# Scalability

# Horizons of Parallel Computation

Gianfranco Bilardi and Franco P. Preparata

Department of Computer Science
Brown University
Providence, Rhode Island 02912

**CS-93-20**
May 1993

# Horizons of Parallel Computation[*]

Gianfranco Bilardi[†] and Franco P. Preparata[‡]

May 28, 1993

**Abstract**

This paper considers the ultimate impact of fundamental physical limitations—notably, speed of light and device size—on parallel computing machines. Although we fully expect an innovative and very gradual evolution to the limiting situation, we take here the provocative view of exploring the consequences of the accomplished attainment of the physical bounds. The main result is that scalability holds only for neighborly interconnections, such as the square mesh, of bounded-size synchronous modules, presumably of the area-universal type. We also discuss the ultimate infeasibility of latency-hiding, the violation of intuitive maximal speedups, and the emerging novel processor-time tradeoffs.

## 1 Introduction

Parallel computation has been for some time a hot topic for computer science research. The extraordinary technological advances—globally referred

[*]A preliminary version of this paper appeared in the Proceedings of the Symposium for the 25th Anniversary of INRIA, Springer-Verlag LNCS 653, pp. 155-174; December, 1992.

[†]Dipartimento di Elettronica ed Informatica, Università di Padova, Via Gradenigo 6/A, 35131 Padova, Italy. This author's work was supported in part by the Italian National Research Council and the Italian Ministry of University and Research.

[‡]Department of Computer Science, Brown University, Providence, RI 02912-1910. This author's work was supported in part by NSF Grant CCR91-96152 and by ONR Contract N00014-91-J-4052, ARPA order 8225.

to as VLSI—which occurred over the last fifteen years exhibit the capability to enable what was previously the object of mainly academic speculation. Despite the enabling technology, there is wide consensus, eloquently and authoritatively expressed [S86, V90a, V90b], that parallel computation has not enjoyed to this day the development that was to be reasonably expected, notwithstanding some remarkable realizations (see, e.g., [Be92]) that have appeared in the last few years. A major reason for this failed expectation has been convincingly identified [V90a] in the lack of an agreed-upon model of parallel computation, one that would unleash the independent development of hardware and software, the key feature of the extraordinary success of the von Neumann serial computer. Such model would be the "bridge" between a growing library of programs and an evolving variety of machines conforming with it. In its absence, largely different interconnection structures result in largely diverse programming styles: indeed, the recently marketed systems—comprising hypercubes, CCC-variants, fat trees, meshes, toruses, ring-of-trees, etc.—offer a substantially varied spectrum of solutions.

This paper, however, does not confront this important issue directly. Our present outlook is of a different nature, the focus being on how physical limitaitons are likely to constrain the structure of very large computers. The connection to the theme of the model is supplied by the observation that the physical constraints can play an important role in identifying—almost forcing it on us—the most appropriate model.

The unrelenting progress of computing technology over the past half-century is frequently expressed by sentences whose syntax is: parameter "so-and-so" has been improved by "so-large-a" factor every "so many" years. Such sentences naturally encourage the view of unbounded technological progress. The repercussion of this outlook upon the theory of computing machines is something we may dub the "topological view." In the topological view, a system is a directed graph, i.e., an interconnection of *sites* (be they devices, modules, subsystems, etc.), so that communication between adjacent sites occurs with fixed delay (or, for that matter, instantaneously, if the fixed communication delay is interpreted as some kind of set-up time at the site). This view is commonly, although not felicitously, denoted "synchronous model", reference being made to the assumption that changes of state occur in synchrony with some sort of clocking mechanism. Perhaps the qualifier "topological" could be more appropriate for the following reason: *topology* (in our context, simply a directed graph) will always be implemented as a *geom-*

*etry* (in our context, a laid-out assembly of wires and physical devices). The synchronous model corresponds to the assumption that timing of operations depends only upon the topology of the system and is independent of its realizing geometry (that is, it takes constant time to transmit through an edge regardless of its layout length). This view was naturally elicited by the early generations of computing systems, encompassed by the following eloquent characterization by Vitanyi [Vi86]: "a wire had magical properties of transmitting "instantly" from one place to another." It must be recognized that the synchronous model has been the basis of a number of significant results on size-time trade-offs, simulations between different topologies, achievable speed-ups, latency-hiding mechanisms, which adequately express its algorithmic capabilities. It must also be understood that to some extent, through the clever deployment of a careful mix of technologies and architectural tricks, this view is supported by current computer engineering.

Nevertheless, any buildable machine will consist of physical components and thus will be subject to the laws (and limitations) of physics. Some of these limitations are of an absolute nature (finiteness of the speed of light and layout in ordinary space), others are related to material science as we know it today. These constitute ultimate bounds to what is achievable, for what we know. We fully realize that the road towards these bounds will not end catastrophically, but will be more like a gradual evolution tempered by engineering insights. However, in this paper we take the provocative outlook of considering a technology for which these limitations are fully operative and with which arbitrarily large machines are to be built. Thus, we radically separate topological and geometric structures. The illustrated consequences are admittedly projected in the future, but involve principles that are lurking and must ultimately be reckoned with.

Several of the considerations and suggestions presented here do not entirely originate with this paper. They are an outgrowth of an intellectual climate to which an entire community of researchers has contributed over the years through both scholarly work and machine design. The pioneering work of Chazelle and Monier [CM85] raised severe objections to the synchronous hypothesis, and unequivocally pointed to the goal of laying "the foundation of a general theory of physical computability," which is also the focus of this paper. Vitanyi [Vi88] established lower bounds to wire lengths in term of network diameter under very liberal assumptions, and suggested that "mesh-connected architectures may be the ultimate solution for interconnecting the extremely large computer complexes of the future," a conclusion fully endorsed by this paper. Quite recently Feldman and Shapiro [FS92], alighting from fundamental physical constraints, propose as a realistic model of parallel computation a 3D-mesh-interconnection dubbed "spatial machine."

Further developing these threads, in this paper we analyze how fundamental physical limitations appear to naturally lead to a scalable parallel computing structure in the form of a spatial mesh. Once the mesh is accepted as the candidate interconnection for large computing machines in the limiting technology, we shall recognize the spontaneous emergency of parallelism (i.e., the desirability of a fixed processor/memory ratio), the disappearance of traditional space-time trade-offs, the ultimate infeasibility of latency-hiding mechanism, and the attainability of unintuitive speed-ups through the optimal exploitation of data locality.

## 2 Fundamental Limitations

Although it is very fruitful and appropriate to think of a computation as of an abstract process involving precisely defined mathematical objects—as, for example, the constituents of the Turing machine model—one should never lose sight of the fact that any concrete implementation of that process involves the utilization of physical phenomena. The physical reality will ultimately determine what can be accomplished by means of computing machines.

One may see that the present situation is vaguely reminiscent of the dichotomy between classical and modern views that characterized physics in the early part of this century: on one hand the inducement into indefinitely extrapolating common experience, on the other the awareness of limitations that alter the common-sense intuition of reality. When clock rates are of the order of 1 millisecond, it is natural to view as instantaneous the signal propagation on a wire; when a transistor area is of the order of square millimeters, it is natural to view a size reduction by a factor of two as a desirable technological improvement not affecting the functionality of the device.

Both views have become shakier under the relentless technological progress that occurred in the past decade. The pillars of the revolution that ushered modern physics—relativity theory and quantum mechanics—become increasingly relevant to the structure of computing machines, by exhibiting some hard limitations to what is ultimately physically realizable. We shall now discuss these limitations.

**1. Speed-of-light limitation.** This is the crucial most fundamental constraint. Speed of light is finite, and this fact is already quite relevant to computing systems. This relevance is striking when one considers that the duration of $1ns$ is already "long" for today's technology, and that in $1ns$ light travels a bare $30cm$. Although signal propagation on electric conductors is frequently governed by slower processes (such as charging or discharging a capacitive load) [MC80, BPP82, CM85], we shall assume that signal transmission time is proportional to the length of the connection, which, for sufficiently long wires [BPP82], is the best achievable performance.

**2. Size limitation.** It is reasonable to make reference to a *fixed* technology, characterized by a minimum attainable value of the "feature" [MC80] of an integrated circuit. This, of course, does not naively mean the preclusion of technological improvements (reductions of feature size) that are bound to occur in the near future; our assumption is that forthcoming technological advances will ultimately lead to a situation where the same down-scaling mechanisms will no longer work. The debate is open [BL85] on the minimum size of a digital device or on the minimum energy required by a logical operation. Although no definitive answer appears forthcoming, there is a consensus that a bounded volume can only store a bounded number of bits. This enables our assumption that there exists a minimum size for a digital device (in other words, down-scaling is not endless).

**3. Degree-boundedness.** The topology of any digital network is a directed graph with bounded indegree and outdegree. This corresponds to assume the use of digital gates with bounded fan-in and fan-out, which is a well-known requirement if one wishes to use gates with basically homogeneous performance.

**4. Planar layout.** Although this is not a prescription dictated by any fundamental law, it is reasonable to assume that a digital computing system be physically realized by laying out its constituents on a surface. In a sense, this corresponds to the current common practice of printed-circuit-board assembly of integrated circuits. There are clear engineering reasons favoring this solution: ease of assembly, ease of maintenance, and ability to remove heat generated by switching devices, and, more cogently, ease of fabrication, since the third dimension is used by the processes that create two-dimensional layouts. However, we argue that the removal of this assumption is inconsequential. In fact, in all cases system layout must occur in the ordinary three-dimensional space. Even postulating a rather implausible layout technology where the three dimensions have comparable lattices, we would only achieve a nonessential modification of the conclusions; typically, the replacement of square-roots with cubic-roots in some performance measures related to the physical diameter of the layout domain [Ros81]. Therefore, all space analyses will be expressed in terms of "area".

These four limitations are the basis of the conclusions derived in the forthcoming sections. We shall frequently refer to the model governed by the above constraints as the "limiting technology."

## 3 Machine Modelling and Scalability

A pervasive objective of computer science is the development of increasingly large machines with the capabilities to solve increasing-size instances of any given problem. This motivation has been a driving force in the brief history of the discipline, although serious *caveats* are being raised about postulating arbitrarily large problem sizes (a criticism of asymptotic analysis).

In the most abstract formulation, a digital computing system is a finite-state machine whose behavior is formally specified by a state-transition function. Of course, a description in terms of "states" is totally unwieldy. Instead, computer architecture is normally concerned with structural descriptions of machines as assemblies of well-defined and relatively simple functional constituents. Typically, the description makes reference to a small collection of characteristic parameters, each representing the measure of some significant item or resource (for example, input size, memory size, etc.). Thus, a given architecture implicitly defines a "family" of machines, parametrized as illustrated. Our present objective is to develop a simple but adequate model of such families, referred to as *families of networks of processors*, containing as a special case the conventional uniprocessor systems.

In view of the size limitation discussed in Section 2, it is ideally desirable that machines of a given family be buildable with a homogeneous technology, i.e., by assembling increasing numbers of building blocks of fixed types. In common parlance, this property is called "scalability," and is one of the central objectives of computer architecture. In order to deal with this important notion under the limitations introduced above, we shall now attempt a simple formalization of scalability.

At the abstract level, a computation is a sequence of steps (state transitions) of the machine, and its cost is the number of steps. In a chosen physical implementation, however, each step has a duration (measured in seconds) and the cost of a computation is appropriately measured by execution time. For a class of machines, scalability means that the duration of one machine step is the same for all members of the family (i.e., in the appropriate units, the number of steps is a correct measure of running time). To deal with the more flexible notion of "degree of scalability," we propose the following definition.

**Definition 1.** Let $\mathcal{F} = \{M_k : k \in I\}$ be a family of machines, where $I$ is a set of tuples of architectural parameters. With reference to a chosen physical implementation, we say that $\mathcal{F}$ has *slowdown* $\tau(k)$ if the maximum duration (measured in seconds) of any step of $M_k$ is $\tau(k)$, for all values of $k$. We say that $\mathcal{F}$ is *strict-sense scalable* if $\tau(k) = O(1)$.

We now illustrate the previous definition with a few examples. Often the families arise by considering finite versions of structures theoretically defined as infinite machines.

**Example 1.** The most immediate example of a strict-sense scalable machine is a $d$-dimensional ($d = 1, 2, 3$) *cellular automaton.* We are referring here to the case where for, say, $d = 3$, $k = (k_1, k_2, k_3)$ and $M_k$ consists of $k_1.k_2.k_3$ identical finite-state machines (with structure independent of $k$) placed at the points of integer coordinates of a parallelepiped of sidelengths $k_1, k_2$, and $k_3$, with near-neighbor connections. Observe that, if the structure of the finite-state machine were considered a free parameter, then the resulting family would no longer be strict-sense scalable. Indeed, a machine with a larger number of states requires a larger layout area, and hence a longer time to execute one state transition.

**Example 2.** *One-head Turing machines* also form families of strict-sense scalable machines. Within a family, all members have identical finite-state control and $M_k$ has $k$ cells of tape. $M_k$ is transformable to a one-dimensional cellular automaton of $k$ cells, each containing an instance of the Turing machine's finite-state control, with only one cell active at any time (in other words, a $k$-cell cellular automaton can simulate with no slowdown a Turing machine with a $k$-cell tape).

The fundamental reason why the above are families of scalable machines is that communication occurs only within bounded distance (essentially between spatially contiguous modules whose size is independent of $k$). These are the only examples of strict-sense scalable machines we know of. Consider now the following examples:

**Example 3.** In *multihead Turing machines*, communication between heads introduces a delay proportional to tape length, which precludes strict-sense scalability.

**Example 4.** The *von Neumann machine*, or *Random Access Machine* (RAM) [CR73], $M_m$ has a single memory of $m$ addressable words, each of $\lceil \log_2 m \rceil$ bits (to store addresses), and a CPU tailored to $\lceil \log_2 m \rceil$-bit words. Under the speed-of-light limitation, the memory access time for $M_m$ is $\Omega(\sqrt{m \log m})$ since, in the most favorable situation, the memory is laid out in a planar domain of diameter $\Omega(\sqrt{m \log m})$ and the clock period is tailored to the farthest access. These lower bounds are achievable so that $\tau(m) = \theta(\sqrt{m \log m})$, yielding the conclusion that the von Neumann machine *is not* strict-sense scalable.

This conclusion may seem in contrast with the half century of indisputable success enjoyed by the von Neumann architecture, expressed by both faster clocks and larger memories. However, it can be argued that the "apparent scalability" of the RAM is due to the impressive technological progress that has occurred in its brief history, resulting in faster and smaller devices. Indeed, if we consider just gains in device speed, by increasing $m$ only to a fraction of the size which would maintain constant access time (which is essentially proportional to the *clock period*), a concomitant reduction of $\tau(m)$ has been achieved. Analogous results are obtainable by judiciously allocating gains in device size. Moreover, commercial machines are augmentations of the abstract RAM with a memory hierarchy leading to different access times for different regions of memory. For programs exploiting *locality* the result is a reduction of the average access time, hence of the observed $\tau(m)$.

As indicated earlier, of central importance to parallel computing are machine families based on the notion of network of processors.

**Definition 2.** A *network of processors* is a machine $M_{p,G,m}$ characterized by three parameters: $p$, the number of processing elements (PE's), $G$, a directed graph of $p$ vertices defining the processor interconnection, and $m$, the number of addressable memory cells of $\lceil \log_2 m \rceil$ bits each. Each PE is essentially a RAM with a memory of $m/p$ cells, and is uniquely identified by an ID-tag (an integer between 0 and $p-1$). The PE's correspond to the vertices of $G$. The RAM repertoire of instructions is augmented with (one-step) communication primitives allowing a PE to exchange data with its (topological) *neighbors.* The fundamental timing parameter is the largest execution time of such instructions.

We assume here that a network of processors is programmed in a Single Program Multiple Data (SPMD) mode [F66]. Specifically, all PE's execute the same program, but each PE has its own program counter. Although the communication primitives are restricted to adjacent PE's, addresses of instruction operands may range over the entire machine memory. Global memory accesses must be supported by specialized programs (and cannot be regarded as one-step operations). Network architectures are defined around a family of graphs of increasing sizes such as binary trees, meshes, hypercubes, etc.

**Definition 3.** Given a set of positive integers $J$ and a set of directed graphs $\mathcal{G} = \{G_p : p \in J, |G_p| = p\}$, the set $\{M_{p,G_p,m} : G_p \in \mathcal{G}\}$ is a *family of networks of processors.*

To assess the scalability of a family of networks, we must evaluate the maximum execution time of the instructions of its PEs. We consider separately accesses to the PE's private memory and communication primitives.

A PE has a memory with $\theta((m/p)\log m)$ bits. Then, assuming that the CPU has a size comparable to or smaller than the size of the memory, the area of a PE is proportional to $(m/p)\log m$ and, due to the speed-of-light limitation on memory access time, $\tau(p,m) = \Omega(\sqrt{(m/p)\log m})$. The value of $m/p$ will probably be dictated by a number of engineering tradeoffs. We assume here this value to be a constant (although most likely a large one) and will develop later arguments suggesting the adoption of a fixed ratio between memory area and CPU area as $m$ grows. Then, we can regard $\sqrt{(m/p)\log m}$ as essentially constant. (Indeed, for present-day microprocessors $\log m$ is between 32 and 64, and 256 bits would suffice to address each of the $10^{80}$ charged particles estimated to exist in the universe.)

A more stringent constraint on $\tau(p,m)$ arises from the communication primitives. Indeed, $\tau(p,m) = \Omega(L_p)$, where $L_p$ is the length of the longest

Table 1:

| | $B_p$ | $D_p$ | $L_p$ |
|---|---|---|---|
| BINARY TREE | $\theta(1)$ | $\theta(\log p)$ | $\Omega(\sqrt{p}/\log p)$ |
| MESH | $\theta(\sqrt{p})$ | $\theta(\sqrt{p})$ | $\Omega(1)$ |
| CCC | $\theta(p/\log p)$ | $\theta(\log p)$ | $\Omega(p/\log^2 p)$ |
| CUBE | $\theta(p)$ | $\theta(\log p)$ | $\Omega(p/\log p)$ |
| FAT TREE | $\theta(\sqrt{p})$ | $\theta(\log p)$ | $\Omega(\sqrt{p}/\log p$ |

wire in the layout of $G_p$ minimizing such length. It has been shown [L81] that $L_p = \Omega(\max(\sqrt{p}, B_p)/D_p)$, where $B_p$ is the bisection bandwidth and $D_p$ is the topological diameter of $G_p$. We remark that, whereas networks with small diameter and high bandwidth appear desirable in the topological model, they are categorically penalized by the speed-of-light limitation, as the following table illustrates:

The table clearly singles out as highly desirable the mesh of processors described by the following definition:

**Definition 4.** For $p$ a perfect square, the *p-processor mesh* is a network of PEs $\{P_{ij} : 1 \leq i, j \leq \sqrt{p}\}$, where the neighborhood of $P_{ij}$ is $\{P_{i-1,j}, P_{i+1,j}, P_{i,j-1}, P_{i,j+1}\}$ (ignoring $PEs$'s with indices out of range).

In a mesh, each processor can be laid out in area $O((m/p)\log m)$, and the entire network takes area $O(m \log m)$. Considering both memory accesses and near-neighbor communication, we conclude that $\tau(p,m) = \theta(\sqrt{(m/p)\log m})$. Essentially, the slowdown is that due to the PEs, which we have seen to be practically insensitive to $p$ (for constant $m/p$). Hence, we reach the central conclusion that the *mesh of processor is scalable*, substantiating the reported suggestion by Vitanyi [Vi88]. By contrast, the binary tree, the hypercube [P77], and the CCC [PV81] are not scalable. In the rest of this paper we shall adopt the mesh interconnection as the most suitable architecture in the limiting technology, and investigate its important features.

*Remark.* Although meshes and families of bidimensional cellular automata are all arrays of finite-state machines, there is a subtle but far reaching difference between the two families. Whereas nodes of a cellular automaton are indistinguishable from each other and independent of array size, nodes of a mesh are distinguished by their ID, which supports addressability and programmability, and their size increases, albeit slowly, with the array size.

## 4 The Mesh-Connected Structure

Informally, the outstanding feature of the mesh interconnection is that its operation is insensitive to the speed-of-light limitation. Clearly, any other near-neighbor interconnection that admits of a planar layout with bounded length wires, such as the toroidal mesh or the hexagonal mesh, would behave analogously to the standard mesh. Therefore we shall restrict our considerations to the latter with no loss of generality.

In this section we examine in some detail the ability of the mesh to support a shared-memory (P-RAM) programming style, a desirable feature from the user's viewpoint.

There is consensus on the fact that programming for a bounded-degree network in general, and for the mesh in particular, places on the programmer the burden of some tasks, such as *memory allocation*, i.e., assignment of each variable to a specific node of the network (private memory of some PE) where to store it, and *message routing*, i.e., the selection, for each message resulting from a memory reference, of a network path and a schedule between source and destination. It would definitely be convenient for the programmer if these tasks were handled somehow automatically by the compiler and by the runtime system. As is well known, the most serious drawback of this approach is represented by contentions, occurring either among requests/answers competing for the same transmission link or among memory references directed to the same memory node. A number of solutions to this problem have been proposed, based both on randomized and on deterministic algorithms[MV84, UW87, AHMP87, Ran87, KU88, HB88].

These schemes are collectively referred to as P-RAM simulations, and each of them is categorized on the basis of its *slowdown*, i.e., the number of network steps needed to simulate a P-RAM step (a global memory reference). Fortunately, the mesh interconnection lends itself to a P-RAM simulation whose slowdown is of the same order as the intrinsic mesh-traversal time. Indeed, the randomized scheme proposed in [Ran87] for the butterfly network can be adapted in a relatively straightforward manner to simulate one step of a $p$-processor P-RAM on a $p$-node mesh in time $O(\sqrt{p})$, with high probability [A.G. Ranade, private communication, 1992]. Since, on the average, for a set of $p$ random references on a $p$-processor mesh, a substantial fraction of them are destined to nodes at distance $\Omega(\sqrt{p})$ from their respective sources, Ranade's result appears quite satisfactory and indicates the ability for the mesh to efficiently support a shared memory programming model of the P-RAM type, a feature which is highly desirable, and probably necessary, for a machine to be effectively usable. However, it is by no means clear that it would be sufficient, since a P-RAM programming style either ignores or destroys locality, which is an undisputed source of algorithmic efficiency. We shall return to this important issue in Section 8.

## 5 Processor-time Trade-offs

The traditional view of processor-time trade-offs is expressed by the so-called "Brent's scheduling principle," which embodies a general simulation scheme of a system with $p$ processors by a system of $p' < p$ processors of the same type (provided that processor allocation is not a problem). Specifically, although in its original formulation Brent's principle does not refer to any particular computation model, we may legitimately consider simulations between P-RAMSs (for clarity, we shall refer to such simulations by the term "emulation," reserving the term "simulation" to the situation where a physical machine simulates a P-RAM.) For a given problem instance, we have an algorithm running in $S$ steps on a $p$-processor P-RAM and we wish to emulate this computation on a $p'$-processor P-RAM of the same type, with $p' < p$. Then Brent's principle states that the product $pS$ (denoted $W$, "potential work") is a nondecreasing function of $p$, as we can always carry out the outlined emulation with $p/p'$ slowdown, each processor of the smaller P-RAM emulating $p/p'$ processors of the larger one. In other words, Brent's principle embodies a neat slowdown processor-time tradeoff, which rests crucially on the assumption that the execution time of one step is independent of $p$, a characteristic property of the topological model.

We now wish to analyze the nature of such tradeoffs in the limiting technology, which sharply separates the notions of ordinary time T and of number of steps S. Here the tradeoff to be investigated concerns meshes of identical processors, each simulating a P-RAM with the same number of processors. Therefore, the machine to be emulated is a $p$-processor mesh simulating a $p$-processor P-RAM (and the emulating machine is an analogous $p'$-processor mesh). We let $m_0$ denote the size of the processor local memory for which the memory access time equals the CPU instruction execution time. In order to explore the full range of $p'$, we assume that, for a problem requiring global memory $m$, we have $p >> m/m_0$ (corresponding to $S$ execution steps).

The running time of the algorithm will not only depend upon the number of executed steps, but also upon the duration of each such step, which in turn depends upon the layout area of the system. Concretely, let $p', S', T'$ denote number of processors number of steps and running time, respectively, of the emulating system, with $p' \leq p$. Clearly, $S' = Sp/p'$, as specified by Brent's principle. However since processor area dominates memory area, the global mesh area satisfies $A' = \theta(p')$, so that

$$T' = \theta(S'\sqrt{p'}) = \theta(Sp/\sqrt{p'}).$$

Since the product $Sp$ is a constant, we obtain that $\sqrt{p'}T'$, or equivalently $A'T'^2$, is invariant as long as $p' \geq m/m_0$, and that time is at a premium, since a doubling of computation time allows a four-fold reduction of area. This type of trade-off extends down to $p' = m/m_0$, below which value, the layout area is basically determined by the memory requirement $m$, i.e., $A' = \theta(m)$ and

$$T' = \theta(S\sqrt{m}) = \theta(Sp\sqrt{m}/p').$$

In this situation, using fewer than $m/m_0$ processors leads to a clear waste of work since the time $T'$ increases inversely to $p'$ with no corresponding area benefit. The conclusion is that the tradeoff embodied by Brent's principle is not consistent with the speed-of-light limitation, which suggests instead a *balancing of memory and processor resources.* This validates our previous assumption to regard the ratio $m/p$ as a constant.

This tradeoff is diagrammatically illustrated in the $(A, T)$-plane (Figure 1). The dotted curve represents the traditional Brent tradeoff, and the solid curve represents the analogous tradeoff in the limiting technology. The number of processors decreases as we traverse this curve from *a* to *b*.

*Remark.* We now note that if for a given problem instance there exists a sequential ($p = 1$) algorithm running in $S_1$ steps with memory $m$, then in the limiting technology no more than $\theta(mS_1^2)$ processors can ever be effectively used to speed up the solution. Indeed, the running time of sequential algorithm is $S_1\theta(\sqrt{m})$. A parallel system with $mS_1^2$ processors requires area $\Omega(mS_1^2)$ whose traversal uses time $\Omega(\sqrt{m}S_1)$; thus the simulation of a *single* step of the parallel system—which runs in time of the same order of the layout traversal time—runs *in* time comparable to the execution of the *entire* sequential algorithm.

Figure 1: Diagram illustrating the area/time tradeoff.

This observation carries some implication with respect to the widely investigated parallel complexity class NC [P79], which contains exactly those problems that can be solved in polylogarithmic time by a P-RAM with polynomially many processors. Membership in NC is often equated with parallel feasibility. Unfortunately, the correspondence is questionable in two ways. On the one hand, there are algorithms not of the NC type which are excellent parallel algorithms (e.g., many systolic algorithms); on the other hand, some NC algorithms require so many processors that, in the limiting technology, they may be practically slower than some known sequential algorithm for the same problem.

It must be stronly underscored, however, that while it is important to understand the impact of physical limitaitons on the performance of parallel algorithms, a deeper understanding of the structure of computational problems is often better obtained in abstract models of computation (for example, the class NC on the P-RAM) that are not encumbered by the physical limitations. Such models have played and will continue to play a vital role in the development of algorithms.

*Remark.* The preceding limiting-technology analysis suggests an interesting interpretation of the emergence of parallelism. As the memory size grows beyond a given value—basically corresponding to an access time comparable to the appropriate processor clock period—resources are best utilized by introducing additional processing capabilities within the memory real estate, so that the private memories of the processors are kept at a basically constant size.

## 6 Is Latency Hiding Feasible?

*Latency* $\lambda$ (measured as a number of system's steps) is the time elapsed between the issue of a data request and the delivery of data to the requestor. Latency, an increasing function of the memory layout area, has been correctly targeted as a negative measure for a computing system. The following "pipeline" approach, referred to as *latency hiding*, has been sometimes advocated to overcome latency.

In a network with $n$ nodes, $e$ edges, and latency $\lambda$, each of $p \leq n$ processors is time-shared, suppose in a round-robin fashion, by a set of $s \leq \lambda$ processes. Whether these processes correspond to subproblems of the same application, or to different pipelined instances of some application, or totally unrelated applications, is immaterial for the present analysis. We assume that the $n$ nodes are all of comparable size, $p$ of them are full-fledged processors, while the remaining $(n - p)$ ones provide only routing and storage capabilities. A processor devotes to any given process one out of every $\lambda$ steps, so that the time interval between two consecutive steps allocated to the same process is sufficient for a memory reference to be satisfied. The approach assumes that there are $ps$ concurrent processes, for example, as the result of executing on a *physical* $p$-processor machine an algorithm written for a *virtual* machine with $ps$ processors. Full latency hiding corresponds to $s = \lambda$.

We now discuss the feasibility of such an intriguing scheme in terms of network capabilities. We denote by "edge use" the utilization of a network edge to route a request/response between two adjacent processors. Therefore, each memory request accounts for $\lambda$ edge uses for its satisfaction. During an interval of duration $\lambda$ (latency cycle), $ps$ memory references are issued and satisfied, so that $ps\lambda$ edge uses are requested by the latency-hiding mechanism but only $e\lambda$ uses are offered by the network. It follows that $ps\lambda \leq e\lambda$, that is,

$$ps \leq e. \tag{1}$$

For any bounded-degree network, $e = O(n)$, and inequality (1) yields $p =$

$O(n/s)$. Thus, the average number of operations per step (executed by the $p$ processors) is $t = O(ps/\lambda) = O(n/\lambda)$, independently of the value of $p$. Essentially, fewer processors can be active if any of them must be time-shared by more than one process. We now examine the feasibility of latency hiding for the mesh and for the hypercube networks.

We begin by referring to the topological model, where the latency is given by the network diameter. An $n$-node mesh has a diameter $\theta(\sqrt{n})$, so that $\lambda = \theta(\sqrt{n})$. Therefore $t = O(n/\lambda) = O(\sqrt{n})$. This performance can be achieved by choosing either (1) $p = \sqrt{n}$ and $s = \theta(\sqrt{n})$ (only $\sqrt{n}$ processors issuing a memory request at each time unit, i.e., implementing a full latency hiding scheme), or (2) $p = n$ and $s = 1$ (all processors issuing a memory request every $\theta(\sqrt{n})$ time units, i.e., implementing a conventional P-RAM simulation scheme). This shows that, in full generality, no advantage can arise from a latency hiding approach on the mesh interconnection. In essence, latency hiding reduces for the mesh to time-multiplexing the $s$ processes.

By contrast, consider an $n$-node hypercube, for which $e = n \log n$ and the diameter is $logn$. Therefore $\lambda = \theta(\log n)$. Choosing $p = n$ and $s = \log n$, which satisfy (1), yields a number of operations per step $t = O(n)$. Thus, in the topological model, the hypercube has the potential to achieve full latency hiding. Indeed, this potential is exploited in some schemes to simulate an $(n \log n)$-processor P-RAM on an $n$-node hypercube, within $O(\log n)$ steps with high probability [V90b]. Such scheme, however, rests on a node-degree $\theta(\log n)$, which is technically questionable as $n$ grows.

Taking now into consideration the speed-of-light limitation, we realize that the previous analysis of the mesh-interconnection remains valid (because transmission along an edge takes unit time in either model). However, the outlined latency-hiding scheme for the hypercube breaks down, since the hypercube has a slowdown $\tau(n) = \Omega(n/\log n)$ (see Table 1), which translates into $O(\log n)$ operations per unit time, rather than $O(n)$. At first sight, this difficulty may be attributed to the width of the wires, since in standard layout arguments wires are responsible for the large area of high-bandwidth networks. However, an analogous, albeit weaker, lower bound on wire length holds for the hypercube even for hypothetical zero-width wires [Vi88].

On the other hand, the speed-of-light and device-size limitation jointly undermine the feasibility of latency hiding for any network in the limiting technology, even under the extreme assumption of zero-width links (which could model optical connections). Indeed, a processor time-shared by $s$ processes must use $\Omega(s)$ area to store their state information. For $p$ processors, this requirement leads to overall network area $A = \Omega(ps)$, latency $\lambda = \Omega(\sqrt{ps})$, and hence $O(ps/\lambda) = O(\sqrt{ps})$ operations per second. In this case, full latency hiding occurs for $s = p$ (under the optimistic assumption that context switching between processes be doable in $O(1)$ time). The situation is interpreted as follows: For $s = p$, we have a machine (of course, a $p$-node mesh) of area $A = \theta(p^2)$ performing at most $p^2$ operations every $p$ steps. But in the same area, we could have a $p^2$-processor mesh that can emulate a $p^2$-processor P-RAM with the same overall performance ($p^2$ operations every $p$ steps). The only difference between the two schemes is whether the $p^2$ processes are time-interleaved or not. This indicates that under the speed-of-light limitation no performance advantage can derive from implementing latency hiding schemes.

The above discussion relates to the assumption that a single variable is involved in each memory access by a processor. However, similar considerations hold in situations when multiple variables are retrieved in a single access (higher-granularity access).

## 7 Synchronous Regions

Although the speed-of-light and device-size limitations ultimately appear to lead to machines whose structure on the large scale is of the (geometric) near-neighbor type (neighborly networks), the same limitations do not directly dictate the small and medium size structure of the machine. To obtain insight on some of the relevant factors, we analyze in more detail the fundamental features of communication as it occurs in digital systems.

Each communication action involves the transmission of a message from a source to a destination; the message consists of a number of bits (typically encoded as sequences of voltage transitions on suitable electric lines) which may be transmitted serially, in parallel, on in combination of thèse two basic modes. At the most elementary level, transmission occurs from driving gate to driven gate within a boolean network, from the instant the driving gate begins to switch, to the instant the driven gate begins an analogous transition. At a less elementary level, we have a memory read-out which runs from the instant a memory cell is selected to the instant the read-out information is usable at the output of some CPU register, etc...

In the above examples, as in any other, we recognize the following common features: an initial constant-duration set-up event (for example the switching of a driving transistor) is followed by a propagation event, which takes place in a selected support medium, and whose duration is an increasing function of the source to destination distance (for example, charging the capacity of the wire connected to the input gate of a driven transistor [BPP82].)

In all cases, it appears that the duration $\delta$ (delay) of a transmission event can be expressed as the sum of two terms: a constant set-up duration $\sigma$ and a variable transmission time. Conveniently, the latter can be approximated as a linear function of the travelled distance, so that

$$\delta = \sigma + \alpha\ell. \tag{2}$$

This general form of the communication delay has some fundamental implications for parallel computation. Note first that, as long as the two terms of the right-hand side of (2) are comparable, the distance $\ell$ plays a negligible role in the transmission delay $\delta$, to the point that for $\ell \leq \sigma/\alpha$ we may assume $\delta$ to be a constant. This indicates that within layout regions of geometric diameter $\sigma/\alpha$ distance becomes inconsequential: we may appropriately call such regions "synchronous", and they are correctly encompassed by the traditional synchronous model of VLSI. One should not naively infer, however, that purely technological parameters determine the size of synchronous regions. Architectural considerations may suggest a substantial enlargement of such domains by externally forcing the beginning and the end of communication events within the regions by means of a clock. Nevertheless, this approach typically introduces a slowdown with respect to the inherent speed of the deployed technology, which, beyond some limit, would offset any advantages accruing from architectural innovation. In consideration of such diverse factors, for the purposes of this paper we shall say that a *synchronous region is* implicitly defined as a layout domain where communication is allowed to take a single clock cycle.

Therefore, in the synchronous regions, the limitations that asymptotically dictate the geometric near-neighbor connection are not active constraints, and the interconnection topology is determined by other factors. Indeed, the opportunity arises to accelerate information transfer by adopting a low-diameter topology. However, if wire delay can be neglected, wire area remains an important factor and essentially constrains the available bandwidth across given sections of the machine. For example, the hypercube and its derivatives, whose bisection bandwidth is linear (or slightly sublinear) in the number of nodes, have quadratic (or nearly quadratic) area [Tho80]. Hence, within the fixed areas of the synchronous regions, only a number of nodes proportional to (or slightly larger than) the square root of that area can be connected according to those topologies. Therefore, the available area would be vastly underutilized by algorithms that can take more advantage of number of processors than they can do of bandwidth (e.g., consider systolic matrix multiplication).

Motivated by considerations of the type developed in the above paragraph, Leiserson [Lei85] proposed the concept of *area-universal* network. Informally, an area-universal network is one that can route any collection of messages almost as efficiently as any other network with similar layout area. This property is clearly desirable for a general-purpose computer, which at different times will have to execute algorithms with different communication requirements. However, a priori, the very existence of area-universal networks is not an obvious matter. Leiserson demonstrated the existence of some such networks in [Lei85] and called them *fat-trees.* A number of subsequent investigations have been devoted to the subject [GL89, LM88, BB90, G90, BB93] with various findings. Exploiting the insights obtained over a decade of studies on the relation between bandwidth of networks and their layout area [Tho80, L81, V81, BL84, BP86], some networks have been designed that can solve any instance of the routing problem by losing at most some logarithmic factors (typically two) against any other network of similar area (even one designed specifically for that instance). These results are remarkable and have already influenced the design of some commercial machines [Letal92].

In conclusion, we reach the view of an "asymptotically" large machine as a square mesh interconnection of small machines, each approximately of the size of the synchronous regions, exhibiting (in addition to the mesh structure) area-universal topologies. In the next section we shall elaborate on the possible impact of this mesh interconnection of area universal modules on the programming of parallel machines.

## 8 Considerations on Programming Models

It is of interest to speculate what programming model might be appropriate for the type of machine structure imposed by the physical constraints. This subject deserves a deep and systematic investigation well beyond the extent of this section, which offers only some preliminary considerations.

A convenient programming environment is provided by the *shared-memory* model. As we have seen in Section 4, the mesh supports an efficient implementation of shared memory, through the mechanism of P-RAM simulation. When this approach is adopted, a $p$-processor mesh will execute, on the average, $O(\sqrt{p})$ operations per unit time. We have also argued that this performance cannot be improved by latency-hiding techniques. It is well-known, however, that a number of algorithms of great practical interest exhibit considerable locality with respect to the mesh interconnection, that is, most of the memory references generated by any such algorithm are destined to nodes that are quite close to their respective source nodes, and hence do not incur the worst-case latency.

To illustrate this point, consider the problem of multiplying two $\sqrt{p} \times \sqrt{p}$ matrices and the two following solutions: (1) A uniprocessor system (von Neumann machine) with a memory of size $\theta(p)$, executing the standard serial algorithm, involving $\theta(p^{3/2})$ memory accesses and additions/multiplications. Since the memory is laid out in area $\theta(p)$, each memory access will take time $\theta(\sqrt{p})$ on the average, resulting in total execution time $\theta(p^{3/2}.\sqrt{p}) = \theta(p^2)$. (2) A mesh-connected $\sqrt{p} \times \sqrt{p}$ network of processors, executing the well-known systolic algorithm for the problem [K82], which runs in time $\theta(\sqrt{p})$, performing $\theta(p)$ useful arithmetic operations per time step. This is essentially due to the fact that each arithmetic operation executed by a PE involves data stored in adjacent PE's. Thus, under the speed-of-light limitation, we have (without substantially altering the used area) a parallel solution using $p$ processors that improves over the time of the best sequential solution by a factor of $p^{3/2}$, and therefore *greater* than $p$. This result is at variance with the intuitive principle sometime called the Fundamental Law of Parallel Computation [S86], which arises in the synchronous model. The superlinear speedup is a combined effect of parallelism and locality. The comparison would be less dramatic if made against a uniprocessor with a hierarchical memory. Indeed, in the limiting technology, memory could be realized with access time $\theta(\sqrt{x})$ for the word with address $x$. In this model, matrix multiplication would take time $\theta(p^{3/2} \log p)$ [ACS87].

In general, up to an $O(\sqrt{p})$ factor in time could be gained in a mesh by minimizing the distance of references. Within the area-universal synchronous regions, the latency of individual references becomes less significant, as the distances are logarithmic. However, the $O(\sqrt{p})$ bisection bandwidth is still a bottleneck, and up to an $O(\sqrt{p})$ factor in time could be gained by an appropriate reduction of the bandwidth requirements on all sections of the network.

Unfortunately, the P-RAM style of programming does not allow control of data allocation and hence precludes the consideration of locality. Worse yet, all known methods to simulate P-RAMs on bounded-degree networks are based on memory allocations or maps (hashing in randomized approaches, and generalized expanders in deterministic ones) which randomize the correspondence between logical and physical address spaces, thereby destroying any locality implicit in the logical address space. To some extent, sophisticated compiler and runtime system techniques may help in the objective of exploiting locality, and this approach will no doubt receive much attention. However, in light of the preceeding considerations, it appears desirable that, beyond the support of shared memory, the adopted programming language should allow the programmer to take a more direct control of machine resources.

The need for realistic models that would give the programmer a good sense of parallel program performance has motivated a number of proposals [S86, ACS89, V90a, Cetal92]. When developing such models, a tradeoff must be faced between accuracy of performance estimates (suggesting that all relevant features of the machine ought to be modelled) and portability (suggesting that only features common to all target machines ought to be modelled) [B89]. We predict that the constraints posed by physical limitations on machine structure will simplify the above problem considerably, by reducing the meaningful class of target machines.

## 9 Conclusions

Technological progress in digital system engineering brings about a serious re-examination of the current models of parallel computing. In this paper, we have taken the extreme view of a technological situation where the ultimate physical limits represented by speed-of-light and device-size (under degree-boundedness) have been attained, and examined the consequences of such premises. These hypotheses appear to lead, naturally, to an asymptotically scalable parallel computing structure, consisting of a mesh-like neighborly interconnection of synchronous units, each of the latter being itself, for example, an area-universal network. While this conclusion appears to negatively constrain the feasible network choices, on the positive side it disposes in a sense of the topological chaos of potential computing structures, which represented a widely held view in past years. In addition, the convergence towards a "favorite" interconnection may provide the consensus model that would finally let hardware and software flourish autonomously.

We have also examined how the fundamental physical limitations challenge some principles and schemes that are quite natural within the currently prevailing synchronous model, such as latency masking and processor-time tradeoffs.

Of course, we do not naively suggest that the limit model discussed in this paper should govern near-term parallel computing. As history teaches us, the evolution towards the outlined situation is likely to be smooth and rich with engineering innovations. There is ample room for retention of the synchronous model for a long time, but we should be aware of what lies at the end of the road.

## References

[ACS87] Aggarwal,A., Chandra, A.K., Snir,M.: A model for hierarchical memory, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, (1987), 305-314

[ACS89] Aggarwal,A., Chandra, A.K., Snir,M.: On communication latency in PRAM computation, Proceedings of the 1st ACM Symposium on Parallel Algorithms and Architectures, (1989), 11-22

[AHMP87] Alt, H., Hagerup, T., Mehlhorn, K., Preparata, F.P.: Simulation of idealized parallel computers on more realistic ones, SIAM Journal on Computing, **16**, 5 (1987), 808-835

[B89] Bilardi, G.: Some observations on models of parallel computation. In Opportunities and Constraints of Parallel Computation, J.L.C. Sanz Editor, Springer-Verlag, (1989), 11-13

[BB90] Bay, P.E., Bilardi, G.: Deterministic on-line routing on area-universal networks. Proceedings of the 31$^{st}$ Annual Symposium on Foundations of Computer Science, (1990), 297-306

[BB93] Bay, P.E., Bilardi, G.: An area-universal vlsi circuit, Proceedings of the 1993 Symposium on Integrated Systems, March 1993.

[Be92] Bell, G.: A teraflop before its time, Comm. of the ACM, **35**, 8 (1992), 26-47

[BL85] Bennett, C.H., Landauer, R.: The fundamental physical limits of computation, Scientific American, (July 1985), 48-56.

[BL84] Bhatt, S.N., Leighton, F.T.: A framework for solving VLSI graph layout problems, Journal of Computer and Systems Sciences, **28**, 2 (1984), 300-342

[BP86] Bilardi, G., Preparata, F. P.: Area-time lower-bound techniques with application to sorting, Algorithmica, **1** (1986), 65-91

[BPP82] Bilardi, G., Pracchi, M., Preparata, F.P.: A critique of network speed in VLSI models of computation, IEEE Journal Solid-State Circuits, **17** (1982), 696-702

[B74] Brent, R.P.: The parallel evaluation of general arithmetic expressions, Journal of the ACM, **21**,2 (1974), 201-206

[CM85] Chazelle, B., Monier, L.: A model of computation for VLSI with related complexity results, Journal of the ACM, **32** (1985), 573-588

[C74] Cook, S.A.: An observation of time-storage tradeoff, JCSS, **9**,3 (1974), 308-316

[C81] Cook, S.A.: Towards a complexity theory of synchronous parallel computation, Enseign. Math. **27** (1981), 99-124

[CR73] Cook, S.A., Reckhow, R.A.: Time bounded random access machines, Journal of Comput. System Science, **7** (1973), 354-375

[Cetal92] Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., von Eicken, T.: LogP: Towards a realistic model of parallel computation, Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, May 1993

[F66] Flynn, M.J.: Very high-speed computing systems, Proc. IEEE **54**, 12 (1966), 1901-1909

[FS92] Feldman, J., Shapiro, E.: Spatial machines: a more realistic approach to parallel computation, Communications of the ACM, **35**, 10 (1992), 61-73

[G90] Greenberg, R.I.: The fat-pyramid: A robust network for parallel computation, In Dally W.J., editor, Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, (1990), 195-213

[GL89] Greenberg, R. I., Leiserson, C.E.: Randomized routing on fat-trees. In Micali, S., editor, *Randomness and Computation*, JAI Press, Inc., (1989), 345-374

[HB88] Herley, K., Bilardi, G.: Deterministic simulations of PRAMs on bounded-degree networks, Proceedings of the 26th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, (1988), 1084-1093

[KU88] Karlin, A.,Upfal, E.: Parallel hashing - an efficient implementation of shared memory, Journal of the ACM, **35**, 4(1988), 876-892

[KR90] Karp, R.M., Ramachandran, V.: Parallel algorithms for shared-memory machines, Handbook of Theoretical Computer Science: Algorithms and Complexity (J.v. Leeuwen, ed.) Elsevier-The MIT Press, 1990

[KRS90] Kruskal, C.P., Rudolf, L., Snir, M.: A complexity theory of efficient parallel algorithms, Theoretical Computer Science, **71** (1990), 95-132

[K82] Kung, H.T.: Why systolic architectures? Computer Magazine, **15**, 1 (1982), 37-46

[L75] Ladner, R.E.: The Circuit Value problem is logspace complete for P, SIGACT News **7**, 1 (1975), 18-20

[L91] Leighton, F.T.: *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*, Morgan Kaufmann, 1991

[L81] Leighton, F.T.: *Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 1981

[Letal92] Leiserson, C.E., Abuhamdeh, Z.S., Douglas D.C., Feynman C.R., Ganmukhi, M.N., Hill, J.V., Hillis, W.D., Kuszmaul, B.C., St. Pierre, M.A., Wells, D.S., Wong, M.C., Yang, S.W., Zak, R.: The network architecture of the Connection Machine CM-5, Proceedings of the $4^{th}$ Annual ACM Symposium on Parallel Algorithms and Architectures (1992), 272-285

[Lei85] Leiserson, C.E.: Fat-trees: Universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers, **C-34**,10 (1985), 892-900

[LM88] Leiserson, C.E., Maggs, B.M.: Communication-efficient parallel algorithms for distributed random-access machines. Algorithmica, **3** (1988), 53-77

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

[MV84] Mehlhorn, K., Vishkin, U.: Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories, Acta Informatica, **21**, 4 (1984), 339-374

[P77] Pease, M.C.: The indirect binary $n$-cube microprocessor array, IEEE Transactions on Computers, **C-26**, 5 (1977), 458-473

[P79] Pippenger, N.: On simultaneous resource bounds, Proceedings of the 20th IEEE Symposium Foundation of Computer Science (1979), 307-311

[PV81] Preparata, F.P., Vuillemin, J.: The cube-connected cycles: A versatile network for parallel computation, Communications of the ACM, **24**, 5 (1981), 300-309

[Ran87] Ranade, A.G.: How to emulate shared memory, Proceedings of the 28th Annual Symposium on the Foundations of Computer Science (1987), 185-192

[Ros81] Rosenberg, A.L.: Three-dimensional integrated circuitry. In H.T. Kung, B. Sproull, and G. Steele, editors, Proceedings of the CMU Conference on VLSI Systems and Computations, Computer Science Press (1981), 69-80

[Smi81] Smith, B. J.: Architecture and applications of the HEP multiprocessor system, Signal Processing IV, 298 (1981), 241-248

[S86] Snyder, L.: Type architectures, shared memory, and the corollary of modest potential, Annual Review of Computer Science, **1** (1986), 289-317

[Tho80] Thompson, C.D.: *A Complexity Theory for VLSI.* Ph.D. dissertation, Carnegie-Mellon University, August 1980

[UW87] Upfal, E., Wigderson, A.: How to share memory in a distributed system, Journal of the ACM, **34**, 1 (1987), 116-127

[V90a] Valiant, L.G.: A bridging model for parallel computation, Communications of the ACM, 33,8 (1990), 103-111

[V90b] Valiant, L.G.: General purpose parallel architectures, Handbook for Theor. Computer Science [J.v. Leeuwen, ed.], Elsevier-MIT Press, 1990

[V81] Valiant, L.G.: Universality considerations in VLSI circuits, IEEE Transactions on Computers, **C-30**, 2 (1981) 135-140

[Vi86] Vitanyi, P.M.B.: Nonsequential computations and laws of nature, Aegean Workshop on Computing, Proceedings. In Lecture Notes in Computer Science 227: VLSI Algorithms and architectures, Springer-Verlag, Berlin (1986), 108-120

[Vi88] Vitanyi, P.M.B.: Locality, communication, and interconnect length in multicomputers, SIAM J. on Computing, **17**, 4 (1988), 659-672.