A Model of Computation for VLSI with Related Complexity Results

BERNARD CHAZELLE AND LOUIS MONIER

Carnegie-Mellon University, Pittsburgh, Pennsylvania

Abstract. A new model of computation for VLSI, based on the assumption that time for propagating information is at least linear in the distance, is proposed. While accommodating for basic laws of physics, the model is designed to be general and technology independent. Thus, from a complexity viewpoint, it is especially suited for deriving lower bounds and trade-offs. New results for a number of problems, including fan-in, transitive functions, matrix multiplication, and sorting are presented. As regards upper bounds, it must be noted that, because of communication costs, the model clearly favors regular and pipelined architectures (e.g., systolic arrays).

Categories and Subject Descriptors: B.7.1 **[Integrated Circuits]**: Types and Design Styles—*VLSI* (*very-large-scale integration*)

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Chip complexity, lower bounds

1. *Introduction*

The importance of having general models of computation for very-large-scale integration (VLSI) is apparent for various reasons, chief of which are the need for evaluating and comparing circuit performance, establishing lower bounds and trade-offs on area, time, and energy, and, more generally, building a complexity theory of VLSI computation.

Although models must be simple and general enough to allow for mathematical analysis, they must also reflect reality regardless of the size of the circuit. We justify the latter claim by observing that if today's circuits are still relatively small, the use of high-level design languages combined with larger integration, bigger chips, and improved yield may make asymptotic analysis quite relevant in the near future.

As circuits are pushed to their limits, however, constraints that could be ignored before become major problems and must be accounted for in the models. For example, basic laws of physics show that the propagation of information takes time at least proportional to the distance. Although this limitation is not effective for chips of reasonable size (see the analysis of [3]), it shows that any model assuming faster propagation delays will break down in the limits presumed by the very notion of *asymptotic* analysis.

This research was supported in part by the Defense Advanced Research Project Agency under contract F33615-78-C-1551, monitored by the Air Force Office of Scientific Research.

Authors' addresses: B. Chazelle, Department of Computer Science, Brown University, Box 1910, Providence, RI 02912; L. Monier, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304.

Our purpose in this paper is to propose a model of computation based on the linear-delay propagation assumption. This model makes claim of realism only in the asymptotic sense. Paradoxically, it *does not* conflict with previous models based on different propagation delays. We do not question the fact that with some technologies, and for some reasonable range of input size, more slowly growing delay functions might provide a more accurate model of reality. It cannot be denied, however, that such models must break down asymptotically. Whether the threshold of breakdown is completely beyond the present VLSI horizon or is within reach is a question of great interest, but irrelevant to our discussion. In particular, whether or not the results of this paper will be of any use to today's chip designer is deliberately left unaddressed.

Instead, we take the theoretical approach of studying intrinsic limitations of physical computing devices. The lower bounds that we present in this paper are physical barriers in the asymptotic sense. They might be effectively circumvented on a small scale (and the practical importance of doing so amply justifies research on different VLSI models), but what is appealing, at least theoretically, is that in the limit these barriers cannot be overcome. Naturally, our approach in this paper will lead us to lower bounds and trade-offs rather than to upper bounds, since interest in the latter tends to be more dependent on practical reality.

Our main results include lower bounds on the complexity of fan-in, addition, cyclic shift, integer multiplication, convolution, linear transform, product of matrices, and sorting. In the last section, we illustrate how additional assumptions may be needed for building technology-dependent models. We have chosen the NMOS technology as an example of power considerations leading to more stringent requirements. We show that it is then possible to derive stronger lower bounds on the complexity of some problems.

2. *The Model*

Our model is referred to as the *iterative model*, to borrow terminology from cellular automata theory. It is for the most part a refined version of the current planar models found in the literature [4, 18, 20]. These models, which are all in fact very similar, have been used to establish lower bounds and make comparative analyses of circuits. Aside from the possibility of parallel processing, these models differ from the traditional RAM model by charging area costs for the transmission of information. In addition, the iterative model will charge time costs proportional to the length of the wires carrying information. As a result, a circuit appears as a fully geometric object rather than a purely combinatorial one.

We insist upon the fact that we are interested in asymptotic results, that is, we consider circuits much larger than the ones built today. The ratio between the size of a circuit and a minimum-size device (transistor or wire) is today around 1000. We believe that in the near future the use of wafer-scale integration and sophisticated packaging, associated with better lithography technologies, will bring this ratio much higher. Circuits will be much more complex, and the notion of asymptotic analysis will be more meaningful than it is today.

2.1 MODEL OF COMPUTATIONAL DEVICE.

We can think of a computational device as a *black box* which computes a Boolean function $(y_1, y_2, \ldots) = F(x_1, x_2, \ldots)$. Information is treated digitally and is communicated to the device through I/O ports in a fixed format. With each variable x_i (respectively, y_i) there is associated both an input (respectively, output) port and a chronological rank on the set of inputs (respectively, outputs). In addition, to each variable there must correspond information to tell when it is available on its port. This information may be independent of the inputs, which involves fixing the times and locations at which the I/O bits can be used [20]. On the other hand, the information may be provided by the circuit (output) or the user (input), thus allowing for self-timed computations [11, 16]. We may wish to make several copies of the input available to the circuit, thus assuming that duplicates correspond to distinct x_i's.

Internally, a computational device is a circuit connecting nodes and wires into a directed graph and is defined by a geometrical layout of this graph. The layout is supposed to be planar, meaning that all the nodes are on the same plane and that the wires are allowed to lie on a constant number of parallel layers. This implies that there are at most a constant number of crossovers at any point.

Wires may intersect only at the nodes, and their width must exceed a minimum value λ. Similarly, all nodes occupy a fixed area. We distinguish I/O nodes (ports), where input and output values are available from the logical nodes (gates) that compute Boolean functions. The circuit is laid out within a convex region with all the I/O nodes lying on its boundary. We assume that inputs can be multiplexed, that is, that input ports can be used several times. However, we do not allow free duplication of inputs. We also assume that the times and locations of the I/O bits are fixed and independent of the values of the inputs. See [7], [9], and [10] for discussion of these problems.

2.2 Model of Computation. Two parameters characterize the computational complexity of a circuit: its *area A* and its *time of computation T*. For a fixed value of the inputs, T measures the time between the appearances of the first input bit and the last output bit. The maximum value of T for all possible inputs defines the time complexity of the circuit. In this paper the time T of a circuit always refers to this worst-case measure. Other approaches, involving average or even best time of computation, can be considered. Of course, this is pertinent only with self-timed circuits.

Another important parameter is the *period* of a circuit [20]. Since it is possible to pipeline the computations on several sets of inputs, we define the period P as the minimal time interval separating two input sets. More precisely, if $(a_1, \ldots, a_N)$ and $(b_1, \ldots, b_N)$ are two sets of inputs, and if the time separating the appearance of a_i and b_i is the same for all i, this interval defines the period P.

We next turn to the actual computation of a problem. It can be viewed as the propagation and logical treatment of information bits across nodes and wires. Viewing the circuit as a directed graph, we associate with each node (including I/O ports) a Boolean variable $I_i(t)$ (respectively, $O_j(t)$) for each incoming (respectively, outgoing) wire, which is defined at all times t. These variables represent the information available at the entrance and exit points of a node. In addition, there corresponds to each node a state $S(t)$ chosen from among a finite number of possible states. Then each node of the circuit computes a function F of the form

$$[S(t+\tau), \ldots, O_j(t+\tau), \ldots] = F[S(t), \ldots, I_i(t), \ldots]. \tag{1}$$

Informally, this relation means that a node can produce a result only a delay τ after its inputs have been available.

The most drastic departure from previous models, however, comes from the next assumption, which expresses that the time of propagation across a wire is at least proportional to the length of the wire. Let $I(t)$ and $O(t)$ be the variables associated with the ends of a wire of length L. We require that

$$I(t+T) = O(t) \qquad \text{for} \quad T = \Omega(L). \tag{2}$$

The last assumption asserts the bandwidth limitation of wires. Although we do not wish to restrict the storage capacity of the wire to 1 bit, we assume that a wire can carry a number of bits at most proportional to its length. The simplest way to express this is to regard a wire of length L as a sequence of L subwires of unit length connected by nodes computing the identity function. Then for each subwire we have $I(t+1) = O(t)$, meaning that each subwire stores exactly 1 bit of information and has a unit delay. Note that assumption (2) follows directly from this approach.

We finally introduce the important concept of *datapath.* We say that there exists a datapath from an input variable x to a node V if there is a directed path to V from the input port where x is read in. Moreover, we require that information be propagated from x to V before the circuit completes its computation. Note that a datapath involves not only a physical connection between two points, but also the possibility that information may be transmitted from one point to another within the duration of the computation.

2.3 PHYSICAL JUSTIFICATION OF THE ASSUMPTIONS. Like previous models, this model strongly reflects present technologies, especially electrical ones (NMOS, CMOS, TTL); for example, a circuit is taken to be planar, convex, with its I/O ports on the boundary, and we assume minimal dimensions for every part (node or wire). All these constraints find justifications in today's fabrication and packaging processes. Other stronger reasons can be advanced:

—*Planarity*: This is indeed a matter of choice, since *three-dimensional* computing devices are conceivable [14]. Although most of present-day circuits are planar, it would be very interesting to explore the features and properties of a three-dimensional model. We are still inclined to think that because of heat dissipation problems, circuit designers will be unlikely to give up planarity. Indeed, today's circuits all use the third dimension for cooling purposes.

—*Convexity and I/O ports*: We believe that circuits should be easy to manipulate and connect together. A good way to achieve handiness is to make circuits into closed systems, where interactions with the outside occur only through the boundary. Convexity thus appears as a natural requirement.

—*Minimal size*: This seems to be an intrinsic feature of any *physical* device (quantum mechanics argument). As a consequence, gates and wires may not be arbitrarily small, if they are to be material.

—*Propagation delays*: The ultimate justification for our assumption comes from a speed-of-light argument. No information can propagate faster than the light. Moreover, in practice, parasitic effects reduce the speed of propagation several orders of magnitude below that limit.

We can illustrate the latter point by briefly examining the delays incurred in MOS technologies. The information is coded as a potential, and its propagation involves loading the capacitance of a wire. Since a wire is a piece of conducting material, it has a nonnull resistance and capacitance, and because of current process conditions, both are proportional to the wire length. Detailed analysis of this situation can be found in [5]. On today's small circuits, the capacitance of wires is the main limiting factor, and well-known techniques (increasing the size of drivers) can reduce the delay to a constant value, close to the switching time of a gate. This is possible, however, because the wires are limited to a few millimeters in length and because we consider only metal wires, a constraint that may not always be satisfied.

For the future, we believe that use of wafer-scale integration or the packaging of many chips on the same substrate will force us to consider much larger circuits. Then wires will be much longer, and owing to current density limitations it will no longer be possible to drive them in constant time. Also, because of the diffusion law, delays on wires are in fact proportional to the *square* of the length, and on long wires the quadratic term may become dominant. In this technology, therefore, it appears necessary to decompose long wires into pieces of constant length connected by nodes (globally computing the identity function) in order to achieve delays proportional to the length of the wires. Moreover, the maximum speed of information propagation is largely dominated by the speed of light [16], mainly because the current intensities involved are very low and many electrical phenomena (overheat, metal migration [6]) impose a limit on them.

3. *Distributing and Collecting Information*

Fanning in and fanning out information are two of the most common operations performed by circuits; thus we first turn to these problems, from which we can best measure the significant departure of our model from the previous ones. The results will be basic tools in analyzing further problems. We need the following geometric lemma.

LEMMA 1. *If P is an arbitrary convex polygon with a perimeter N, the maximum distance between any vertex of P and an arbitrary point in the plane is $\Omega(N)$.*

We omit the proof, which is straightforward. As a result, it takes $\Omega(N)$ time to propagate a bit from any point M to N points on a convex boundary (e.g., input or output ports).

3.1 FAN-OUT. A fan-out of degree N refers to the distribution of N copies of an information bit at N different locations on the circuit. We have the following.

THEOREM 2. *It takes time $T = \Omega(N^{1/2})$ to perform a fan-out of degree N.*

PROOF. A consequence of the fact that the maximum distance between a point and N arbitrary points in the plane is at least $N^{1/2}$. □

3.2 FAN-IN. The fan-in is essentially the reverse operation of the fan-out, as N information bits must now converge from several sources to one destination point. Yet it is a little more general, since the information may be submitted to logical operations on the way to its destination. Typically, the problem is to compute a Boolean function of N inputs and one output. However, to ensure that all the input bits are used in the computation, we give the following definition of a fan-in.

Definition 3. A Boolean function $y = f(x_1, \ldots, x_N)$ is a fan-in of degree N if there is an assignment of the N variables such that for any $i, f(a_1, \ldots, a_i, \ldots, a_N) \neq f(a_1, \ldots, \neg a_i, \ldots, a_N)$. The N-tuple $(a_1, \ldots, a_N)$ of bits is called a *hard input* of the function f.

The reader can check that OR-ing or AND-ing N bits, as well as computing the last carry of the sum of two N-bit integers, involves a fan-in of degree N.

If the N inputs are valid at the same time, we have results similar to the fan-out since there must be a datapath from any input port to the point where the value of the function appears. In the more general case where pipelining is allowed and the inputs are valid at arbitrary times, we can show the following.

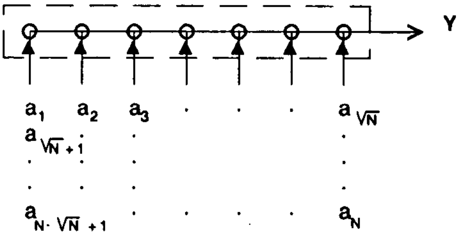

FIG. 1. Computing $Y = a_1$ op $a_2 \cdots$ op a_N takes $\Omega(N^{1/2})$ time and area.

THEOREM 4. *If T (respectively, A) denotes the minimum time (respectively, area) for performing a fan-in of N inputs, we have* $T = \Omega(N^{1/2})$ *and* $AT = \Omega(N)$.

PROOF. Let p denote the total number of input ports actually used. For obvious reasons, all the bits of a hard input have to be read in, which takes $\Omega(N/p)$ time. Also, there must be a datapath from any input variable to the point where the value of the function is available. Then, with the input ports lying on a convex boundary, Lemma 1 shows that $T = \Omega(p)$. Observing that $A = \Omega(p)$, the result is then immediate. □

Note that to perform a fan-in on a hard input always takes $\Omega(N^{1/2})$ time. We also observe that the above lower bounds are still valid for Boolean functions with an arbitrary number of outputs as long as at least one output is a fan-in of all the input values. Addition, for example, falls in that category, since the last carry depends on all the operand bits. If the Boolean function is a commutative, associative operation on N variables, these lower bounds are tight with today's circuits, as shown in Figure 1.

4. *Addition*

Since the iterative model relates the time of computation to the geometry rather than to the topology of the circuit, we can show that many complete binary-tree-based schemes cease to have the logarithmic time complexity that they enjoyed in previous models. Notable examples include the fan-in and fan-out operations studied earlier, or the addition of two N-bit integers, to which we next turn our attention. Our results are expressed in the following.

THEOREM 5. *If T is the time, P the period, and A the area required by any circuit to add two N-bit integers, we have*

$$T = \Omega(N^{1/2}), \qquad AT = \Omega(N), \qquad ATP = \Omega(N^2).$$

The first two results come from the computation of the last carry, which can be easily shown to involve a fan-in of degree N. The proof for the last lower bound is more difficult, and requires a few technical lemmas.

To simplify the notation, we equate all constant factors with 1 when this does not compromise the reasoning. Let $Y = y_{N+1}y_N \cdots y_0$ be the result of the addition of the two $N+$ 1-bit integers $X = x_N \cdots x_0$ and $111 \cdots 1_2$. Since all the bits of X are not necessarily read at the same time, let $t_1, \ldots, t_p$ be the instants when bits of X are read in. If Z_i denotes the set of X's bit input at time t_i, we can partition the bits of X into p subsets $Z_1, \ldots, Z_p$. Note that the Z_i's may not form subsequences of X since bits may not need to be read in order. Turning now to the distribution of $X_i = x_i \cdots x_0$ among the sets $Z_1 \cdots Z_p$, we call t_{s_i} the time when all the bits of X_i have finally been read into the circuit. Clearly, $X_i \subset Z_1 \cup \cdots \cup Z_{s_i}$. For each j,

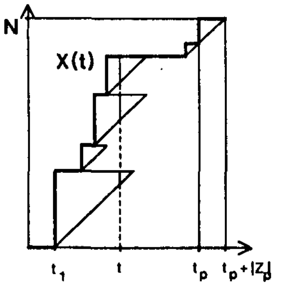

FIG. 2. The lower bound on ATP for the addition.

$1 \leq j \leq s_i$, we define $L_j = |X_i \cap Z_j|$, that is the number of bits in $\{x_i, \ldots, x_0\}$ read at time t_j. The sum of the L_j's satisfies $L_1 + \cdots + L_{s_i} = |X_i| = i + 1$. We can now establish a lower bound on the time required for computing y_i.

LEMMA 6. *For any i, $0 \leq i \leq N$, y_i cannot be computed before time $t = max\{L_1 + t_1, \ldots, L_{s_i} + t_{s_i}\}$.*

PROOF. Recall that for simplicity, all constants are taken to be 1. We observe that the function $y_i = f_i(x_i, \ldots, x_0)$ is a fan-in with $(0, 0, \ldots, 0)$ as a hard input. Therefore for any j, $1 \leq j \leq s_i$, a fan-in must be performed on the L_j bits of $X_i \cap Z_j$ before $f_i(0, 0, \ldots, 0)$ can be evaluated. Since these L_j bits are all read at time t_j, y_i cannot be computed before time $L_j + t_j$, which completes the proof. It is crucial to observe that the result is true for all i because the input $X = 0$ is hard for all the functions f_i. □

Let $Y(t)$ designate the number of Y's bits output by time t. Similarly, $X(t)$ denotes the number of X's bits already read at time t. By definition we have $X(t_i) = |Z_1| + \cdots + |Z_i|$. We use the previous result to evaluate the growth of the function $Y(t)$.

LEMMA 7. *For any i, $1 \leq i \leq N$, and for any t, $t_i \leq t < t_{i+1}$, we have $Y(t) \leq \sum_{1 \leq j \leq i} min(|Z_j|, t - t_j)$.*

PROOF. Let y_m be the highest order bit output by time t. We clearly have

$$m \geq Y(t) - 1. \tag{3}$$

From Lemma 6, it follows that for all j, $1 \leq j \leq s_m$, we have $L_j \leq t - t_j$. Since we also have $L_j \leq |Z_j|$, we derive $m + 1 = \sum_{1 \leq j \leq s_m} L_j \leq \sum_{1 \leq j \leq s_m} \min(|Z_j|, t - t_j)$ and

$$m + 1 \leq \sum_{1 \leq j \leq i} \min(|Z_j|, t - t_j), \tag{4}$$

since at time t, all the bits $x_1, \ldots, x_m$ have been already read, and thus $t_{s_m} \leq t_i \leq t$. Combining (3) and (4) proves the claimed result. □

Theorem 5 is proved as follows. Figure 2 gives a geometric interpretation of Lemma 7. $Y(t)$ is bounded by the total length of the dashed lines. It follows from this interpretation that the quantity $\int_{t_1}^{t_p+|Z_p|} [X(t) - Y(t)]\,dt$ dominates the total area of the p triangles. Actually, since a fan-in on all the bits of Z_p must be performed in order to compute y_N, the total time of computation T is at least $t_p + |Z_p|$. Therefore, setting t_1 to 0, we have

$$\int_0^T [X(t) - Y(t)]\,dt \geq |Z_1|^2 + \cdots + |Z_p|^2,$$

and, since the minimum of the right-hand side is achieved for all the $|Z_i|$ equal, the relation $\sum_i |Z_i| = N + 1$ implies

$$\int_0^T [X(t) - Y(t)]\, dt \geq \frac{N^2}{p}. \tag{5}$$

At this point, it would be straightforward to derive the relation $AT^2 = \Omega(N^2)$. We can, however, improve upon this bound by using a general technique introduced by Baudet [1]. Let P be the period of the circuit. Assuming that problems are treated in a pipeline fashion, let $P_1, \ldots, P_m$ be the problems still in progress at time t. We assume this number m of problems to be the same at all times. By definition, at time t, P_j is in the same stage as P_1 was at time $t - (j - 1)P$. Therefore, at this time, at least $\sum_{1\leq j\leq m}[N - Y(t - (j - 1)P)]$ output bits (respectively, $\sum_{1\leq j\leq m}(N - X(t - (j - 1)P))$ input bits) have yet to be produced (respectively, read) with regard to problems $P_1, \ldots, P_m$. Let A be the area of the circuit. Since at most A bits can be stored at any time, a simple counting argument involving the states of the circuit shows that

$$A + \sum_{1\leq j\leq m} [N - X(t - (j - 1)P)] \geq \sum_{1\leq j\leq m} [N - Y(t - (j - 1)P)];$$

hence

$$A \geq \sum_{1\leq j\leq m} [X(t - (j - 1)P) - Y(t - (j - 1)P)]$$

or

$$A \geq \sum_{1\leq j\leq m} [X(t + (j - 1)P) - Y(t + (j - 1)P)].$$

Integrating over t from 0 to P leads to

$$AP \geq \sum_{1\leq j\leq m} \int_0^P [X(t + (j - 1)P) - Y(t + (j - 1)P]\, dt,$$

that is, $AP \geq \int_0^T [X(t) - Y(t)]\, dt$. From (5) we derive the inequality $APp \geq N^2$, and since $T \geq p$, we conclude $APT = \Omega(N^2)$, which completes the proof of Theorem 5. □

5. *Transitive Functions*

In a recent paper [20], Vuillemin has shown that the transitivity of a function has heavy consequences on its complexity in a VLSI model. The function $(z_1, \ldots, z_N) = F(x_1, \ldots, x_N, s_1, \ldots, s_b)$ is transitive of degree N if, by assigning special values to the variables $s_1, \ldots, s_b$, the output is simply a permutation of the variables $x_1, \ldots, x_N$. This is to say that a function is transitive of degree N if it computes a transitive group of permutations acting on N elements. We also require that the set of all possible permutations thus obtained form a transitive group; that is, any x_i can be mapped onto any output bit z_j. Among well-known problems that involve computing transitive functions, we can list:

—cyclic shift on N bits;

—product of two p-bit integers, $N = 2p$;

—convolution of two p-element vectors, $N = (2p + 1)(2k + \log_2 p)$;

—linear transform of a p-element vector, $N = p(2k + \log_2 p)$ (where transform coefficients are counted as inputs);

—product of three $p \times p$ matrices, $N = p^2(2k + \log_2 p)$.

When necessary, we assume that all numbers are presented on k bits, with $k \geq \log_2 p$.

Using the geometric nature of our model, we can improve upon Vuillemin's results.

THEOREM 8. *Computing a transitive function of degree N takes time $T = \Omega(N^{1/2})$ and requires an area $A = \Omega(N)$.*

PROOF. The result on the area has already been shown by Vuillemin [20]. Let p be the number of output ports actually used. Consider one of the input variables being permuted. Since it can be mapped onto any output position and its input port P is fixed, there are possible datapaths from P to p distinct points on a convex boundary. Then Lemma 1 shows that at least one of them has length $\Omega(p)$; hence $T = \Omega(p)$. On the other hand, observing that it takes time at least proportional to N/p to output the result completes the proof. □

It is worthwhile noting the serious gap existing between this model and the previous ones, in which a transitive function could be computed in logarithmic time (e.g., the CCC-scheme [12, 13] or other recursive schemes [2, 19]). In fact, our model rules out any logarithmic time circuit. However, good performances on the period can be expected from pipelining the computation. The well-known trade-off $AT^{2\alpha} = \Omega(N^{1+\alpha})$, valid for $0 \leq \alpha \leq 1$, remains unchanged, although less leeway is now given on the time component.

6. *Sorting*

Although this problem is not transitive, we can prove similar bounds on A and T. Note that Thompson [18] and Savage [15] have already established bounds on AT^2 similar to those known for transitive functions.

6.1 THE MINIMAL BISECTION ARGUMENT REVISITED. We can improve on the general technique of *minimal bisection* [17] by introducing geometric arguments. Consider a line L partitioning the circuit into two parts C_1 and C_2, each producing half the output bits, or as close to half as possible given the fact that multiplexing may prevent a perfect dichotomy. We define the minimal cross-flow I to be the minimal number of bits crossing L. More precisely, let U_1 (respectively, V_1) and U_2 (respectively, V_2) denote the set of input (respectively, output) variables assigned in C_1 and C_2, respectively. For any fixed assignment of the inputs U_2, consider the total number of output sets V_2 obtained for all possible inputs U_1; we define Z_1 as its maximum value over all possible sets U_2. Similarly, we can define Z_2 by inverting the indices, and we call Z the maximum of Z_1 and Z_2. Finally, I is defined independently of the circuit as the minimum value of $\log_2 Z$ over all possible bisections and all possible circuits computing the function. Without loss of generality assume that $Z = Z_1$. It is clear that I bits must cross the separating line L from C_1 to C_2, and, moreover, to each of these bits there corresponds a datapath going from U_1 to V_2. We can prove the following result.

LEMMA 9. *Any circuit computing a function with minimum cross-flow I requires time $\Omega(I^{1/2})$.*

PROOF. Using the above notation, we choose L to be perpendicular to a diameter of the circuit, and we call ω the number of wires used by the I bits to cross L. Consider the wire closest to the middle of the chord L. Since it must carry a "cross-flow" bit, there exists a datapath from an input port of C_1 to an output port of C_2 using this wire, and an elementary geometric argument based on the convexity of the circuit shows that its length is $\Omega(\omega)$. It follows that the time T is $\Omega(\omega)$. Finally, since I bits crossing ω wires require time I/ω, we have $T = \Omega(\omega + I/\omega) = \Omega(I^{1/2})$. □

Since the minimum cross-flow of a transitive function of degree N is proportional to N [20], Lemma 9 gives a new proof of Theorem 8.

6.2 SORTING. The problem is now to sort N numbers, each of them being represented on k bits. We assume that $k \geq 2\log_2 N$, implying in particular that all the numbers *can* be different, and that all the bits of a number are input through the same port.

THEOREM 10. *Any circuit sorting N k-bit numbers, with $k \geq 2\log_2 N$, has an area $A = \Omega(N)$ and takes time $T = \Omega(Nk)^{1/2}$.*

PROOF. To prove the result on the area, we show that the circuit must be able to realize any permutation of the N lowest order bits. It suffices to observe that, if the N numbers deprived of their lowest order bit can take N distinct values, any permutation on these N values will induce a permutation on the N lowest order bits. Clearly, the condition on N and k ensures this property, which shows that sorting N numbers involves computing a transitive function of degree N, and establishes the result.

We will use the result of Lemma 9 to prove the result on the time. To do so, we must evaluate the minimum cross-flow I associated with sorting. For simplicity, we first assume that it is possible to bisect the circuit by a line into two parts C_1 and C_2, so that each part will produce $N/2$ output numbers. Let C_2 be the part that receives the more input variables. Let $a_1, \ldots, a_{N/2}$ be the ranks of the output variables of C_2 in the sorted order of the N input numbers. We extend the sequence to $a_0 = 0$ and $a_{N/2+1} = N + 1$. Note that these ranks are independent of the input values. Letting $\{1, \ldots, F\}$ be the range of possible keys, we next define the sequence $b_0, \ldots, b_{N/2+1}$ by the recurrence relation

$$b_0 = 0, \qquad b_{i+1} = b_i + \alpha(a_{i+1} - a_i - 1),$$

where $\alpha = \lfloor F/N \rfloor$. We can verify that all b_i's lie in the key range. The next step is to assign all the N_1 input numbers of C_1 and any set of $(N/2 - N_1)$ input numbers in C_2 to $b_1, \ldots, b_{N/2}$.

Now, we know that, if we assign the remaining input numbers (all of which are in C_2) to any values such that for all i, $a_{i+1} - a_i - 1$ of them lie between b_i and b_{i+1}, the inputs will be mapped to the $N/2$ output ports of C_1. Therefore, the total number Q of output sequences obtainable in C_1 will give a lower bound on 2^I.

To evaluate Q, we must count the number of ways to choose $a_{i+1} - a_i - 1$ numbers between b_i and b_{i+1}. To avoid repetitions, it is easier to assume that these numbers are all distinct. Since $a_{i+1} - a_i - 1 = 0$ implies $b_i = b_{i+1}$, we have

$$Q \geq \prod_i \binom{\alpha(a_{i+1} - a_i - 1) - 1}{a_{i+1} - a_i - 1} \qquad \text{for all} \quad i,\ a_{i+1} - a_i - 1 \neq 0$$

with $a_{N/2+1} = N$ and $a_0 = 0$. Since $\sum_{0 \le i \le N/2}(a_{i+1} - a_i - 1) = N/2 - 1$, we derive

$$Q \ge \min \prod_i \binom{\alpha N_i - 1}{N_i} \tag{6}$$

with $0 \le i \le m \le N/2$, $N_i \ne 0$ and $\sum_i N_i \ge N/2 - 1$. Hence, $Q \ge (\alpha - 1)^{N/2-1}$ since $\binom{\alpha x - 1}{x} \ge (\alpha - 1)^x$ for any $x \ge 1$ and $\alpha \ge 2$. It follows that $I = \Omega(N \log_2 \alpha)$ when $\alpha \ge 2$, and finally $I = \Omega(N(k - \log_2 N)) = \Omega(Nk)$, since $k \ge 2 \log_2 N$.

We can now generalize to the case in which we cannot bisect the N output variables exactly. If M is the maximum number of keys passing through an output port, at worst we can only bisect the circuit so that C_2 produces $N/2 + M/2$ output numbers. In this case, the same reasoning leads to an inequality similar to (6), with $0 \le i \le m \le N/2 + M/2$, $N_i \ne 0$ and $\sum_i N_i \ge (N - M)/2 - 1$, yielding $Q \ge (\alpha - 1)^{N/2 - M/2 - 1}$. Therefore, from the fact that it takes at least time $\Omega(Mk)$ to output M numbers on a single port, the result of Lemma 9 shows that $T = \Omega((Nk - Mk)^{1/2} + Mk) = \Omega(Nk)^{1/2}$. □

7. *Matrix Arithmetic*

7.1 MATRIX MULTIPLICATION AND RELATED PROBLEMS. Although computing the product of three matrices is transitive, multiplying two matrices is not; yet it carries similar though weaker properties, which lead to the same results.

THEOREM 11. *Let A and T be, respectively, the area and the time of a circuit that computes the (Boolean or integer) product of two square matrices, each represented on N bits. We have $A = \Omega(N)$ and $T = \Omega(N^{1/2})$.*

PROOF. We first prove that computing a product $H = F \times G$ of two Boolean matrices requires memorizing most of the bits. Let $N = m^2$ denote the number of bits in each matrix. A remarkable feature of the matrix product is that if we set F (respectively, G) to be a permutation matrix, H appears as a permutation of the rows (respectively, columns) of G (respectively, F). In particular, any input bit can be mapped into m distinct output ranks. Recall that the order in which the bits are input into the circuit is fixed. A pair of bits (one of each matrix) can be mapped into $2m - 1$ different positions on H, and m pairs of bits can be mapped into at least $2m - 1$ positions. Similarly, pm pairs of bits can be mapped into $2pm - p^2$ distinct positions, since p lines and p columns have p^2 common points.

If we consider the $N/2$ bits of F and G input last, we observe that they can be mapped into $3N/4$ distinct positions. Therefore, there exists an input bit x of rank greater than $N/2$ which can be mapped onto an output bit y of rank less than $N/4$. Without loss of generality, assume that it is a bit from F, and let G be a permutation matrix performing this mapping. Only $N/4$ of $N/2$ bits of F input before x can be output before y, which implies by a simple counting argument that the circuit must memorize at least $N/4$ bits; hence $A = \Omega(N)$.

When the m^2 elements of the input matrices are now k-bit integers ($N = m^2 k$), we can use a technique borrowed from Vuillemin [20] to reduce the problem to the previous one.

One matrix is a permutation matrix, except for the nonnull elements that are replaced by 2^i, for $0 \le i < k/2$. The elements of the other matrix are k-bit numbers of the form $(x_{k/2} \cdots x_1 x_{k/2} \cdots x_1)$, and we consider the mapping of the bits $x_1 \cdots x_{k/2}$ onto the positions occupied by the bits $y_k \cdots y_{k/2}$ of H, where an element of H is of the form $y_{2k} \cdots y_1$. Noting that we can permute both lines and columns, as well as perform cyclic shifts on the x_i's, we find that pairs of input bits can be mapped into $(2m - 1)k/2$ distinct positions. Similar to the Boolean case, the last $Nk/4$ pairs of bits to be read in can be mapped into $3Nk/8$ positions. Since we have restricted our attention to the mapping of $Nk/2$ pairs of inputs onto $Nk/2$ outputs, one input bit can be mapped to an output bit of rank less than $Nk/8$. Therefore the circuit must store $Nk/4 - Nk/8 = Nk/8$ bits, which proves the result on the area.

We can use a general result from Savage to bound the time of computation [15]. He has shown that the minimal cross-flow associated with the multiplication of two matrices is at least $Nk/20 - M/2$, where M is the maximum number of bits output on one port. Since it takes time M to output these M bits, T satisfies $T = \Omega((Nk - M)^{1/2} + M) = \Omega(Nk)^{1/2}$. □

Note that the hex-connected systolic array proposed by Kung and Leiserson [8] is optimal in area and time with today's technologies. Also, Savage [15] shows how to reduce inversion and transitive closure of matrices to matrix multiplication. Thus, the previous results are also valid for these two problems.

7.2 DETERMINANT. We can apply the notion of fan-in to derive a lower bound on the complexity of computing determinants.

LEMMA 12. *The time required to compute the determinant of an arbitrary $m \times m$ matrix is $\Omega(m)$.*

Because of Theorem 4, we simply have to show that computing a determinant involves a fan-in on $\Omega(m^2)$ elements.

PROOF. Consider the matrix A_k, defined by the recurrence $A_1 = (a_{1,1})$, and

$$A_k = \left[\begin{array}{ccc|c} & & & r_i \\ & A_{k-1} & & \vdots \\ & & & r_{k-1} \\ \hline a_{k,1} & \cdots & a_{k,k-1} & 0 \end{array}\right],$$

where the r_i's are the sums of A_{k-1} rows; that is, $r_i = a_{i,1} + \cdots + a_{i,k-1}$. Noting that we can rewrite A_k as

$$A_k = \left[\begin{array}{ccc|c} & & & 0 \\ & A_{k-1} & & \vdots \\ & & & 0 \\ \hline 0 & 0 \cdots & 0 & 1 \end{array}\right] \times \left[\begin{array}{ccc|c} & & & 1 \\ & I_{k-1} & & \vdots \\ & & & 1 \\ \hline a_{k,1} & \cdots & a_{k,k-1} & 0 \end{array}\right]$$

we derive the relation

$$\det(A_k) = -\det(A_{k-1}) \times (a_{k,1} + \cdots + a_{k,k-1}),$$

which proves that the assignment $a_{ij} = 1$ for all i, j; $1 \le j < i \le k$, gives a hard input, that is, an input for which a change in any one of these assignments alters the value of the determinant. This shows that computing $\det(A_k)$ involves a fan-in on $k(k - 1)/2 + 1 = \Omega(k^2)$ elements, which completes the proof. □

8. *Linear Transforms and Discrete Fourier Transform*

Vuillemin [20] has shown that any circuit that can compute any linear transform on N k-bit elements (with $k \ge \log_2 N$) computes a transitive function of degree Nk.

It follows from Theorem 8 that space (A) and time (T) requirements satisfy: $A = \Omega(Nk)$ and $T = \Omega(N^{1/2}k^{1/2})$. Note that we are often interested in computing only specific linear transforms. The previous lower bounds no longer hold, since they yield no information on the behavior of a particular transform. We choose to turn our attention to one of the most important, the discrete Fourier transform (DFT). For this problem, a lower bound is already known: $AT^2 = N^2k^2$. We extend this result in the following.

LEMMA 13. *Any circuit computing a DFT on N k-bit numbers requires area* $A = \Omega(N)$ *and time* $T = \Omega(N^{1/2}k^{1/2})$.

PROOF. The DFT is computed in the ring of integers modulo M. Let ω be a primitive Nth root of unity in the ring. Necessarily $M > N$; moreover we suppose that $k = \lfloor \log_2 N \rfloor + 1$. The DFT is defined by $Y = AX$, where $X = (x_0, \ldots, X_{N-1})$, $Y = (y_0, \ldots, y_{N-1})$, and the matrix A is (ω^{ij}), for $0 \leq i, j < N$.

Noting that the first element y_0 is the sum of all the x_i's, we can prove that one of its bits is a fan-in of $O(Nk)$ input bits, by exhibiting a *hard-input*. Let $x_0 = 2^j - 1$ with $j = \lfloor k/2 \rfloor$, and $x_i = 0$ for $i = 1, \ldots, N - 1$. The jth bit of $y_0 = 2^j - 1$ is equal to 1; however, a change in the value of any bit of order $\leq j$ of any $x_i (i > 0)$ will force it to 0. Hence, this particular bit is a fan-in of $Nk/2$ input bits, which yields the desired lower bound on the time.

To prove the result on the area, we show that the circuit must memorize at least N bits. Since the order in which the bits are input is fixed, consider the bit b input last, with b being the sth-order bit of x_j. Any y_i can be written as $y_i = a_i + \omega^{ij}2^s b$, where a_i is independent of b. It is clear that changing the value of b affects the value of all the y_i's, since $\omega^{ij}2^s$ cannot be zero modulo M.

It follows that, at the instant which just precedes the reading of b, at least one bit of every y_i cannot have been output. Since the DFT is invertible, these bits must be able to take on arbitrary values, which implies that the circuit must memorize at least N bits. □

9. *Another Model*

Although we believe that our model is in a sense *minimal*, it is not at all clear whether it can be used to describe precisely the complexity of circuits in real technologies. In this section we consider a more restrictive model tailored to the NMOS technology, that we call the dissipative model. Although it differs from the general model only by three additional assumptions, we can show that it is sufficiently strengthened to give way to stronger lower bounds.

9.1 THE ADDITIONAL ASSUMPTIONS. Introducing the *energy* as a new parameter, we make the following assumptions:

—To switch a gate requires one unit of energy.

—Memorizing a bit requires a unit of energy per unit of time.

—The energy is supplied to the circuit through its (planar) boundary, and the density of energy at any point is bounded by a constant.

These additional assumptions are justified in [5] by electrical considerations. The main constraints are that the electrical power be supplied through conducting wires and that the density of current at any point of a wire be, in practice, always bounded by a constant. Thus it becomes impossible to supply enough power to certain circuit layouts, which of course changes the complexity of some problems a great deal. In the future, this problem may be solved by using the third dimension to supply power, but even in a three-dimensional model, severe problems of power supply and heat dissipation will be difficult to avoid.

9.2 TRANSITIVE FUNCTIONS. It comes as no surprise that, since our second model adds physical constraints to the one in which Vuillemin derived his lower bounds, we can significantly improve upon his results. Before proceeding, we will establish a preliminary result.

LEMMA 14. *If N gates in a circuit are switched at the same time, their convex hull has a perimeter $\Omega(N)$.*

PROOF. Since all the power comes from outside the circuit and is transmitted on the plane, the power inside any convex region of the circuit is at most proportional to its perimeter. Since switching a gate requires a unit of energy, the result is straightforward. □

We can now prove our main result.

THEOREM 15. *Any circuit of perimeter Π that computes a transitive function of degree N in time T satisfies $\Pi = \Omega(N)$, $T = \Omega(N)$.*

PROOF. It has been shown in [20] that the circuit must have the capability of memorizing N bits. Therefore Lemma 14 implies that the circuit must have two active gates G_1 and G_2 at a distance $\Omega(N)$ apart; hence $\Pi = \Omega(N)$. We can always assume that for some values of the inputs, information will be transmitted from G_1 to an output port P_1 (same with G_2 and an output port P_2). Consider now an arbitrary input port R. Since the function is transitive, there exists a path in the circuit from R to P_1 and from R to P_2. Among all possible computations, the four paths $G_1 - P_1$, $G_2 - P_2$, $R - P_1$, and $R - P_2$ will be actual datapaths at least once. From Lemma 1, it then follows that T is at least proportional to $\max\{G_1P_1, G_2P_2, RP_1, RP_2\}$. The sum of these four lengths is greater than $G_1G_2 = \Omega(N)$ as shown in Figure 3, which completes the proof. □

Remark. In this model, these lower bounds are tight for some problems; for example, optimal circuits for performing integer multiplication, based on the Shift&Add scheme, can be found.

9.3 ADDITION

THEOREM 16. *In the dissipative model, the time T required to add two N-bit integers satisfies*

$$T = \Omega(N^{2/3}).$$

PROOF. We can prove this result with the same technique used in the general model. Keeping the same notation, we simply introduce Π as the perimeter of the convex hull of all the active nodes. Since the circuit cannot store more than Π bits at any time, the result of Theorem 5 is still valid if we replace A by Π, which gives $T > N^2/(P\Pi)$. On the other hand, $T > P$ for obvious reasons and since, as we will see, $T > \Pi$, the result follows directly. To prove the last inequality, we consider the two active nodes G_1, G_2 that are furthest apart. There exists a datapath from an input port P_1 to G_1 (respectively, P_2 to G_2). Since there is a fan-in between any pair of input variables, there exists a datapath from P_1 and P_2 to a common point R. The same geometric argument used in the proof of Theorem 15 leads to $T > G_1G_2 = \Omega(\Pi)$, which completes the proof. □

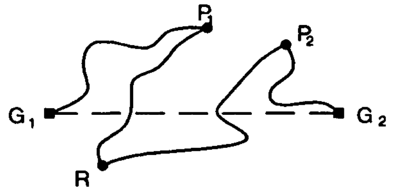

FIG. 3. Computing a transitive function requires linear time.

10. *Conclusions*

The *iterative model of computation* proposed in this paper provides a framework for assessing the asymptotic limitations of *physical computation.* Our aim has been to gather minimal requirements with which any physical computing device must comply. In this model, a circuit is essentially a cellular automaton. Certain assumptions, such as convexity, have been made for the sake of realism and convenience; they might affect complexity results here and there, but they are definitely not essential to the fundamental nature of the model.

Of course, casting problems in a framework of *minimal* models is bound to provide negative insights only, since these models are primarily suited for proving lower bounds. Another avenue of research concerns the study of technology-dependent models, with enough refinement to allow for *à la Knuth* analysis of circuits. We hate to think that each technology should bring along its own model, drastically different from the others. Yet, it is still difficult to evaluate the level of modeling sophistication required for reflecting reality faithfully. This is all the more acute since the analysis of actual circuits should not be only asymptotic but should also apply to arbitrary sizes.

ACKNOWLEDGMENTS. We wish to thank Jean Vuillemin for suggesting this research, and Mike Foster for many interesting discussions about current technologies. Our thanks also go to Gérard Baudet for his contribution in Theorem 5, to H. T. Kung for sharing our interest in this work, and to the referees for helping with the overall presentation.

REFERENCES

1. BAUDET, G. M. On the area required by VLSI circuits. In *VLSI Systems and Computations*, H. T. Kung, R. Sproull, and G. Steele, Eds. Computer Science Press, Rockville, Md., 1981, pp. 100–107.
2. BAUDET, G. M., PREPARATA, F. P., AND VUILLEMIN, J. Area-time optimal VLSI circuits for convolution. *IEEE Trans. Comput. C-32,* 7 (July 1983), 684–688.
3. BILARDI, G., PRACCHI, M., AND PREPARATA, F. P. A critique of network speed in VLSI models of computation. *IEEE J. Solid-State Circuits SC-17* (Aug. 1982), 696–702.
4. BRENT, R. P., AND KUNG, H. T. The chip complexity of binary arithmetic. In *Proceedings of 12th Annual ACM Symposium on Theory of Computing* (Los Angeles, Calif., Apr. 28–30). ACM, New York, 1980, pp. 190–200.
5. CHAZELLE, B., AND MONIER, L. Towards more realistic models of computation for VLSI. In *Proceedings of 2nd Caltech Conference on VLSI* (Pasadena, Calif., Jan. 19–21). 1981, pp. 441–453.
6. CLARK, W. A. From electron mobility to logical structure: A view of integrated circuits. *ACM Comput. Surv. 12,* 3 (Sept. 1980), 325–358.
7. KEDEM, Z. M., AND ZORAT, A. On relations between input and communication/computation in VLSI. In *Proceedings of 22nd IEEE Symposium on Foundations of Computer Science.* IEEE, New York, 1981, pp. 37–44.
8. KUNG, H. T., AND LEISERSON, C. E. Systolic arrays (for VLSI). In *Introduction to VLSI Systems*, C. Mead and L. Conway, Eds. Addison-Wesley, Reading, Mass., 1980.

9. LIPTON, R. J., AND SEDGEWICK, R. Lower bounds for VLSI. In *Proceedings of 13th Annual ACM Symposium on Theory of Computing* (Milwaukee, Wis., May 11–13). ACM, New York, 1981, pp. 300–307.
10. LIPTON, R. J., AND VALDES, J. Census functions: An approach to VLSI upper bounds. In *Proceedings of 22nd IEEE Symposium on Foundations of Computer Science.* IEEE, New York, 1981, pp. 13–22.
11. MEAD, C., AND CONWAY, L., EDS. *Introduction to VLSI Systems.* Addison-Wesley, Reading, Mass., 1980.
12. PREPARATA, F. P., AND VUILLEMIN, J. Area-time optimal VLSI networks based on the cube-connected cycles. Rapport IRIA No. 13, Rocquencourt, France, 1980.
13. PREPARATA, F. P., AND VUILLEMIN, J. The cube-connected cycles: A versatile network for parallel computation. *Commun. ACM 24,* 5 (May 1981), 300–309.
14. ROSENBERG, A. L. Three-dimensional VLSI: A case study. *J. ACM 30,* 3 (July 1983), 397–416.
15. SAVAGE, J. E. Area-time tradeoffs for matrix multiplication and related problems in VLSI models. *J. Comput. Syst. Sci.* (1981), 230–242.
16. SEITZ, C. L. Self-timed VLSI Systems. In *Proceedings of 1st Caltech Conference on VLSI* (Pasadena, Calif., Jan.). 1979.
17. THOMPSON, C. D. Area–time complexity for VLSI. In *Proceedings 11th Annual ACM Symposium on Theory of Computing* (Atlanta, Ga., Apr. 30–May 2). ACM, New York, 1979, pp. 81–88.
18. THOMPSON, C. D. A Complexity Theory for VLSI. PhD dissertation, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pa., 1980.
19. THOMPSON, C. D. Fourier Transforms in VLSI. *IEEE Trans. Comput. C-32,* 11 (Nov. 1983), 1047–1057.
20. VUILLEMIN, J. A combinatorial limit to the computing power of VLSI circuits. *IEEE Trans. Comput. C-32,* 3 (Mar. 1983), 294–300.

RECEIVED OCTOBER 1981; REVISED JANUARY 1985; ACCEPTED JANUARY 1985