CMU-CS-81-107

A Model of Computation for VLSI with Related Complexity Results

Bernard CHAZELLE and Louis MONIER

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

February 1981

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.

Table of Contents

List of Figures

Abstract

We propose a new model of computation for VLSI which is a refinement of previous models and makes the additional assumption that the time for propagating information is linear in the distance. Our approach is motivated by the failure of previous models to allow for realistic asymptotic analysis. While accommodating for basic laws of physics, this model tries to be most general and technology-independent. Thus, from a complexity viewpoint, it is especially suited for deriving lower bounds and trade-offs. We present new results for a number of problems including fan-in, addition, transitive functions, matrix multiplication, and sorting.

1 Introduction

The importance of having general models of computation for VLSI is apparent for various reasons. Among the chief ones, we must include the need for evaluating and comparing circuit performances, showing lower bounds and trade-offs on area, time, and energy, and more generally building a complexity theory of VLSI computation.

While these models must be simple and general enough to allow for mathematical analysis, they must also reflect reality independently of the size of the circuit. We justify the latter claim by observing that if 1980's circuits are still relatively small, the use of high-level languages for designing chips, combined with the possibility of larger integration and bigger chips, will make asymptotic analysis necessary in the near future.

Yet as circuits are pushed to their physical limits, constraints which could be ignored before become major problems and must be accounted in the models. For example, the laws of physics show that the propagation of information takes time at least proportional to the distance, which invalidates the assumptions made in previous models, whereby long wires can be driven in constant time [MC80,TH80]. Generally speaking, one major flaw in those previous models has been to regard a circuit as a topological interconnection of nodes where transmission delays between adjacent nodes could be ignored. Instead, we propose to take into account the geometry of the circuit by assuming a propagation delay at least linear in the distance.

The model which we propose tries to be most general and is thus primarily suited for deriving lower bounds on the complexity of circuits. By using geometric arguments, we are able to prove new complexity results for a varied set of problems, including fan-in, addition, cyclic shift, integer multiplication, convolution, linear transform, product of matrices, and sorting. Another purpose of this paper is to introduce several new techniques of complexity analysis, one of which is due to Gérard Baudet.

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 where power considerations lead to more stringent requirements. We show how it is then possible to derive stronger lower bounds on the complexity of some problems.

2 The Model

Our model is for the most part a refined version of the current planar models found in the literature [TH80, BK80, VU80]. These models, which are all in fact very similar, have been used to both find lower bounds and make comparative analyses of circuits. They differ from the traditional RAM model in charging area costs for the transmission of information. However, they show some inconsistency by failing to account for propagation delays. Our model tries to remedy this shortcoming by including all the parameters necessary for performing *realistic* asymptotic analyses. In particular, the length of wires becomes crucial in evaluating the

time performance of a circuit. As a result, a circuit appears as a fully geometrical object rather than a topological one.

2.1 Model of computational device

We can think of a computational device as a *black box* which computes a boolean function $(y_1,y_2,...)=F(x_1,x_2,...)$. Information is treated digitally, and is communicated to the device through I/O ports in a fixed format. To each variable x_i (resp. y_i) is associated both an input (resp. output) port and a chronological rank on the set of inputs (resp. outputs). In addition, to each variable must correspond an 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 [VU80]. On the other hand, that information may be provided by the circuit (output) or the user (input), thus allowing for self-timed computations [MC80,SE79].

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 the wires are allowed to lie on a constant number of parallel layers. This implies that there are at most a constant number of cross-overs 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) which compute boolean functions. The circuit is laid out within a convex region with all the I/O nodes lying on its boundary.

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 will always refer to this worst-case measure. Other approaches can be considered, involving average or even best time of computation. Of course this is pertinent only with self-timed circuits.

Another important parameter is the *period* of a circuit [VU80]. It is defined only for circuits whose inputs and outputs are available at fixed times and locations. It is then possible to pipeline the computations on several sets of inputs, and we define the period P as the minimal time interval separating two input sets. More precisely, if $(a_1,...,a_N)$ and $(b_1,...,b_N)$ are two sets of inputs, pipelining the computation means that 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. The circuit being viewed as a directed graph, we associate with each node (including I/O ports) a boolean variable $I_i(t)$ [resp. $O_j(t)$] for each incoming [resp. 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, to each node corresponds a state S(t) chosen among a finite number of possible states. Then each node of the circuit computes a function F of the form

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

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 best 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

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

The last assumption asserts the bandwidth limitation of wires. Although we do not wish to restrict the storage capacity of the wire to one bit, we will assume that a wire can carry a number of bits at most proportional to his length. The simplest way to express it 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 one 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 can be propagated from x to V before the circuit completes its computation.

2.3 Justification of the assumptions

Like previous models, this one 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. 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 non-null 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 [CM81]. One major consequence is that, because of the diffusion law, delays on wires are in fact proportional to the *square* of the length. In this technology, it then 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 [SE79], mainly because the current intensities involved are very low, and many electrical phenomena (overheat, metal migration [CL80]) impose a limit on them.

3 Distributing and Collecting Information

To fan in or fan out information being two of the most common operations performed by circuits, we first turn to these problems from which we can best measure the significant departure of our model from the previous ones. These 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 boundary of length 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 time $\Omega(N)$ 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 $y = f(x_1,...,x_N)$ is a fan-in of degree N if there is an assignment of the N variables such that for any i, $f(a_1,...,a_i,...,a_N) \neq f(a_1,...,\neg a_i,...a_N)$. The N-uple $(a_1,...,a_N)$ 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.

Theorem 4: If T (resp. A) denotes the minimum time (resp. 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 time $\Omega(N/p)$. Also, there must be a datapath from any input variable to the point where the value of the function is available. Then, 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. The addition for example falls in that category, since the last carry depends upon 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 our model relates the time of computation to the geometry rather than the topology of the circuit, we can show that many complete binary tree based schemes cease to have the logarithmic time complexity which 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

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

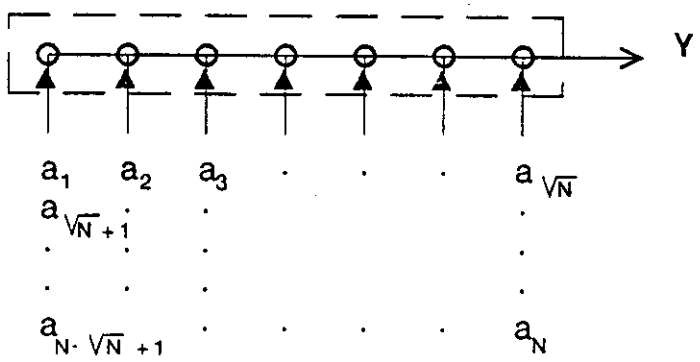

Figure 1: Computing $Y = a_1$ *op* a_2... *op* a_N takes $\Omega(N^{1/2})$ time and area.

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 will equate all constant factors with 1 when this does not compromise the reasoning. Let $Y = y_{N+1}y_N...y_0$ be the result of the addition of the two N+1-bit integers $X = x_N...x_0$ and $111....1_2$. Since all the bits of X are not necessarily read at the same time, let $t_1,...,t_p$ be the instants when bits of X are read in. If Z_i denotes the set of X's bits input at time t_i, we can partition the whole input X into p subsets $Z_1,...,Z_p$. Note that the Z_i's may not be subsequences of X since low-order bits are not necessarily read first. Turning now to the distribution of $X_i = x_i...x_0$ among the sets $Z_1...Z_p$, we call t_{s_i} the time when all the bits of X_i have finally been read into the circuit. Clearly, $X_i \subset Z_1 \cup ... \cup Z_{s_i}$. For each j, $1 \le j \le s_i$, we define $L_j = |X_i \cap Z_j|$, that is the number of bits in $\{x_i,...,x_0\}$ read at time t_j. The sum of the L_j's satisfies: $L_1 + ... + L_{s_i} = |X_i| = i+1$. We can now establish a lower bound on the time required for computing y_i.

Lemma 6: For any i, $0 \le i \le N$, y_i cannot be computed before time $t = \max\{L_1 + t_1,...,L_{s_i} + t_{s_i}\}$.

Proof: We observe that the function $y_i = f_i(x_i,...,x_0)$ is a fan-in with (0,0,...,0) as a hard input. Therefore for any j, $1 \le j \le s_i$, a fan-in must be performed on the L_j bits of $X_i \cap Z_j$ before $f_i(0,0,...,0)$ 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's. □

Let Y(t) designate the number of Y's bits so far output at time t. Similarly X(t) denotes the number of X's bits already read at time t. By convention we define $X(t_i)$ as $|Z_1| + ... + |Z_i|$. We use the previous result to evaluate the growth of the function Y(t).

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

Proof: Let y_m be the highest-order bit output so far at time t. We clearly have

$$(1)\ m \ge Y(t)-1.$$

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

$$(2)\ m+1 \leq \Sigma_{1\leq j\leq i} \min(|Z_j|, t-t_j)$$

since at time t, all the bits $x_1,...,x_m$ have been already read, and thus $t_{s_m} \leq t_i \leq t$. Combining (1) and (2) proves the claimed result. □

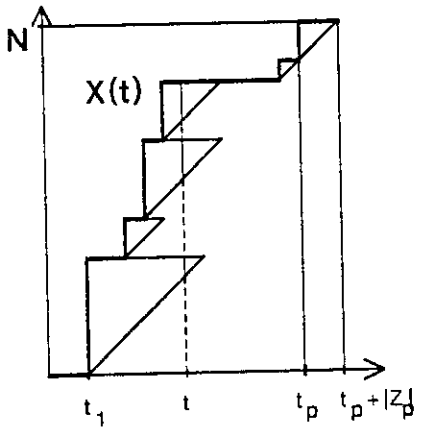

Figure 2: The lower bound on ATP for the addition.

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 we have, setting t_1 to 0,

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

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

$$(3)\ \int_0^T [X(t)-Y(t)].dt \geq N^2/p.$$

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 first introduced by G. Baudet [BA81]. Let P be the period of the circuit. Assuming that problems are treated in a pipeline fashion, let $P_1,....P_m$ 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 $\Sigma_{1\leq j\leq m}$ [N - Y(t-(j-1)P)] output bits [resp. $\Sigma_{1\leq j\leq m}$ [N - X(t-(j-1)P)] input bits] have yet to be produced [resp. read] with regard to problems $P_1,...,P_m$. 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

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

hence

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

or

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

Integrating over t from 0 to P leads to

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

that is $AP \geq \int_0^T [X(t)-Y(t)].dt$. From (3) 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 [VU80], J. Vuillemin has shown that the transitivity of a function has heavy consequences on its complexity in a VLSI model. Roughly speaking, a function is transitive of degree N if it computes a transitive group of permutations acting on N elements. More precisely the function $(z_1,...,z_N) = F(x_1,...,x_N,s_1,...,s_b)$ is transitive of degree N if, by assigning special values to the variables $s_1,...,s_b$, the output is simply a permutation of the variables $x_1,...,x_N$. We also require that the set of all possible permutations thus obtained form a transitive group, i.e., any x_i can be mapped onto any output bit z_j. Among well-known problems which 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, $N = p(2k+\log_2 p)$.
- Product of three $p \times p$ matrices, $N = p^2(2k+\log_2 p)$.

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

Using the geometric nature of our model, we can improve upon J. 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 J. Vuillemin [VU80]. Let p be the number of output ports actually used. Consider one of the input variables which are 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 to notice 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 [PV79,PV80] or other recursive schemes [BPV80,TH80a]). In fact, our model seems to rule out any logarithmic time circuit. However, good performances on the period can be expected from pipelining the computation. Note that although we improve upon the lower bound for T, the well-known trade-off $AT^{2\alpha}=\Omega(N^{1+\alpha})$, valid for $0\leq\alpha\leq 1$, remains unchanged.

6 Sorting and Matrix Multiplication

Although these two problems are not transitive, we can prove similar bounds on A and T. Note that C. Thompson [TH80] and J. Savage [SA79] 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* [TH79] by introducing geometric arguments. Consider a line L partitioning the circuit into two parts C_1 and C_2, each producing roughly half the output bits. We define the minimal cross-flow I to be the minimal number of bits crossing L. More precisely, let U_1 (resp. U_2) and V_1 (resp. V_2) denote the set of input (resp. 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. Wlog, 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 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. 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 [VU80], 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 will assume that $k \geq 2\log_2 N$, implying in particular that all the numbers *can* be different.

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. We assume that each I/O port handles all the bits of a number and not only parts of the bits. 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 which 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

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

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}, then 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 $Q \geq \prod_i \binom{\alpha(a_{i+1} - a_i - 1)}{a_{i+1} - a_i - 1}$ for all i, $a_{i+1} - a_i - 1 \neq 0$ with $a_{N/2+1} = N$ and $a_0 = 0$.

Since $\sum_{0 \leq i \leq N/2}(a_{i+1} - a_i - 1) = N/2 - 1$, we derive

$$(1)\ Q \geq \min \prod_i \binom{\alpha N_i - 1}{N_i}$$

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

We can now generalize to the case where 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 an inequality similar to (1), with $0 \le i \le m \le N/2+M/2$, $N_i \neq 0$ 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 $T = \Omega((Nk-Mk)^{1/2}+Mk) = \Omega(Nk)^{1/2}$. □

6.3 Matrix multiplication

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 which computes the (boolean or integer) product of two square matrices represented by N bits. We have $A = \Omega(N)$ and $T = \Omega(N^{1/2})$.

Proof: We shall 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 (resp. G) to be a permutation matrix, H appears as a permutation of the lines (resp. columns) of G (resp. 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. Wlog, assume that it is a bit from F, and let G be a permutation matrix performing this mapping. Only N/4 of the 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 ($N = m^2k$), we can use a technique borrowed from J. Vuillemin [VU80] to reduce the problem to the previous one.

One matrix is a permutation matrix, except for the non-null elements which are replaced by 2^i, for $0 \le i < k/2$. The elements of the other matrix are k-bit numbers of the form $(x_{k/2}...x_1x_{k/2}...x_1)$, and we consider the mapping of the bits $x_1...x_{k/2}$ onto the positions occupied by the bits $y_k...y_{k/2}$ of H, where an element of H is of the form $y_{2k}...y_1$. Noticing that we can both permute 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. Similarly 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 J. Savage to bound the time of computation [SA79]. 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 H.T. Kung and C. Leiserson [KL78] is optimal in area and time with today's technologies. Also, J. Savage [SA79] 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 Another Model

Although we believe that our model is in a sense *minimal*, it is not clear at all whether it can be used to precisely describe the complexity of circuits in real technologies. In this section, we consider a more restrictive model tailored to the NMOS technology. 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.

7.1 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 [CM81] by electrical considerations. The main constraint is that the electrical power is supplied through conducting wires, and that the density of current at any point of a wire is 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 *vertical* wires to supply power, but even in a three-dimensional model, severe problems of power supply and heat dissipation will be difficult to avoid.

7.2 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 12: 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. Switching a gate requiring a unit of energy, the result is straightforward. □

We can now prove our main result.

Theorem 13: Any circuit of perimeter Π which computes a transitive function of degree N in time T satisfies $\Pi = \Omega(N)$, $T = \Omega(N)$.

Proof: It has been shown in [VU80] that the circuit must have the capability of memorizing N bits. Therefore Lemma 12 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 $\text{Max}\{G_1P_1, G_2P_2, RP_1, RP_2\}$. The sum of these four lengths is greater than $G_1G_2 = \Omega(N)$ as shown in Fig.3, which completes the proof. □

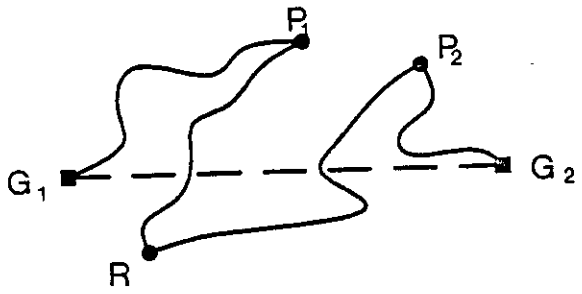

Figure 3: Computing a transitive function requires linear time.

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.

7.3 Addition

Theorem 14: In the NMOS 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 which are furthest apart. There exists a datapath from an input port P_1 to G_1 (resp. 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 13 leads to $T>G_1G_2=\Omega(\Pi)$, which completes the proof. □

8 Conclusions

We have proposed a new model of computation which claims to be more realistic than previous ones, yet tries to remain simple and general. Our aim has been to gather minimal requirements with which any physical computing device should comply. We feel that the ultimate achievement of such endevour would be to lay the foundations of a general theory of *physical computability*. Considerations of *physical complexity* should also be included, as we have tried to do in this paper on a small number of well-known problems.

Of course, casting problems in a framework of *minimal* models can only provide negative insights, since these models are primarily suited for establishing lower bounds. Another avenue of research concerns the study of technology-dependent models granting 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 hard to evaluate the level of modelling sophistication required for reflecting reality faithfully. This is all the more acute as 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, and to H. T. Kung for sharing our interest in this work.

References

[BA81] G.M. Baudet, *On the Complexity of VLSI Circuits : A New Tool*, INRIA 1981, to appear.

[BK80] R.P. Brent and H.T. Kung, *The Chip Complexity of Binary Arithmetic*, proc. 12th Annual ACM Symposium on Theory of Computing, ACM, pp. 190-200, May 1980.

[BPV80] G.M. Baudet, F.P. Preparata and J. Vuillemin, *Area-Time Optimal VLSI Circuits for Convolution*, Rapport INRIA # 30, Rocquencourt France, 1980.

[CL80] W.A. Clark, *From Electron Mobility to Logical Structure: A View of Integrated Circuits*, Computing Surveys, Vol.12, No 3, September 1980.

[CM81] B.M. Chazelle and L.M. Monier, *Towards More Realistic Models of Computation for VLSI*, proc. of Caltech Conference on VLSI, January 1981.

[KL78] H.T. Kung and C.E. Leiserson, *Systolic Arrays for (VLSI)*, appeared in [MC80].

[MC80] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[PV79] F.P. Preparata and J. Vuillemin, *The Cube-Connected-Cycles: A Versatile Network for Parallel Computation*, proc. 20th Annual Symposium on Foundations of Computer Science, Oct. 1979.

[PV80] F.P. Preparata and J. Vuillemin, *Area-Time Optimal VLSI Networks based on the Cube-Connect Cycles*, Rapport IRIA # 13, Rocquencourt France, 1980.

[SA79] J.E. Savage, *Area-Time Tradeoffs for Matrix Multiplication and Related Problems in VLSI Models*, proc. 17th Annual Allerton Conference on Communications, Control and Computing, 1979.

[SE79] C.L. Seitz, *Self-timed VLSI Systems*, proc. of Caltech Conf. on VLSI, 1979.

[TH79] C.D. Thompson, *Area-Time Complexity for VLSI*, proc. 11th Annual ACM Symposium on Theory of Computing, ACM, pp. 81-88, May 1979.

[TH80] C.D. Thompson, *A Complexity Theory for VLSI*, Ph.D. dissertation, Department of Computer Science, Carnegie-Mellon University, 1980.

[TH80a] C.D. Thompson, *Fourier Transforms in VLSI*, Memorandum No.UCB/ERL M80/51, University of California, Berkeley, October 1980.

[VU80] J. Vuillemin, *A Combinatorial Limit to the Computing Power of V.L.S.I. Circuits*, proc. 21st Annual Symposium on Foundations of Computer Science, Oct. 1980.