SIAM J. COMPUT.
Vol. 11, No. 4, November 1982

SOME AREA-TIME TRADEOFFS FOR VLSI*

RICHARD P. BRENT† AND LESLIE M. GOLDSCHLAGER‡

Abstract. Area-time bounds on VLSI circuits for context-free language recognition, for the evaluation of propositional calculus formulae and for set equality and disjointness questions, are considered. In all cases, a lower bound $AT^{2\alpha} = \Omega(n^{1+\alpha})$ is proved, where A is the chip area, T the execution time, and $0 \leqq \alpha \leqq 1$. Similar results were known for computations with $\Omega(n)$-bit outputs, but the computations considered here have only 1-bit outputs. Upper bounds are also discussed.

Key words. area-time tradeoffs, VLSI, formula evaluation, circuit-value problem, context-free language recognition, set equality, set disjointness, computational complexity, crossing sequences, models of computation

1. Introduction. Very large scale integrated circuit chips present an economic method of building general-purpose parallel machines as well as special-purpose chips which could be used as plug-in modules to conventional computers [9], [15], [17], [18], [21], [22].

The main complexity measures relevant to VLSI technology are the execution time of the algorithm ("parallel time") and the chip area required for the implementation. This chip area consists of both the active processing elements (transistors) and the wires used to interconnect them. If λ is the narrowest wire width allowed by the technology of the day, then chip area can usefully be measured in units of λ^2.

Because speed increases obtained through parallelism often require a large volume of information transfer between the active processing elements, it has frequently been observed that the resource needs of parallel algorithms are dominated by data communication. For VLSI circuits this phenomenon manifests itself in the chip area required for interconnections between active processing elements [15], [22], [23].

It is reasonable to expect that, as one increases the degree of parallelism used to solve a problem in order to decrease the execution time, the required inter-processor communication will increase. In a VLSI setting, we could therefore expect to find tradeoffs between time and chip area. Such tradeoffs have been reported for the discrete Fourier transform, matrix multiplication, Gaussian elimination, transitive closure, sorting and permutation problems, and for integer addition and multiplication [1], [5], [6], [20], [21], [22], [24], [25].

This paper gives area-time tradeoffs for context-free language recognition, finding the truth value of a formula given the values of its variables, determining whether two sets are equal (or disjoint), and determining if all elements of a set have distinct values. These problems differ from those mentioned above in that they have only a one-bit output. After our results were announced in [3], we learned that similar results were obtained independently by Lipton and Sedgewick [14].

Preliminary results and definitions are given in §2, lower bounds on area-time products are given in §3, and upper bounds are considered in §4. Finally, in §5, we mention some open problems.

* Received by the editors January 5, 1981 and in revised form January 15, 1982.

† Department of Computer Science, Australian National University, Canberra, Australia.

‡ Department of Computer Science, University of Queensland, St. Lucia, Australia. Now at Basser Department of Computer Science, Sydney University, Sydney, Australia.

2. Definitions and basic results.

DEFINITION 2.1. Let G denote a context-free grammar and $L(G)$ the language generated by G [11]. Then CFMEMBER $= \{\langle G, s\rangle | s \in L(G)\}$.

DEFINITION 2.2. Let F denote a formula of the propositional calculus in disjunctive normal form (DNF), and let ϕ denote a truth assignment to the variables of F. Then FVAL $= \{\langle F, \phi\rangle | F$ is true under $\phi\}$.

DEFINITION 2.3. A circuit α is a sequence $(\alpha_1, \cdots, \alpha_n)$ where each α_i is either a variable $x_1, x_2, \cdots$ or a gate AND (j, k), OR (j, k) or NOT (j) where $j \leqq k < i$. Let ϕ be a truth assignment to the variables of α, and ϕ' be the natural extension of ϕ to the gates of α [13]. Then the circuit value problem CVP $= \{\langle \alpha, \phi\rangle | \phi'(\alpha_n) = \text{true}\}$.

Let $X = (x_1, \cdots, x_n)$ and $Y = (y_1, \cdots, y_n)$ be sequences of integers in the range 0 to $n-1$ and $N = \{1, 2, \cdots, n\}$.

DEFINITION 2.4.

$$\text{DISTINCT} = \{\langle X\rangle | \text{ for all } i, j \in N,\ i \neq j \Rightarrow x_i \neq x_j\},$$

$$\text{DISJOINT} = \{\langle X, Y\rangle | \text{ for all } i, j \in N,\ x_i \neq y_j\},$$

$$\text{NONDISJOINT} = \{\langle X, Y\rangle | \text{ for some } i, j \in N,\ x_i = y_j\},$$

$$\text{EQUAL} = \{\langle X, Y\rangle | a \in X \text{ iff } a \in Y\}.$$

The VLSI model we use is that of [6]; see also [1], [2], [4], [5], [20], [21], [22], [23]. Essentially, the model is a collection of processing elements, some of which receive inputs or produce outputs, and which are connected by wires with finite width. For completeness we state the model here. Comments and justification are given following the statement of each assumption A1–A9. Our lower bound results are insensitive to minor details of the model.

A1. The computation is performed in a convex planar region R of area A.

Because of heat dissipation, packing, and testing requirements, a two-dimensional planar model is reasonable. The convexity assumption is not restrictive in the sense that almost all existing chips or useful modular designs do have convex boundaries for packaging or modularity reasons.

A2. Wires have minimal width $\lambda > 0$.

λ is assumed constant, but in applications of our results it will of course depend on the technology. We also assume R has width at least λ in every direction.

A3. At most $\nu \geqq 2$ wires can overlap (or intersect) at any point in R.

A chip may consist of ν layers. Wire crossings through different layers are allowed. In fact, transistors are typically formed by crossovers of wires. Since $\nu \geqq 2$, the graph of wires (edges) and gates (nodes) need not be planar in a graph-theoretic sense.

A4. I/O ports each contain a $\lambda \times \lambda$ square and thus have area at least $\rho \geqq \lambda^2$. An I/O port can be multiplexed to handle more than one input or output variable.

If R is a complete chip, ρ will be large compared to λ^2. If R is only part of a chip and I/O is to other regions on the chip, ρ could be of order λ^2. We do not require each input (or output) variable to appear on a distinct input (or output) port, as required in [21]. I/O ports may be multiplexed as they often are in practice.

A5. A bit requires minimal time $\tau > 0$ to propagate along a wire or to be transmitted through an I/O port. The time for one gate computation and an arbitrary fanout of the result is included in τ.

Since dimensions are limited by the minimal wire width λ and minimal gate area, a minimal propagation time is reasonable. We do not need to assume that the propagation time increases with the length of the wire. With the (small) sizes of chips we now have or anticipate, the propagation time, which is the time needed to charge or discharge a wire, is limited by the wire capacitance rather than the velocity of light. A longer wire will generally have a larger capacitance and thus require a larger driver to maintain constant propagation time, but the driver area need not exceed a fixed percentage of the wire area and so can be ignored if λ is increased slightly. A thorough discussion of this point may be found in [2]. Although it would be reasonable to assume bounded fanout, we do not need this assumption for proving lower bounds. When proving upper bounds in §4, we do assume bounded fanout.

A6. The times and locations at which input and output bits are available are fixed and independent of the values of the input bits.

In the terminology of [14], the input-output schedule is "where-oblivious" and "when-oblivious".

A7. Storage for one bit of information takes area at least $\beta > 0$.

β is typically several times larger than λ^2.

A8. Each input bit is available only once.

There is no free memory outside R. If the same input bit is required at different times, it must be stored within R, taking area at least β (see A7).

A9. Computation is synchronous with clock period at least τ.

This assumption, not required in [5], [6], simplifies the definition of "crossing sequence" given below.

DEFINITION 2.5. If V is any subset of the processing elements (i.e. gates and I/O ports), then a *bisection* of V is a cut of V into disjoint sets B and C such that, for some chord L perpendicular to a diameter D of the chip, the elements in B lie (at least in part) on one side of L and those in C lie (at least in part), on the other side of L.

As in [5], [6], we shall assume that processing elements can be shrunk to points and the chord L perturbed slightly so that L intersects only wires and not processing elements.

DEFINITION 2.6. Given a bisection as above, the (maximal) *information transfer* across the cut during a computation of time T is $I = WT/\tau$, where W is the number of wires which intersect L.

(Informally, I is the maximum number of bits which can be transferred in either direction between the processing elements in B and those in C. This definition is closer to those implicit in [5], [6] than those given in [1], [21], [22].)

DEFINITION 2.7. The *crossing sequence* associated with a given bisection and computation is the sequence of (binary) values on the wires crossing the cut (with some fixed ordering) at intervals of the clock period during the computation.

THEOREM 2.8. *Given a set V of processing elements, if there is a bisection of V with information transfer I, then*

$$AT^2 = \Omega(I^2).$$

Proof. Let L be the length of the chord associated with the bisection of V. Then, by Assumptions A1–A3, with W as in Definition 2.6, $A = \Omega(L^2)$ and $L \geqq \lambda W/\nu$, so $AT^2 = \Omega(W^2T^2) = \Omega(I^2)$.

(The proof is similar to that of Thompson [21], [22] except that, because of our Assumption A1 and Definition 2.6, we do not need to use the concept of "bisection width" or to minimize over a class of bisections. Our argument is used in the proof of [6, Thm. 3.1].)

Results similar to Theorem 2.8 have been used in the following way [1], [5], [6], [21], [22]: one considers the bisection of V into two regions B and C with a split of inputs $x_1, \cdots, x_n$ between them and an arbitrary distribution of outputs $y_1, \cdots, y_m$. Assuming that $|Y_C| \geqq |Y_B|$, the idea is to prove that there must be a sizeable information transfer I between X_B and Y_C. If one can show that I is proportional to n, then an area-time tradeoff of the form $AT^2 = \Omega(n^2)$ follows from Theorem 2.8.

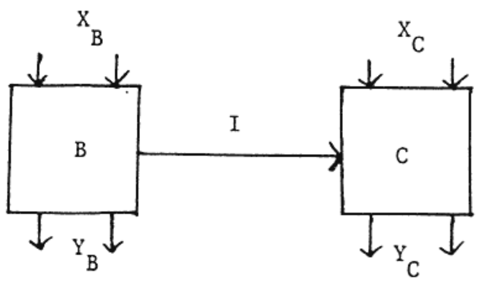

FIG. 2.1. *One-way information flow.*

For the problems considered in this paper, there is only one output, whose Boolean value answers the membership question posed by the instance of the problem which appears on the inputs. When the circuit produces only one output, the proof techniques required to bound the information transfer differ from the techniques used in [2], [5], [6], [20], [21], [22], [23], [24], [25]. In particular, it is necessary to consider the information transfer in both directions I_B and I_C (see Fig. 2.2). The total information

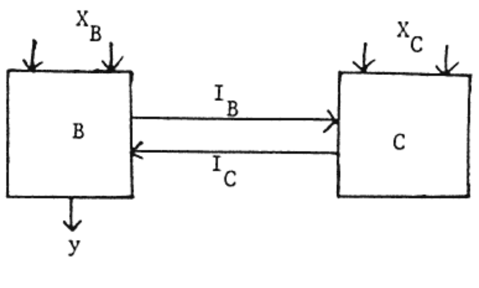

FIG. 2.2. *Two-way information flow.*

transfer $I = I_B + I_C$. Our technique may be regarded as a hybrid of the "crossing sequence" argument used to obtain lower bounds for Turing machine computations [10] and the "information flow" arguments previously used to obtain lower bounds on AT^2 for VLSI computations. All such techniques have the inherent limitation of yielding lower bounds on AT^2 no larger than cn^2, since all the inputs and outputs can be trivially redistributed with I proportional to n. This limitation results from ignoring the processing performed in B and C.

3. Lower bounds.

THEOREM 3.1. *For any* VLSI *circuit of area A which accepts as input a propositional calculus formula in disjunctive normal form with up to n literals and a set of truth values of its variables, and has as output the truth value of the formula in* (*worst-case*) *time T,*

$$AT^2 = \Omega(n^2).$$

Proof. Without loss of generality we assume that the formula has exactly n literals, $n/2$ variables, and that n is divisible by 8. Let M be the maximum number of input variables sharing (or multiplexing) one input port. If $M \geqq n/4$ the result is immediate, for $T = \Omega(M)$. Thus, we may assume that $M < n/4$. It follows that there is a bisection of the circuit into two parts B and C so that the input ports for between $n/8$ and $3n/8$ of the variables x_i are in B, and those for the remaining variables are in C. We denote the variables in B by $b_1, \cdots, b_k, \cdots$ and those in C by $c_1, \cdots, c_k, \cdots$, where $k = n/8$, and write $\mathbf{b} = (b_1, \cdots, b_k)$, $\mathbf{c} = (c_1, \cdots, c_k)$. The $2k$ variables not in $\mathbf{b}$ or $\mathbf{c}$ are denoted by $\mathbf{d} = (d_1, \cdots, d_{2k})$. Assume without loss of generality that the output port is in B.

Consider the formula

$$f(\mathbf{b}, \mathbf{c}) = \bigvee_{i=1}^{k} (b_i \& \bar{c}_i) \bigvee_{i=1}^{k} (\bar{b}_i \& c_i) \bigvee_{i=1}^{2k} (d_i \& \bar{d}_i)$$

which is in disjunctive normal form with $8k = n$ literals and $4k = n/2$ variables. (Intuitively, $f(\mathbf{b}, \mathbf{c})$ returns the truth value of $\mathbf{b} \neq \mathbf{c}$. By assumption A6, the bisection into B and C is independent of f.) Let I be the information transfer between B and C during a computation of $f(\mathbf{b}, \mathbf{c})$. Assume, by way of contradiction, that $I < k$. There are at most $2^I < 2^k$ crossing sequences (of bits across the cut), but there are 2^k possible values of $\mathbf{b}$, so there exist distinct $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ such that the computations of $f(\mathbf{b}^{(1)}, \mathbf{b}^{(1)})$ and $f(\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$ give identical crossing sequences. It follows that the circuit computes identical values for $f(\mathbf{b}^{(1)}, \mathbf{b}^{(1)})$ and $f(\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$, contrary to the definition of f. Thus $I \geqq k = \Omega(n)$, and the result follows from Theorem 2.8.

THEOREM 3.2. *With the assumptions of Theorem* 3.1, $A = \Omega(n)$.

Proof. Without loss of generality we may assume that there are $n/2$ variables and that n is divisible by 8. Suppose that there are P input ports. If $P \geqq n/4$ the result is immediate, for $A = \Omega(P)$. Thus, we may assume that $P < n/4$. At most P truth values can be read in simultaneously, so at some time during the computation, the truth values of a subset B of variables will have been read in, where $n/8 \leqq |B| \leqq 3n/8$. Let C be the complementary set of variables, and suppose $B = \{b_1, \cdots, b_k, \cdots\}$, $k = n/8$. Using the same function f as in proof of Theorem 3.1, we see that the output $f(\mathbf{b}, \mathbf{c})$ distinguishes between all 2^k possible vectors $\mathbf{b} = (b_1, \cdots, b_k)$ for suitable choice of $\mathbf{c} = (c_1, \cdots, c_k)$. Thus, the circuit must have at least 2^k internal states, i.e., it must store at least $k = \Omega(n)$ bits of information. The result now follows from Assumptions A7 and A8.

THEOREM 3.3. *With the assumptions of Theorem* 3.1,

$$AT^{2\alpha} = \Omega(n^{1+\alpha}) \quad \textit{for all } \alpha \in [0, 1].$$

Proof. The result follows by combining the bounds of Theorems 3.1 and 3.2 (as in the proof of [6, Thm. 3.3]).

COROLLARY 3.4. *For* VLSI *circuits which evaluate Boolean formulae in general (not necessarily disjunctive normal) form,*

$$AT^{2\alpha} = \Omega(n^{1+\alpha}) \quad \textit{for all } \alpha \in [0, 1].$$

COROLLARY 3.5. *For* VLSI *computation of the circuit value problem* CVP,

$$AT^{2\alpha} = \Omega(n^{1+\alpha}) \quad \textit{for all } \alpha \in [0, 1].$$

THEOREM 3.6. *Any* VLSI *circuit which computes* DISTINCT *has* $AT^{2\alpha} = \Omega(n^{1+\alpha})$ *for all* $\alpha \in [0, 1]$, *where n is the number of elements in the input sequence.*

Proof. Call an element of an input sequence a *block*, and note that each block consists of at least $\lceil \log_2 n \rceil$ bits.

We proceed much as in the proof of Theorem 3.1, taking $k = n/4$, and consider any bisection of the least significant input bits of each block into two parts B and C, $k \leqq |B| \leqq 3k$. Let $\mathbf{b} = (b_1, \cdots, b_k)$ denote a subset of the bits in B, and similarly for $\mathbf{c} = (c_1, \cdots, c_k)$. Now for each i, $1 \leqq i \leqq k$, if b_i or c_i is the least significant bit of block x_i, then assume that the other bits of x_i equal the binary representation of $i-1$. Let I be the information transfer between B and C. If $I < k$, then there exist distinct $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$ such that $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(1)}$ results in the same crossing sequence across the cut as $\mathbf{b} = \mathbf{b}^{(2)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(2)}$, where $\bar{\mathbf{b}}$ is $(\bar{b}_1, \cdots, \bar{b}_k)$. So the circuit computes the same output for $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(1)}$, as for $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(2)}$. But this is a contradiction, as in the former case all blocks are distinct, while in the latter there is at least one pair of blocks which have identical input values. Hence $I \geqq k$.

The remainder of the proof is similar to the proofs of Theorems 3.1–3.3.

THEOREM 3.7. *Any* VLSI *circuit which computes* DISJOINT (or NONDISJOINT) *has* $AT^{2\alpha} = \Omega(n^{1+\alpha})$ *for all* $\alpha \in [0, 1]$, *where n is the number of elements in the input sequences.*

Proof. As in the proof of Theorem 3.6, taking $k = n/8$, consider any bisection of the least significant input bits of each block into two parts B and C, $2k \leqq |B| \leqq 6k$. At least half of the least significant bits of X must be in one part (say B), and thus at least half of the least significant bits of Y must be in the other part (C). Let $\mathbf{b}$ and $\mathbf{c}$ (respectively) denote k-subsets of these bits, assume that the most significant bits of their corresponding blocks are as above, and that all remaining blocks in X equal $2k$ and in Y equal $2k+1$. Choose $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ as above. Now $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(1)}$ produce the same output as $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \bar{\mathbf{b}}^{(2)}$, but in the former case X and Y are disjoint, whereas in the latter they have some element in common.

THEOREM 3.8. *Any* VLSI *circuit which computes* EQUAL *has* $AT^{2\alpha} = \Omega(n^{1+\alpha})$ *for all* $\alpha \in [0, 1]$, *where n is the number of elements in the input sequences.*

Sketch of proof. The proof is similar to that of Theorem 3.7, but now $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \mathbf{b}^{(1)}$ produces the same crossing sequence as $\mathbf{b} = \mathbf{b}^{(2)}$ and $\mathbf{c} = \mathbf{b}^{(2)}$, and so the circuit produces the same output for $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \mathbf{b}^{(1)}$ as for $\mathbf{b} = \mathbf{b}^{(1)}$ and $\mathbf{c} = \mathbf{b}^{(2)}$.

Remark. The proof of Theorem 3.8 breaks down if we allow probabilistic algorithms which have a positive (albeit very small) probability of error [M. O. Rabin, personal communication, August 1981].

THEOREM 3.9. *Let* $m = O(n)$. *There is a context-free grammar G with the following property. Any* VLSI *circuit which, given a string* $s = s_1 s_2 \cdots s_m$, *determines whether*

$$\langle G, s \rangle \in \text{CFMEMBER},$$

has $AT^{2\alpha} = \Omega(n^{1+\alpha})$ *for all* $\alpha \in [0, 1]$.

Proof. Let G be the context-free grammar with start symbol S, nonterminal symbols D, E, F, terminal symbols 0, 1, \$, ¢, and productions

$$S \to EF¢E$$

$$F \to 0F0|1F1|¢E\$E$$

$$E \to ED¢|\varepsilon$$

$$D \to D0|D1|\varepsilon \qquad \text{(where } \varepsilon \text{ is the empty string).}$$

It is easy to see that G generates the language

$$L(G) = \{x_1¢ \cdots x_p¢\$y_1^R¢ \cdots y_q^R¢ \mid p > 0, q > 0, x_i = y_j \text{ for some } i \leqq p, j \leqq q\}$$

where each x_i, y_j is a (possibly empty) string of binary digits and y_j^R is the reverse of y_j. $L(G)$ is just a generalization of NONDISJOINT. Hence, a circuit which can determine if $\langle G, s\rangle \in$ CFMEMBER, i.e., if $s \in L(G)$, can also determine if $\langle X, Y\rangle \in$ NONDISJOINT provided $\langle X, Y\rangle$ is encoded as the string $x_1¢ \cdots x_n¢\$y_1^R¢ \cdots y_n^R¢$. Hence, the result follows from Theorem 3.7.

4. Upper bounds. The following theorem shows that, for several of the problems considered in §3, the exponent of n in the lower bound $AT^2 = \Omega(n^2)$ is the best possible.

THEOREM 4.1. *There is a* VLSI *circuit which solves the problems of Theorem* 3.1, 3.6, 3.7 *or* 3.8 *with*

$$A = O(n^2 \log^3 n)$$

and

$$T = O(\log n).$$

Proof. We shall consider the problem of Theorem 3.1 (evaluating propositional calculus formulae in disjunctive normal form with up to n literals); the problems of computing DISTINCT, DISJOINT and EQUAL may be dealt with in a similar fashion.

A literal is a variable or its negation, so any formula with at most n literals has at most n variables. We shall construct a circuit for which there are n input ports along the north edge, the ith such port giving the Boolean value of the variable x_i, for $i = 1, 2, \cdots, n$. A literal is of the form x_i or $\bar{x}_i$, so can be encoded by $\lceil \log_2 n\rceil + 1$ bits. Since the formula is in disjunctive normal form, it can be encoded simply as a string of literals separated by encoded operators "$\&$" and "V". The encoded formula will enter the west edge of our circuit through $O(n \log n)$ input ports. The overall layout is illustrated in Fig. 4.1.

To simplify the description we first assume unbounded fan-out. Consider the evaluation of a single literal $p_i = N_i x_i$, where N_i may be either the identity or negation operator. It is easy to construct a circuit, of area $O(n \log n)$ and delay $O(\log n)$, which has a grid of n lines running north to south (of which the ith is to be selected), and along the south edge a fan-in tree whose result may be complemented (depending on N_i) and fed out along the east edge. This is illustrated in Fig. 4.2.

It remains to specify an evaluation tree which inputs the value of each literal and the operators, and outputs the value of the formula. This is illustrated in Fig. 4.3, where the wires between each node transmit six bits $x\theta_x y\theta_y z\theta_z$, where x, y and z can be 0 or 1 and θ_x, θ_y and θ_z can be encodings of $\&$ or V.

Initially, the ith node of the evaluation tree carries the signal $1 \;\&\; 1 \;\&\; p_i\theta_i$, where p_i is the value of the ith literal and θ_i the operator to its right (irrelevant if $i = n$).

Each node of the evaluation tree takes two six-bit signals f_1 and f_2 representing partially evaluated formulae, and evaluates as much as possible of the concatenated formula f_1f_2. It is straightforward to verify by induction that each six-bit signal must have one of the forms

$$xVyVz\&,$$

$$1\&yVz\theta,$$

or

$$1\&1\&z\theta$$

where x, y, z are 0 or 1 and θ is $\&$ or V.

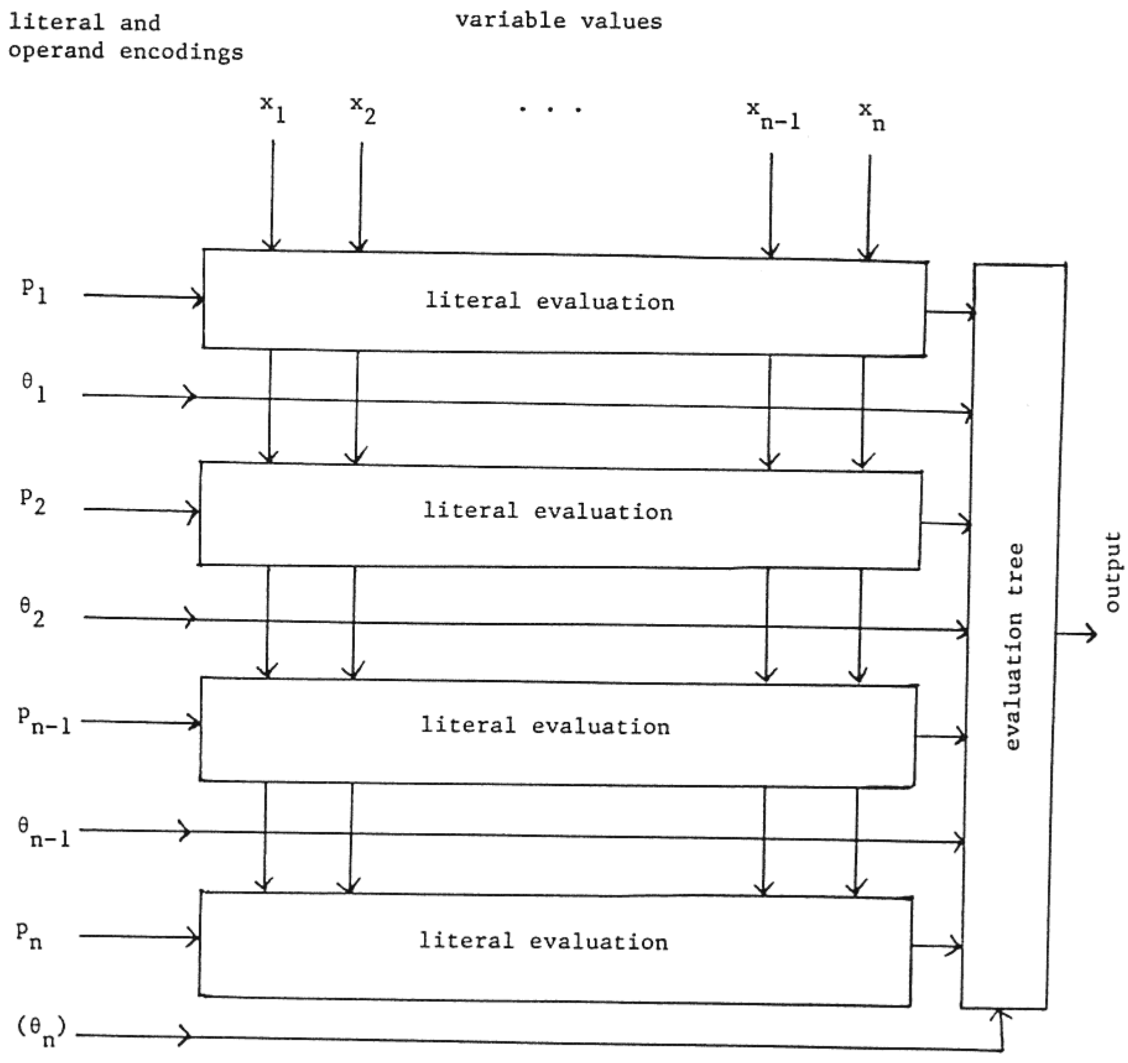

FIG. 4.1. *Overall layout.*

The root of the evaluation tree evaluates the final six-bit signal to give the values of the formula.

The circuit constructed above has width $O(n)$, height $O(n \log n)$, and delay $O(\log n)$. However, we have neglected the problem of fanout. In order to reduce the fanout of the input lines entering along the west edge (i.e., the literal encodings) we must separate each line by a distance $O(\log n)$ to accommodate fanout trees of width $O(n)$ and height $O(\log n)$ (see [4]). This makes the height of the complete circuit $O(n \log^2 n)$. A similar problem arises with the lines (carrying variable values) which enter along the north edge and must be fanned out to the sub-circuits evaluating each literal. We separate each line by a distance $O(\log n)$ to allow space for fanout trees. Thus, the circuit has been "stretched" by a factor $O(\log n)$ in both directions, giving it a width of $O(n \log n)$, height of $O(n \log^2 n)$, and area $O(n^2 \log^3 n)$. (A more detailed analysis might lower the exponent of $\log n$, but we are more interested in the exponent of n, which is optimal by the results of §3.)

Remark 4.2. The problems of evaluating DISTINCT, DISJOINT and EQUAL can clearly be solved rather easily if a sorting circuit is available. Hence, upper bounds on the area and time required for these problems may be obtained from the corresponding upper bounds for sorting networks. In particular, these problems can be solved with $AT^{2\alpha} = O(n^{1+\alpha} \log^c n)$ for some constant c. See, for example, Thompson [23].

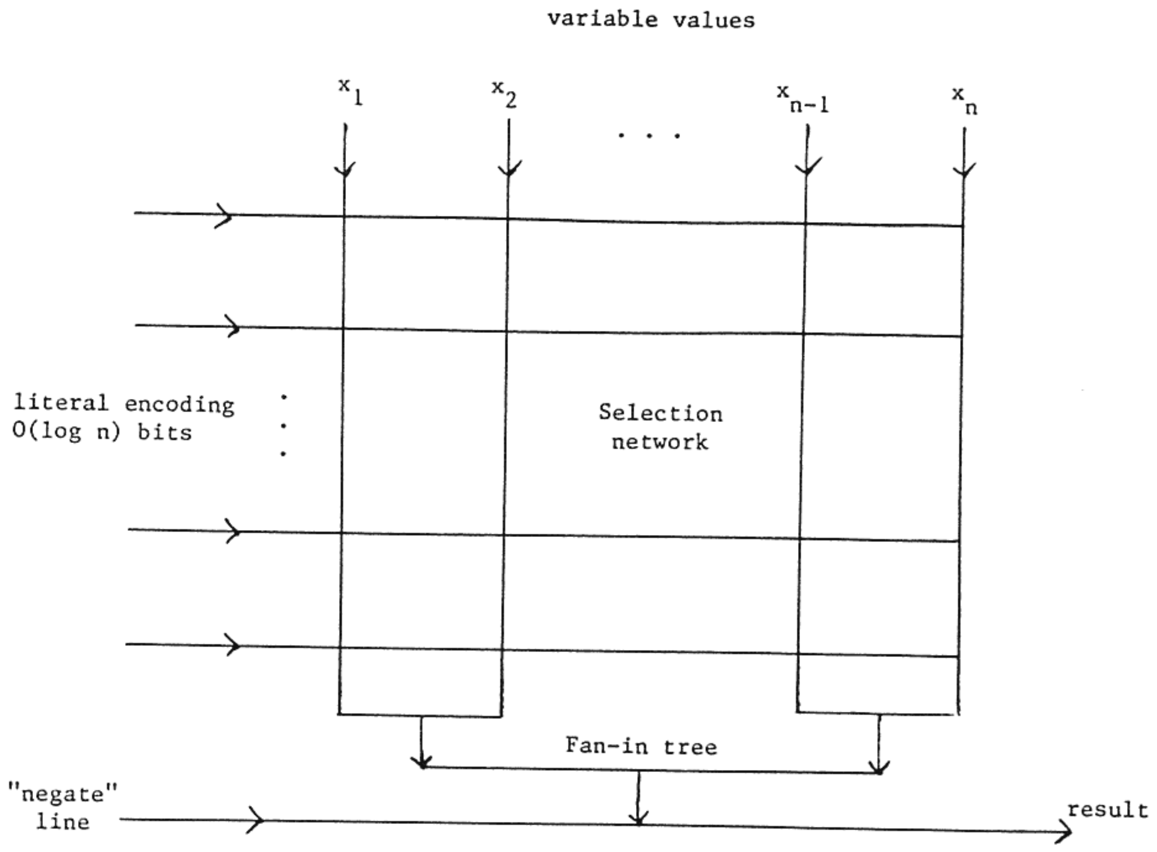

FIG. 4.2. *Literal evaluation (fan-out restrictions ignored).*

In [5], [6] it was shown that the exponent $1+\alpha$ in the lower bound $AT^{2\alpha}=\Omega(n^{1+\alpha})$ was sharp (for all $\alpha \in [0, 1]$) for the problem of binary multiplication of n-bit numbers. Except for the cases covered by Theorem 4.1 and Remark 4.2, and the trivial case $\alpha=0$, we do not know if the same is true for the problems considered in §3. In fact, this is unlikely for the context-free language recognition problem, as the best known serial algorithm uses n by n matrix multiplication [11], [16], [20]. Combining ideas of Preparata and Vuillemin [18] and Ruzzo [19], it may be shown that the context-free language problem can be solved by a VLSI circuit having $T=O(\log^2 n)$ and $A=O(n^c)$ for some constant c. (We believe that $c \leqq 8$, but have not yet worked out all the details. c can be reduced by the method of [12], at the expense of increasing T to $O(n)$.) Recognizing regular expressions appears to be a much easier problem [7], [8].

5. Conclusion. We have given lower bounds $AT^{2\alpha}=\Omega(n^{1+\alpha})$ for several natural recognition problems. The case $\alpha=\frac{1}{2}$ is interesting, as the area-time product AT can be viewed as the "rental cost" for a VLSI chip for the duration of the computation. On the other hand, for VLSI circuits which allow pipelining, AT overestimates the cost since portions of the chip can be re-used for subsequent computations before earlier computations are completed. In these cases the area alone (i.e., the case $\alpha=0$) may be a better measure of cost.

Possible areas for future research include:

a) Finding a technique to prove lower bounds better than $AT^{2\alpha}=\Omega(n^{1+\alpha})$.

b) Obtaining sharper upper bounds for problems of practical interest, e.g., the context-free language recognition problem.

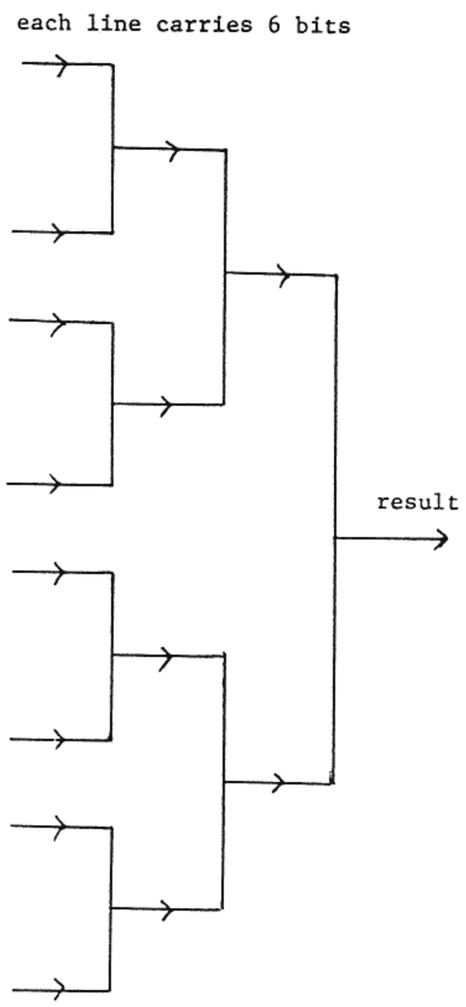

FIG. 4.3. *Final evaluation tree.*

c) Extending our results to probabilistic algorithms (see the remark following the proof of Theorem 3.8).

Acknowledgments. We are grateful to Al Borodin for suggesting the set equality and disjointness problems, and for fruitful discussions. We also thank the referees and Professor W. L. Ruzzo for their comments which helped us to sharpen Theorem 3.9 and to correct the proof of Theorem 4.1.

REFERENCES

[1] H. ABELSON AND P. ANDREAE, *Information transfer and area-time tradeoffs for* VLSI *multiplication*, Comm. ACM, 23 (1980), pp. 20–23.
[2] G. BILARDI, M. PRACCHI AND F. P. PREPARATA, *A critique and appraisal of* VLSI *models of computation*, VLSI Systems and Computations, H. T. Kung et al., eds., Computer Science Press, Rockville, MD, 1981, pp. 81–88.
[3] R. P. BRENT AND L. M. GOLDSCHLAGER, *Some area-time tradeoffs for* VLSI, Report 22, Dept. Computer Science, University of Queensland, Australia, August 1980.
[4] R. P. BRENT AND H. T. KUNG, *On the area of binary tree layouts*, Inform. Proc. Letters, 11 (1980), pp. 46–68.
[5] ——— *The chip complexity of binary arithmetic*, Proc. 12th Annual ACM Symposium on Theory of Computing, New York, April 1980, pp. 190–200.

[6] ——— *The area-time complexity of binary multiplication*, J. Assoc. Comput. Mach., 28 (1981), pp. 521–534.

[7] R. W. FLOYD AND J. D. ULLMAN, *The compilation of regular expressions into integrated circuits*, Proc. 21st Annual IEEE Symposium on Foundations of Computer Science, New York, 1980, pp. 260–269.

[8] M. J. FOSTER AND H. T. KUNG, *Recognize regular languages with programmable building blocks*, VLSI 81, J. P. Gray, ed., Academic Press, New York, 1981, pp. 75–84.

[9] L. J. GUIBAS, H. T. KUNG AND C. D. THOMPSON, *Direct* VLSI *implementation of combinatorial algorithms*, Proc. Conference on VLSI: Architecture, Design, Fabrication, California Inst. of Technology, Jan. 1979.

[10] F. C. HENNIE, *On-line Turing machine computations*, IEEE Trans. Electronic Computers, EC-15 (1966), pp. 35–44.

[11] J. E. HOPCROFT AND J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[12] S. R. KOSARAJU, *Speed of recognition of context-free languages by array automata*, this Journal, 4 (1975), pp. 331–340.

[13] R. E. LADNER, *The circuit value problem is log space complete for* P, SIGACT News, 7, 1, Jan 1975, pp. 18–20.

[14] R. J. LIPTON AND R. SEDGEWICK, *Lower bounds for* VLSI, Proc. 13th Annual ACM Symposium on Theory of Computing, New York, 1981, pp. 300–307.

[15] C. A. MEAD AND L. C. CONWAY, *Introduction to* VLSI *Systems*, Addison-Wesley, Reading, MA, 1980.

[16] V. YA. PAN, *New fast algorithms for matrix operations*, this Journal, 9 (1980), pp. 321–342.

[17] F. P. PREPARATA AND J. E. VUILLEMIN, *The cube-connected-cycles: a versatile network*, Proc. 20th IEEE Symposium on Foundations of Computer Science, New York, Oct. 1979, pp. 140–147.

[18] ——— *Area-time optimal* VLSI *networks for multiplying matrices*, Inform. Proc. Letters, 11 (1980), pp. 77–80.

[19] W. L. RUZZO, *On uniform circuit complexity*, Proc. 20th IEEE Symposium on Foundations of Computer Science, Oct. 1979, pp. 312–318.

[20] J. E. SAVAGE, *Area-time tradeoffs for matrix multiplication and related problems in* VLSI *models*, J. Comput. System Sci., 22 (1981), pp. 230–242.

[21] C. D. THOMPSON, *Area-time complexity for* VLSI, Proc. 11th Annual ACM Symposium on Theory of Computing, New York, April, 1979, pp. 81–88.

[22] ——— *A complexity theory for* VLSI, Report TR-CS-80-140 (Ph.D. thesis), Dept. Computer Science, Carnegie-Mellon University, Pittsburgh, August 1980.

[23] ——— *The* VLSI *complexity of sorting*, in VLSI Systems and Computations, H. T. Kung et al., eds., Computer Science Press, Rockville, MD, 1981, pp. 108–118.

[24] J. E. VUILLEMIN, *A combinatorial limit to the computing power of* VLSI *circuits*, Proc. 21st Annual IEEE Symposium on Foundations of Computer Science, New York, 1980, pp. 294–300.

[25] A. C. YAO, *The entropic limits on* VLSI *computations*, Proc. 13th Annual ACM Symposium on the Theory of Computing, New York, 1981, pp. 308–311.