INFORMATION AND COMPUTATION **96**, 168–178 (1992)

Branching Programs Provide Lower Bounds on the Areas of Multilective Deterministic and Nondeterministic VLSI-Circuits

JURAJ HROMKOVIČ

Department of Theoretical Cybernetics, Comenius University, 842 15 Bratislava, Czechoslovakia

AND

MATTHIAS KRAUSE AND CHRISTOPH MEINEL

Sektion Mathematik, Humboldt-Universität zu Berlin, 1086 Berlin, PF 1297, Germany

AND

STEPHEN WAACK

Karl-Weierstrass-Institut für Mathematik, 1086 Berlin, PF 1304, Germany

Each (nondeterministic) multilective VLSI-circuit C of area A can be simulated by an oblivious (disjunctive) branching program of width $\exp(O(A))$ which has the same multiplicity of reading as C. That is why exponential lower bounds on the width of (disjunctive) oblivious branching programs of linear depth provide lower bounds of order $\Omega(n^{1-2\alpha})$, $0 \leqslant \alpha < \frac{1}{2}$, on the area of (nondeterministic) multilective VLSI-circuits computing explicitly defined one-output Boolean functions, if the multiplicity of reading is bounded by $O(\log^{\alpha} n)$. Lower bounds are derived for the *sequence equality problem* (SEQ) and the *graph accessibility problem* (GAP). © 1992 Academic Press, Inc.

1. INTRODUCTION AND MAIN RESULT

VLSI-circuits represent a basic model of computation which has gained great importance in practical applications. VLSI-complexity theory which deals with the study of the computational power of these VLSI-circuits has to tackle two major issues. These consist, first, in proving good upper bounds, i.e., to present area and/or time efficient VLSI-circuit designs performing a given task, and, second, in deriving nontrivial lower bounds, i.e., to explore minimal resources (time, area, combinations of time and area) needed to solve a certain computational problem. The present paper is devoted to the second issue. It contributes a new and quite powerful method for proving lower bounds on the area A of (deterministic or nondeterministic) multilective VLSI-circuits (i.e., VLSI-circuits which are allowed to read input variables more than once) that compute explicitly defined Boolean functions. The method is based on deriving exponential lower bounds on the width of length bounded oblivious branching programs obtained recently by Krause, Meinel, and Waack (1989). Its novelty is that it is applicable to one-output functions while so far known techniques (see, e.g., Savage (1984), Siegel (1984)) work for multi-output problems only.

In detail, we prove the following theorem:

THEOREM. *Let k be fixed. If C is a multilective VLSI-circuit that reads each input variable at most k times and computes the Boolean function sequence equality (resp. graph accessibility), then the area of C is $\Omega(n)$. This result still holds if C is a nondeterministic multilective VLSI-circuit. If each (ordinary) variable is read at most $O(\log^{\alpha} n)$ times, $\alpha < \frac{1}{2}$ ($\alpha < \frac{1}{4}$), the lower bound becomes $\Omega(n^{1-2\alpha})$.*

The theorem is proved by combining the following two facts:

(a) A (nondeterministic) multilective VLSI-circuit of area A can be simulated by a (nondeterministic) oblivious branching program of width $O(2^{3A})$ whose length equals the number of clock periods the VLSI-circuit gets input values from outside during a computation.

(b) Each oblivious branching program of length $O(n \cdot \log^{\alpha} n)$, $0 \leqslant \alpha < \frac{1}{2}$ ($\alpha < \frac{1}{4}$) that computes the Boolean function *sequence equality* (*graph accessibility*) is of exponential width.

Part (a) is an improvement of a simulation method described in Hromkovič and Prochàzka (1988). It will be proved in Section 4. The lower bounds of Part (b) were proved in Krause, Meinel, and Waack (1989) and are quoted in Section 5. In Section 2 we recall the definition of multilective VLSI-circuits (see, for example, Ullman (1984), Hromkovič (1988)) and introduce the corresponding nondeterministic VLSI-circuit model. Section 3 presents μ-branching programs and disjunctive μ-branching programs which generalize the well-known branching program model (see, e.g., Meinel (1988)).

For all natural numbers n we denote by X_n a collection $\{x_1, ..., x_n\}$ of n Boolean variables, and by $\mathbb{B}_n$ the set of all Boolean functions

f: $\{0, 1\}^n \to \{0, 1\}$. Let A be a language over the alphabet $\{0, 1\}$ and let $A^n = A \cap \{0, 1\}^n$. Throughout this paper we make no difference between sets and their characteristic functions.

2. Multilective VLSI-Circuits

Recall the definition of multilective VLSI-circuits given, e.g., in Ullman (1984) or Hromkovič (1988).

A *grid-graph* G is a directed graph drawn in a rectangular grid such that each square of the grid has one of the following contents:

(1) filled up,

(2) a straight line in horizontal or vertical direction,

(3) a bent line entering the grid square in the horizontal or vertical direction and leaving the square in the vertical or horizontal direction, respectively,

(4) a crossing of a horizontal line and a vertical line, or

(5) the empty content.

The interpretation of these contents is as follows. A grid square of type (1) corresponds to a part of the layout of a node of G, which is usually assumed to be a convex region of squares. Grid squares of types (2) and (3) correspond to parts of the layout of an edge of G and squares of type (4) depict the place of the crossing of some two edges of G. The *area* of a grid-graph is the area of a minimal rectangle comprising all nonempty squares of the grid.

A *multilective VLSI-circuit* C *over the set* $X_n = \{x_1, ..., x_n\}$ of input variables is a synchronous sequential circuit which is laid out as a grid-graph. The *area* $A(C)$ of C is the area of a minimal grid-graph which lays out C.

In more detail, we assume that

— C has some *input nodes* (i.e., nodes of indegree zero) and one *outpute node* (i.e., a node of outdegree zero) different from the input nodes.

— Each inner node v of C is labelled with some operation $p(v)$: $\{0, 1\}^i \to \{0, 1\}^j$, where i equals the indegree and j the outdegree of v.

— C has a fixed working time $T(C) \in \mathbb{N}$. At each time t, $0 \leqslant t \leqslant T(C)$, all edges are labelled by some Boolean constant from $\{0, 1\}$ representing the bit transmitted at time t along this edge. At time $t = 0$ all edges of C are labelled by 0.

The computation performed by C is based on an *I/O-schedule* I, which for each time unit $t \in \{0, 1, ..., T(C)\}$ defines a labelling $I(t)$ of the input nodes by elements from $X_n \cup \{0, 1\}$. By $X(t)$, $t \in \mathbb{N}$, we denote the set of those input variables which are "*read*" by C at time unit t, i.e., which occur as a label of an input node via $I(t)$. C is said to *read at time t* if $X(t) \neq \emptyset$.

If we assume each input node (the single output node) to be equipped with an additional input edge (output edge), than we can describe *internal states* s of C completely by assignments of Boolean constants from $\{0, 1\}$ to the edges of C. Let $\alpha = (\alpha_1, ..., \alpha_n) \in \{0, 1\}^n$ be an input vector. The *initial state* s_0, i.e., the state at time $t = 0$, assigns

— the value $\alpha_i \in \{0, 1\}$, $1 \leqslant i \leqslant n$, to the additional input edges of those input nodes labelled by x_i under $I(0)$, and

— the value 0 to the (original) edges of C and to the additional output edge.

If C is in state s at time t, $0 \leqslant t < T(C)$, then, at time $t+1$, C enters the state s' that assigns

— the value $\alpha_i \in \{0, 1\}$, $1 \leqslant i \leqslant n$, to the additional input edges corresponding to those input nodes that are labelled by x_i under $I(t+1)$,

— the value $p(v)(\alpha_{v,1}, ..., \alpha_{v,i})$ to the additional output edge that corresponds to the output node v of indegree i, where $\alpha_{v,l}$ are the values assigned by state s (at time t) to the edges $e_{v,l}$, $1 \leqslant l \leqslant i$, leading to v, and

— the value $\mathit{projection}_m\,(p(v)(\alpha_{v,1}, ..., \alpha_{v,i}))$ to edges $e'_{v,m}$, $1 \leqslant m \leqslant j$, leaving the inner node v of indegree i and outdegree j, where $\alpha_{v,l}$ are the values assigned by state s (at time t) to the dges $e_{v,l}$, $1 \leqslant l \leqslant i$, leading to v.

The *output value* $C(\alpha)$ of C is defined to be the value of the additional output edge of C at time $T(C)$. We say that C *computes* the Boolean function in $\mathbb{B}_n$ that associates the value $C(\alpha)$ with $\alpha \in \{0, 1\}^n$. This function is also denoted by C.

A multilective VLSI-circuit C is called a *VLSI*$(k(n))$*-circuit*, $k\colon \mathbb{N} \to \mathbb{N}$, if each input variable x is read at most $k(n)$ times; i.e., there are at most $k(n)$ time units t with $x \in X(t)$. Obviously, VLSI(1)-circuits, i.e., multilective VLSI-circuits for which each variable is available exactly once, are *multiplective VLSI-circuits*.

Finally, let us introduce multilective nondeterministic VLSI-circuits. Let $Y_m = \{y_1, ..., y_m\}$ be another set of Boolean variables. Let C be a multilective VLSI-circuit over the set $X_n \cup Y_m$ of input variables. C can be thought to be a *multilective nondeterministic VLSI-circuit* (for short multilective *ND-VLSI-circuit*) over the sets X_n of deterministic and Y_m of nondeterministic variables which computes a Boolean function from $\mathbb{B}_n$ if we modify the definition of acceptance in the following way: C accepts an input $\alpha \in \{0, 1\}^n$ if there is an assignment $\beta \in \{0, 1\}^m$ for the nondeterministic variables such that C computes the output 1 on $\alpha \cup \beta$, i.e., $C(\alpha, \beta) = 1$.

C is said to be a *$ND(l(n)) - VLSI(k(n))$-circuit*, 1, k: $\mathbb{N} \to \mathbb{N}$, if each nondeterministic variable is read at most $l(n)$ times and each deterministic variable is read at most $k(n)$ times.

3. Branching Programs

A *μ-branching program* (*$\mu - BP$* for short) P, $\mu \in \mathbb{N}$, over the set $X_n = \{x_1, ..., x_n\}$ of Boolean variables is a finite, directed, labelled, acyclic graph that has the following properties.

— P contains exactly one node of indegree zero (the *source*) and two nodes of outdegree zero (the *sinks* of the program). One sink is labelled by 1, the other by 0.

— Each non-sink v of P has outdegree 2^μ and is labelled by a subset $X(v)$ of X_n with $\# X(v) = \mu$.

— The edges of P are labelled by elements from $\{0, 1\}^\mu$ such that all 2^μ edges leaving a nonsink node v of P have different labels. I.e., for all nonsinks v of P each vector from $\{0, 1\}^\mu$ occurs exactly once as a label of an edge leaving v.

Each μ-branching program P over X_n *computes* a function $f \in \mathbb{B}_n$ as follows. Given an input $\alpha = (\alpha_1, ..., \alpha_n) \in \{0, 1\}^n$, the computation starts at the source of P. If the computation reaches a node v of P labelled by $X(v) = \{x_{i_1}, ..., x_{i_\mu}\} \subseteq X_n$ it follows the edge labelled by $(\alpha_{i_1}, ..., \alpha_{i_\mu}) \in \{0, 1\}^\mu$. The *value* $f(\alpha)$ is given by the label of the sink reached by the computation.

The *depth* of P, denoted by Depth(P), is defined to be the length of a longest path in P. The *size* of P, denoted by Size(P), is the number of non-sink nodes of P.

A μ-branching program P over X^n is said to be *k-times-only*, $k \in \mathbb{N}$, with regard to an input variable $x \in X_n$ if on each path in P there are at most. k non-sink nodes with a label containing x. P is called a *k-times-only μ-branching program* if P is k-times-only with regard to each input variable $x \in X_n$. Finally, a sequence $\{P_n\}_{n \in \mathbb{N}}$ of μ-branching programs P_n is said to be *$k(n)$-times-only*, k: $\mathbb{N} \to \mathbb{N}$, if, for each $n \in \mathbb{N}$, P_n is $k(n)$-times-only.

A μ-branching program P is *oblivious* if the set of nodes of P is partitioned into disjoint levels $L_1, ..., L_l, L_{l+1}$ in the following way:

— for each edge (v, v') of P there is an index i, $1 \leqslant i \leqslant l$, such that $v \in L_i$ and $v' \in L_{i+1}$, and

— all nonsink nodes at the same level have the same label.

Obviously, the parameter l coincides with the depth of P. Level L_{l+1} contains merely the two sinks.

For each oblivious μ-branching program P, Width(P) denotes the maximum taken over the cardinalities of all levels of P.

It is an easy exercise to prove the following fact.

Remark 3.1. For each oblivious $k(n)$-times-only μ-branching program P over X_n it holds that

$$\text{Depth}(P) \leqslant k(n) \cdot n/\mu.$$

Let $Y_m = \{y_1, ..., y_m\}$ be another set of Boolean variables, and let P be a μ-branching program over $X_n \cup Y_m$. As in the case of circuits, P can be interpreted as a *nondeterministic μ-branching program* over the set X_n of deterministic and the set Y_m of nondeterministic variables. The program P computes a Boolean function $f \in \mathbb{B}_n$ in such a way that it accepts an input $\alpha \in \{0, 1\}^n$ iff there is an assignment $\beta \in \{0, 1\}^m$ to the nondeterministic variables such that P yields the output "1" on the input (α, β). A nondeterministic μ-branching program P is called a *disjunctive μ-branching program* if it is 1-time-only with regard to the nondeterministic variables occuring in P (Meinel, 1987).

It is easy to verify that in the case $\mu = 1$ all definitions concerning general, $k(n)$-times-only, oblivious and nondeterministic μ-branching programs agree with the corresponding definitions for ordinary branching programs. Consequently, in the following we will speak of branching programs in the case of 1-branching programs. Branching programs represent a basic model of sequential computation which is well investigated (see, e.g., Meinel (1988)). For example, the classes of languages which are computable by sequences of polynomial size deterministic or disjunctive branching programs coincide with the fundamental logarithmic space-bounded Turing machine complexity classes $\mathscr{L}$ or $\mathscr{NL}$, respectively (Pudlák and Žak, 1983; Meinel, 1986).

It is not difficult to show that each Boolean function $f \in \mathbb{B}_n$ can be computed by an oblivious 1-time-only branching program over X_n whose underlying graph is a tree of size 2^n. (For $n = 1$ this assumption is obvious. If $n > 1$ and $f \in \mathbb{B}_n$ then we obtain the tree T_f for f by connecting the trees $T_{f,0}$ and $T_{f,1}$ for the functions $f(x_1, ..., x_{n-1}, 0)$ and $f(x_1, ..., x_{n-1}, 1)$ from $\mathbb{B}_{n-1}$, respectively, via a new source v labelled by x_n and two edges labelled by $\delta \in \{0, 1\}$ such that the edge with label δ connects v with the source of $T_{f,\delta}$.)

By the same idea, each nonsink node v in a given μ-branching program can be simulated by an oblivious 1-time-only branching program over $X(v)$ of size 2^μ. This fact leads to the following remark.

Remark 3.2. Each (nondeterministic) μ-branching program P can be simulated by a (nondeterministic) branching program P' with

$$\text{Depth}(P') \leqslant \mu \cdot \text{Depth}(P) \quad \text{and} \quad \text{Size}(P') \leqslant \text{Size}(P) \cdot 2^\mu.$$

Moreover, if P is $k(n)$-times-only for an input variable $x \in X_n$ then P' has the same property. Especially, if P is a $k(n)$-times-only μ-branching program, or disjunctive, then the same is true for P'.

Finally, if P is oblivious then P' is also oblivious, and

$$\mathrm{Width}(P') \leqslant \mathrm{Width}(P) \cdot 2^{\mu}.$$

4. The Simulation Result

Adapting a construction of Hromkovič and Procházka (1988) we can simulate multilective VLSI-circuits by branching programs.

Proposition 4.1. *Let C be a VLSI $(k(n))$-circuit, $k\colon \mathbb{N} \to \mathbb{N}$, over the set $X_n = \{x_1, ..., x_n\}$ of Boolean input variables. Then there is an oblivious $k(n)$-times-only branching program P over X_n with*

$$\mathrm{Width}(P) \leqslant 2^{3A(C)}$$

that simulates C, i.e., that computes the same function as C.

Proof. Due to Remarks 3.1 and 3.2 it suffices to construct an oblivious $k(n)$-times-only $A(C)$-branching program P over X_n with $\mathrm{Width}(P) \leqslant 2^{2A(C)}$, that simulates C.

Let $S = \{s_1, ..., s_q\}$ be the set of internal states of C. Since C has at most $2A(C)$ edges, the number of internal states of C is bounded by $2^{2A(C)}$, i.e., $q \leqslant 2^{2A(C)}$.

Let $t_1 < \cdots < t_d$ be those time units at which C reads. For short, let $X(i) = X(t_i)$. As the number of input nodes of C is bounded by $A(C)$ we have $\# X(i) \leqslant A(C)$ for all i, $1 \leqslant i \leqslant d$. For convenience, let $t_{d+1} = T(C)$.

We construct the desired $A(C)$-branching program P from nodes v from the set $S \times \{1, ..., d+1\}$, which are arranged in $d+1$ levels $L_i \subseteq S \times \{i\}$, $1 \leqslant i \leqslant d+1$.

If $s_1 \in S$ denotes the internal state of C at time t_1 then we take node $(s_1, 1)$ as source of P. (Note that s_1 does not depend on the input.) Inductively, we define level L_{i+1} and the edges connecting levels L_i and L_{i+1}, $1 \leqslant i \leqslant d$, as follows: $((s, i), (s', i+1))$, $s, s' \in S$, is an edge from level i to level $i+1$ if there is an assignment α to the variables of $X(i)$ such that C is in state s' at time t_{i+1} if C starts in state s at time t_i with input α. (Obviously, the state s' is well-defined since C works deterministically, and the input does not affect the computation of C in the time period (t_i, t_{i+1}).) Level L_{i+1} consists of all nodes $(s', i+1)$ which are reachable in such a way from nodes of level L_i.

Clearly, P is an oblivious $A(C)$-branching program of width not greater than $2^{2A(C)}$ which computes the same Boolean function from $\mathbb{B}_n$ as C. Since C is a VLSI($k(n)$)-circuit, P is $k(n)$-times-only. ▌

By exactly the same argument we can prove a similar result for nondeterministic multilective VLSI-circuits.

PROPOSITION 4.2. *Let C be a $ND(l(n)) - VLSI(k(n))$-circuit, $l, k: \mathbb{N} \to \mathbb{N}$, over the set X_n of deterministic and the set Y_m of nondeterministic variables. Then there is a nondeterministic oblivious branching program P over the set X_n of deterministic and the set Y_m of nondeterministic variables with*

$$\text{Width}(P) \leqslant 2^{3A(C)}$$

that simulates C and that is $k(n)$-times-only with regard to the deterministic variables and $l(n)$-times-only with regard to the nondeterministic ones, i.e.,

$$\text{Depth}(P) \leqslant k(n)n + l(n)m.$$

In particular, for nondeterministic VLSI-circuits that are 1-time-only with regard to the nondeterministic variables we obtain the following corollary.

COROLLARY. *Each $ND(1) - VLSI(k(n))$-circuit C, $k: \mathbb{N} \to \mathbb{N}$, over the set X_n of deterministic variables and the set Y_m of nondeterministic ones, $m = O(n)$, can be simulated by a disjunctive oblivious branching program P over X_n with*

$$\text{Width}(P) \leqslant 2^{3A(C)} \qquad \text{and} \qquad \text{Depth}(P) \leqslant O(n \cdot k(n)).$$

5. LOWER BOUNDS

In this section we derive lower bounds of order $\Omega(n^{1+2\alpha})$, $0 \leqslant \alpha < \frac{1}{2}$, on the area of VLSI($k(n)$)-circuits and nondeterministic ND(1) − VLSI($k(n)$)-circuits, $k(n) = O(\log^{\alpha} n)$, which compute some explicitly defined Boolean functions. These results are obtained on the basis of Proposition 4.1 and Proposition 4.2, respectively, which allow us to apply a method for proving exponential lower bounds on the size of ordinary and disjunctive oblivious $k(n)$-times-only branching programs recently developed in (Krause, Meinel, and Waack, 1989).

Let us consider the two problems GAP and SEQ, which are defined over certain sets $\mathbb{M}_{t,n}$ of Boolean $t \times n$ matrices, $t, n \in \mathbb{N}$.

Recall that each Boolean matrix $A \in \mathbb{M}_{n,n}$ can be thought of as the adjacency matrix of a directed graph $G(A)$ with n vertices $\{1, ..., n\}$ and edges leading from vertex i to vertex j iff $A_{i,j}=1$, $1 \leqslant i, j \leqslant n$. The *graph accessibility problem* $\mathrm{GAP}=(\mathrm{GAP}_n)_{n\in\mathbb{N}}$ consists of the sequence of Boolean functions GAP_n which decide for a given Boolean $n\times n$ matrix A whether vertex n is reachable from vertex 1 in the directed graph $G(A)$, i.e.,

$$\mathrm{GAP}_n(A)=1 \quad \text{iff} \quad \begin{array}{l}\text{there is a path from vertex 1 to}\\ \text{vertex } n \text{ in the directed graph } G(A).\end{array}$$

Let $A\in\mathbb{M}_{2,n}$, let $|A_j|$, $0\leqslant|A_j|\leqslant 2$, denote the column sum of column j, $1\leqslant j\leqslant n$, and let $w(A)$ denote the vector of these column sums,

$$w(A)=(|A_1|, ..., |A_n|).$$

For each word $w\in\{0,1,2\}^t$ let $\mathrm{red}(w)\in\{0,1\}^*$ be the reduced word which is obtained from w by deleting all occurences of the letter "2" in w. For example, if $w=(20120210)\in\{0,1,2\}^8$, then $\mathrm{red}(w)=(01010)$. The *sequence equality problem* $\mathrm{SEQ}=(\mathrm{SEQ}_n)_{n\in\mathbb{N}}$ consists of the sequence of Boolean functions SEQ_n which test for two given matrices A, $A'\in\mathbb{M}_{2,n}$ whether the reduced words of their column sums vectors coincide, i.e.,

$$\mathrm{SEQ}_n(A,A')=1 \qquad \text{iff} \qquad \mathrm{red}(w(A))=\mathrm{red}(w(A')).$$

The following lower bounds on the width of disjunctive oblivious branching programs of length $O(n\cdot\log^\alpha n)$, $0\leqslant\alpha<\frac{1}{2}$ $(0\leqslant\alpha<\frac{1}{4})$ computing the *sequence equality problem* SEQ or the *graph accessibility problem* GAP could be proved in Krause, Meinel, and Waack (1989).

PROPOSITION 5.1 (Krause, Meinel, and Waack, 1989). *For each* $n\in\mathbb{N}$ *let* Q_n *be a disjunctive oblivious branching program computing* SEQ_n *with* $\mathrm{Depth}(Q_n)=O(n\cdot\log^\alpha n)$, $0\leqslant\alpha<\frac{1}{2}$. *Then*

$$\mathrm{Width}(Q_n)=\exp(\Omega(n^{1-2\alpha})).$$

PROPOSITION 5.2 (Krause, Meinel, and Waack, 1989). *For each* $n\in\mathbb{N}$ *let* $N=n^2$ *and* P_n *be a disjunctive oblivious branching program computing* GAP_n *with* $\mathrm{Depth}(P_n)=O(N\cdot\log^\alpha N))$, $0\leqslant\alpha<\frac{1}{4}$. *Then*

$$\mathrm{Width}(P_n)=\exp(\Omega(n^{1-2\alpha}))=\exp(\Omega(N^{(1-2\alpha)/2}).$$

(Observe that N denotes the number of variables GAP_n depends on.)

Due to the simulation result of Proposition 4.2 the exponential lower bounds on the width of disjunctive branching programs given in Propositions 5.1 and 5.2 immediately imply, as already stated, a lower bound on the area of nondeterministic multilective VLSI-circuits with a bounded multiplicity of reading. These bounds, of course, include corresponding lower bounds for deterministic multilective VLSI-circuits.

THEOREM. *Let $k(n) = O(n \cdot \log^{\alpha} n)$, $0 \leqslant \alpha < \frac{1}{2}$. For each $n \in \mathbb{N}$ let S_n be a $ND(1) - VLSI(k(n))$-circuit computing the Boolean function* SEQ_n. *Then*

$$A(S_n) = \Omega(n^{1-2\alpha}).$$

Let $k(n) = O(n \cdot \log^{\alpha} n)$, $0 \leqslant \alpha < \frac{1}{4}$. For each $n \in \mathbb{N}$ let C_n be a $ND(1) - VLSI(k(N))$-circuit computing the Boolean function GAP_n. *(Observe that C_n is defined over $N = n^2$ input variables.) Then*

$$A(C_n) = \Omega(N^{(1-2\alpha)/2}) = \Omega(n^{1-2\alpha}).$$

In particular, if $\alpha = 0$ then $k(n)$ is a constant function and we obtain

COROLLARY. *Let $k \in \mathbb{N}$ be arbitrarily fixed. For each $n \in \mathbb{N}$ let S_n and C_n be $ND(1) - VLSI(k)$-circuits computing the Boolean functions* SEQ_n *and* GAP_n, *respectively. Then*

$$A(S_n) = \Omega(n) \qquad \textit{and} \qquad A(C_n) = \Omega(N^{1/2}) = \Omega(n).$$

Finally, let us only remark that the lower bound for the *sequence equality problem* SEQ_n is optimal. This is true since we have assumed processing elements with unrestricted computational power in our definition of multilective (nondeterministic) VLSI-circuits. Hence, each function f can be trivially computed by a circuit C with one processing node of area

$$A(C) = \text{fan-in}(f) + \text{fan-out}(f).$$

REVEIVED May 14, 1989; FINAL MANUSCRIPT RECEIVED June 18, 1990

REFERENCES

HROMKOVIČ, J. (1988), Some computational aspects of VLSI computations, *Comput. Artificial Intelligence* 7, No. 3, 229.

HROMKOVIČ, J., AND PROCHÁZKA, J. (1988), Branching programs are a tool for proving lower bounds on VLSI-computations and optimal algorithms for systolic arrays, *in* "Proceedings of MFCS' 88," pp. 360–370, Lecture Notes in Computer Science, Vol. 324, Springer-Verlag, Berlin/New York.

KRAUSE, M., MEINEL, CH., AND WAACK, S. (1989), Separating complexity classes related to certain input oblivious logarithmic space-bounded Turing machines, *in* "Proceedings of Structure in Complexity Theory, Oregon, 1989," pp. 240–249.

MEINEL, CH. (1986), *P*-projection reducibility and the complexity classes L(nonuniform) and NL(nonuniform), *in* "Proceedings of FCT' 86," pp. 527–535, Lecture Notes in Computer Science, Vol. 233, Springer-Verlag, Berlin/New York.

MEINEL, CH. (1987), The power of nondeterminism in polynomial-size bounded-width branching programs, *in* "Proceedings of FCT' 87," pp. 302–309, Lecture Notes in Computer Science, Vol. 278, Springer-Verlag, Berlin/New York.

MEINEL, CH. (1988), "Modified Branching Programs and their Computational Power," Habilitation thesis, Berlin 1988; Lecture Notes in Computer Science, Vol. 370, Springer-Verlag, Berlin/New York, 1989.

PUDLÁK, P., AND ŽAK, S. (1983), Space complexity of computation, preprint, University of Prague.

SAVAGE, J. E. (1984), The performance of multilective VLSI algorithms, *J. Comput. System Sci.* **29**, 243.

SIEGEL, A. (1984), "Tight Area Bounds and Provably AT^2 Bounds for Sorting Circuits," Report No. 122, Courrant Institute, NY.

ULLMAN, J. D. (1984), "Computational Aspects of VLSI," Comput. Sci. Press, Rockville, MD.