^i)\big]\Big) \end{aligned}$$

and this yields the recurrence

$$F_{B_2}(T_5^{(n)}) \leqslant 2n^{\frac{1}{2}}\, F_{B_2}(T_5^{(n^{\frac{1}{2}})}) + O(n \log n (\operatorname{loglog} n)^2)$$

from which we obtain

$$F_{B_2}(T_5^{(n)}) = O(n \log n (\operatorname{loglog} n)^3)$$

$\square$

For k = 5 and 6, our formulae for $T_k^{(n)}$ use "padding" to cover cases where $\sum_{i=0}^{n-1} x_i > k$ due to the non-monotonicity of the construction.

Threshold 6

$$T_6^{(n)} = T_1(T_6) \vee T_6(T_1)$$
$$\vee (T_5(T_1) \wedge T_1(T_2))$$
$$\vee (T_4(T_1) \wedge T_1(T_3))$$
$$\vee (T_3(T_1) \wedge T_1(T_4))$$
$$\vee (T_2(T_1) \wedge T_1(T_5))$$

$$\vee \left(\left[\left[\bigoplus_{i=1}^{n^{\frac{1}{2}}} T_2(X^i) \right] = 0 \right] \wedge (T_1(T_4) \vee (T_4(T_1) \wedge T_1(T_2)) \vee (T_3(T_1) \wedge T_1(T_3))) \right)$$

$$\vee \left(\left[\left[\bigoplus_{i=1}^{n^{\frac{1}{2}}} T_3(X^i) \right] = 0 \right] \wedge T_1(T_3) \right)$$

$$\vee \left(\bigwedge_{i=1}^{n^{\frac{1}{2}}} \left[T_1(X^i) \to T_2(X^i) \right] \wedge T_3(T_1) \right)$$

$$\vee (T_4(T_1) \wedge C_{4,3}^{(n)}(X_n))$$

In section 4.3 we noted that

$$F_{B_2}(C_{4,3}^{(n)}) = O(n \log n)$$

Therefore, the above identity yields the recurrence

$$F_{B_2}(T_6^{(n)}) \leqslant 2n^{\frac{1}{2}} F_{B_2}(T_6^{(n^{\frac{1}{2}})}) + O(n \log n (\log\log n)^3)$$

from which we obtain

$$F_{B_2}(T_6^{(n)}) = O(n \log n (\log\log n)^4)$$

$\square$

Open problem

Give an explicit formula of size less than order $n \log^2 n$ for $T_7^{(n)}$.

2. AN UPPER BOUND FOR ALL FIXED K.

Korobkov (1956) used the identity

$$T_k(X_n) = \bigvee_{i=0}^{k} T_i^{(n/2)} (x_o, x_1, \ldots, x_{n/_2 -1}) \wedge T_{k-i}^{(n/2)} (x_{n/2}, x_{n/_2 +1}, \ldots, x_{n-1})$$

and the method of dichotomy, or binary splitting, in order to obtain the bound

$$F_{M_2}(T_k^{(n)}) = O(n (\log n)^{k-1})$$

We now describe a construction which improves upon this upper bound.

Threshold 2^p, p fixed

Definition

$D_i^{(n)}(X_n)$ will represent the ith digit in the binary representation of $\sum_{i=0}^{n-1} x_i$.

Let the argument set X_{2n} be partitioned into two blocks X^1 and X^2 each of size n. The binary digits $D_i^{(2n)}(X_{2n})$ can be expressed recursively by constructing the digits for X^1, X^2 and performing a binary addition on the results.

This binary addition can be performed using a set of "full adders" which compute each new digit d and each new carry c from the digits d', d'' of the summands and the previous carry c'. By employing the formulae

$$d = d' \oplus d'' \oplus c'$$
$$c = c' \oplus ((c' \oplus d') \wedge (c' \oplus d''))$$

in these full adders, we obtain the following results :

For all $i \geq 0$,

$$D_i^{(2n)}(X_{2n}) = D_i^{(n)}(X^1) \oplus D_i^{(n)}(X^2) \oplus C_i$$

where C_i has a Boolean formula consisting of 3^j occurrences of $D_{i-j-1}^{(n)}(X^1)$ and 3^j occurrences of $D_{i-j-1}^{(n)}(X^2)$, for $0 \leq j < i$.

$$F_{B_2}(D_i^{(n)}(X_n)) = O(n(\log n)^i) \text{ for all fixed } i.$$

Now consider the function $T_{2^p}^{(n)}$ where p is fixed. We may express $T_{2^p}^{(n)}$ using the identity :

$$T_{2^p}^{(2n)}(X_{2n}) = T_{2^p}^{(n)}(X^1) \vee T_{2^p}^{(n)}(X^2)$$
$$\vee (C_{p-1} \oplus ((T_{2^{p-1}}^{(n)}(X^1) \oplus C_{p-1})$$
$$\wedge (C_{p-1} \oplus T_{2^{p-1}}^{(n)}(X^2)\ )\ )\ ).$$

This identity, together with the upper bound on $F_{B_2}(D_i)$ given above, yields the recurrence

$$F_{B_2}(T_{2^p}^{(2n)}) \le 2.F_{B_2}(T_{2^p}^{(n)}) + 2.F_{B_2}(T_{2^{p-1}}^{(n)}) + O(n(\log n)^{p-2}).$$

We can prove by induction on p starting from $p=2$ that

$$F_{B_2}(T_{2^p}^{(n)}) = O(n(\log n)^{p-1}.(\log\log n)^2)$$

For example with $p=3$

$$F_{B_2}(T_8^{(2n)}) \le 2.F_{B_2}(T_8^{(n)}) + 2.F_{B_2}(T_4^{(n)}) + O(n \log n)$$

and using our upper bound on $T_4^{(n)}$ we obtain

$$F_{B_2}(T_8^{(n)}) = O(n(\log n)^2(\log\log n)^2)$$

When k is not a power of 2, we can obtain a formula for $T_k^{(n)}$ from one for

$$T_{2^r}^{(n+2^r-k)}$$

where $2^{r-1} < k < 2^r$, by setting arguments to the value 1.

Hence, we have shown that for any fixed k,

$$F_{B_2}(T_k^{(n)}) = O(n(\log n)^{\lceil \log k \rceil -1}.(\log\log n)^2)$$

REFERENCES

Birkhoff, G. (1971)
The role of modern algebra in computing.
SIAM -AMS Proc. Vol.4, 1-47

Borodin, A. (1975)
Some remarks on Time-Space and Size-Depth.
Unpublished manuscript.

Brent, R.,D.Kuck and K.Maruyama (1973)
The parallel evaluation of arithmetic expressions without division.
IEEE Trans. Computers Vol.C-22, 532-534

Cook, S.A. (1971)
The complexity of theorem-proving procedures.
Proc. 3rd Annual ACM Symposium on Theory of Computing, 151-158

Cook, S.A. (1974)
An observation on Time-Storage trade-off.
JCSS 9, 308-316

Erdős, P and J. Spencer (1974)
Probabilistic methods in combinatorics.
Academic Press

Fischer, M.J., A.R. Meyer and M.S. Paterson (1975)
Lower bounds on the size of Boolean formulas : preliminary report.
Proc. 7th Annual ACM Symposium on Theory of Computing, 37-44

Gilbert, E.N. (1954)
Lattice theoretic properties of frontal switching functions.
J. Math. and Phys. 33,No.1, 57-97

Glushkov, V.M. (1966)
Introduction to cybernetics.
Academic Press

Harper, L.H. and J.E. Savage (1972)
On the complexity of the marriage problem.
Advances in Mathematics 9, 299-312

Hodes, L. and E. Specker (1968)
Lengths of formulas and elimination of quantifiers I,
in Contributions to Mathematical Logic (K. Schutte ed.),
175-188. North Holland.

Khasin, L.S. (1970)
Complexity bounds for the realization of monotonic symmetrical functions by means of formulas in the basis $\vee$, & , $\neg$.
Sov. Phys. Dokl. 14, 1149-1151 ; orig. in Doklady Akademii Nauk SSSR 189(4) (1969), 752-755

Khrapchenko, V.M. (1971)
Complexity of the realization of a linear function in the class of Π-circuits.
Math. Notes of the Academy of Sciences of the USSR 9, 21-23 ; orig. in Matematicheskie Zametki 9,1, (1971), 35-40

Khrapchenko, V.M. (1972a)
Method of determining lower bounds for the complexity of P-schemes.
Math. Notes of the Academy of Sciences of the USSR **10**, 474-479 ; orig. in Matematicheskie Zametki **10**,**1**, (1971), 83-92

Khrapchenko V.M. (1972b)
The complexity of the realization of symmetrical functions by formulae.
Math. Notes of the Academy of Sciences of the USSR **11**, 70-76 ; orig. in Matematicheskie Zametki **11**,**1**, (1972), 109-120

Kleitman, D. (1969)
On Dedekind's problem : The number of monotone Boolean functions.
Proc. AMS **21**, 677-682

Korobkov, V.K. (1956)
Realization of symmetric functions in the class of Π-circuits. (Russian)
Dokl. Akad. Nauk SSSR **109**, 260-263

Krichevskii, R.E. (1964)
Complexity of contact circuits realizing a function of logical algebra.
Sov. Phys. Dokl. **8**, 770-772 ; orig. in Doklady Akademii Nauk SSSR **151**(**4**), (1963), 803-806

Lupanov, O.B. (1958)
Ob odnom metode sinteza skhem.
Izv. V.U.Z. (Radiofizika) No.1, 120-140

Lupanov, O.B. (1962)
Complexity of formula realization of functions of logical algebra.
Problems of Cybernetics 3, 782-811 ; orig. in Sborn. Problemi Kibernetiki 3 (1960), 61-80

McColl, W.F. (1976)
The depth of Boolean functions.
Proc. 3rd Int. Colloquium on Automata, Languages and Programming, 307-321.
Edinburgh University Press

McColl, W.F. and M.S. Paterson (1975)
The depth of all Boolean functions..
To appear in SIAM J. on Computing, 6,2 (1977)

Neciporuk, E.I. (1966)
A Boolean function.
Sov. Math. Dokl. 7, 999-1000 ; orig. in Doklady Akademii Nauk SSSR 169(4),(1966), 765-766

Paterson, M.S. (1975)
Complexity of monotone networks for Boolean matrix product.
Theoretical Computer Science 1, 13-20

Paterson, M.S. (1976)
An introduction to Boolean function complexity.
Stanford Computer Science Report STAN-CS-76-557, 19pp.
also in Astérisque (journal of the French Mathematical Society) 38-39, 183-201

Paterson, M.S. and L.G. Valiant (1976)
Circuit size is nonlinear in depth.
Theoretical Computer Science 2, 397-400

Pippenger, N. (1974)
Short formulae for symmetric functions.
IBM Research Report RC 5143
Yorktown Heights.

Pippenger, N. (1975)
Short monotone formulae for threshold functions.
IBM Research Report RC 5405
Yorktown Heights.

Pippenger, N. (1976)
The realization of monotone Boolean functions.
Proc. 8th Annual ACM Symposium on Theory of Computing,
204-209

Post, E.L. (1941)
Two-valued iterative systems of mathematical logic.
Annals of Math. Studies 5
Princeton University Press

Pratt, V.R. (1975)
The effect of basis on size of Boolean expressions.
Proc.16th Annual IEEE Symposium on Foundations of
Computer Science, 119-121

Pratt, V.R. and L.J. Stockmeyer (1976)
A characterization of the power of vector machines.
JCSS 12, 198-221

Preparata, F.P. and D.E. Muller (1971)
On the delay required to realise Boolean functions.
IEEE Trans. Computers Vol.C-20, 459-461

Preparata, F.P. and D.E. Muller (1976)
Efficient parallel evaluation of Boolean expressions.
IEEE Trans. Computers Vol.C-25, 548-549

Reznik, V.I. (1962)
The realization of monotonic functions by means of networks consisting of functional elements.
Sov. Phys. Dokl. 6, 558-561 ; orig. in Doklady Akademii Nauk SSSR 139(3),(1961), 566-569

Riordan, J. and C.E. Shannon (1942)
The number of two-terminal series-parallel networks.
J. Math. and Phys. 21, 83-93

Schnorr, C.P. (1974)
Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen.
Computing(Arch. Elektron. Rechnen.) 13, 155-171

Schnorr, C.P. (1976)
The combinational complexity of equivalence.
Theoretical Computer Science 1,4, 289-295

Shannon, C.E. (1949)
The synthesis of two-terminal switching circuits.
Bell System Technical Journal 28, 59-98

Soprunenko, E.P. (1965)
On the minimal realization of certain functions by schemes of functional elements. (Russian)
Probl. Kib. 15, 117-134

Spira, P.M. (1971)
On the time necessary to compute switching functions.
IEEE Trans. Computers Vol.C 20, 104-105

Stockmeyer, L.J. (1976)
On the combinational complexity of certain symmetric Boolean functions.
IBM Research Report RC 5829
Yorktown Heights.

Vilfan, B. (1972)
The complexity of finite functions.
Ph.D. thesis, M.I.T.
Project MAC Tech. Report MAC-TR-97
M.I.T.

Zhegalkin, I.I. (1927)
The technique of calculation of statements in symbolic logic.
Matem. Sbornik 34, 9-28