Unbounded Hardware is Equivalent to Deterministic Turing Machines

Bernard CHAZELLE and Louis MONIER

Department of Computer Science
Carnegie-Mellon University

October 1981

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

Table of Contents

List of Figures

Abstract

The purpose of this paper is to investigate models of computation from a realistic viewpoint. We introduce the concept of *physical* computation as opposed to *functional* computation, and by referring to the laws of physics we study the basic criteria which a model of computation must meet in order to be realistic. With this formal apparatus, we define a very general, realistic model of planar, digital circuits, which allows for full parallelism. Actually, the assumption of planarity serves only practical purposes, and can be removed without altering our main results. We compare the complexity classes in this parallel model with those associated with sequential models such as the Deterministic Turing Machine (DTM) model. Our main result is that both models are space and time equivalent in the polynomial class. In particular, any circuit can be simulated in polynomial time on a DTM. One consequence is that unbounded hardware does not make NP-hardness tractable. We also address the issue of area-time tradeoffs and show that the area of a circuit can always be bounded by a polynomial function of the sequential time.

1. Introduction

Among the various models which have been defined to describe the behavior of digital computing devices, Turing machines and RAMs [Aho et al. 75] are the most commonly used, and stand out as best illustrating the essentially *functional* nature of these models. By this statement, we mean that the main assumptions in these models are based on the mathematical rather than physical nature of the computations. Informally these models are said to be *sequential* if the number of bits processed at each step is bounded by a constant.

With the advent of VLSI technology, other models have been introduced, which exploit the possibility of unbounded parallelism while trying to remain realistic. Previous work has led to computational schemes which fare significantly better than the corresponding sequential methods. For example, schemes have been proposed to perform complex operations in logarithmic time [Bentley 79, Brent and Kung 80a, Preparata and Vuillemin 78, Thompson 80a] or even to solve NP-complete problems in polynomial time [Mead and Conway 80]. Although we still believe that these circuits can be very efficient for small problems, we can show that they fail to have the expected asymptotic complexity, for the underlying models contradict basic laws of physics.

To remedy these flaws, a very general model of circuit has been recently proposed [Chazelle and Monier 81], which tries to incorporate fundamental physical constraints. In this paper we will give an equivalent formulation of this model in a form suitable for simulation purposes. We will use this canonical description to simulate any computing circuit on a Deterministic Turing Machine (DTM), from which we will prove the relation

Parallel Polynomial Time (Space) = Sequential Polynomial Time (Space).

One consequence of this equivalence is the dismissal of any scheme aimed at cracking NP-hard problems with use of high parallelism [Mead and Conway 80]. The physical nature of a realistic model also leads to take a new approach to the question of area-time tradeoffs. We will show that a circuit used to solve a problem P can always be assumed to have an area bounded by a polynomial in the sequential time required to solve P. Thus all area-time tradeoffs are only valid for a limited range, and more practically, increasing parallelism does not always help.

2. Parallel .vs. Sequential Models

The general model of physical computation which we will consider has been described in previous work [Chazelle and Monier 81]. We recall that it is a model for planar, digital computing devices, and that it is merely a refinement of former models [Brent and Kung 80b, Savage 79, Thompson 80b, Vuillemin 80]. This model, called *iterative*, adds the following important assumption: the propagation speed of information is bounded by a constant. We briefly sketch the main characteristics of the model.

- The information is *digital* (binary) and encoded by the value of a physical parameter at specified times and locations.

- A circuit computes a boolean function $(y_1,...,y_m) = F(x_1,...,x_n)$. The size of a problem is the total number of input and output bits.

- A circuit is a planar layout of a directed graph, where the nodes are finite-state-automata (FSA) and the edges are wires. The inputs and outputs of the FSAs are boolean values stored at the endpoints of the wires, and we can assume long wires to be decomposed into unit-length wires connected by nodes computing the identity function. This allows us to associate each wire with exactly one variable, and thus assume that it has unit bandwidth.

- Communicating information with the outside of the circuit takes place at special nodes called I/O ports and located on the boundary of the circuit.

- Both the area A and the time of computation T have quantized units, usually denoted by λ and τ. A node performs an operation in at least unit time τ, and it has an area at least λ^2. Similarly, wires have width at least λ, and they transmit information at bounded speed.

The iterative model described above is a physical parallel model, since an arbitrary number of bits is processed at any instant. We note its similarity with a cellular automaton, except for the I/O conventions, and we believe that this model is suited to describe any planar, digital, *physical* machine. From now on, we will refer to it as a "circuit model".

On the other side of the spectrum, functional models like Random Access Machine or Deterministic Turing Machine are said to be sequential since the number of bits modified at any time is always bounded by a constant. Simulation between sequential machines has been well-studied, and we wish now to extend this work to physical parallel models, i.e., compare the complexity of problems in a sequential and parallel model.

3. Simulating Parallel and Sequential Machines

We begin by showing how to simulate a circuit on a DTM. Our main result can be stated as follows.

Theorem 1: Any circuit solving a problem of size N in time T and area A can be simulated on a two-dimensional Turing machine in sequential time $T_s = O(NT^3)$, using the same area.

Proof: The crux of the argument relies on the bounded propagation speed of information. Consider the nodes holding "meaningful" information, which we call *active nodes.* In a circuit solving a problem of size N, no more than N nodes can be active when the computation starts, and after a time T, the information diffusing from the input ports can cover an area $O(NT^2)$ only, thus the number of nodes active at least once during the computation is $O(NT^2)$.

Next we show how to simulate each unit of parallel time on a two-dimensional Turing machine. The idea is to move the head of the machine on a planar structure which encodes the state of the circuit. Since each node has a number of inputs, outputs, and states bounded by a constant, it can be encoded on a fixed size square and simulated in constant time; similarly we can decompose the

wires in segments of unit length, encode each segment on a square and simulate each of them in constant time, as shown in Figure 1.

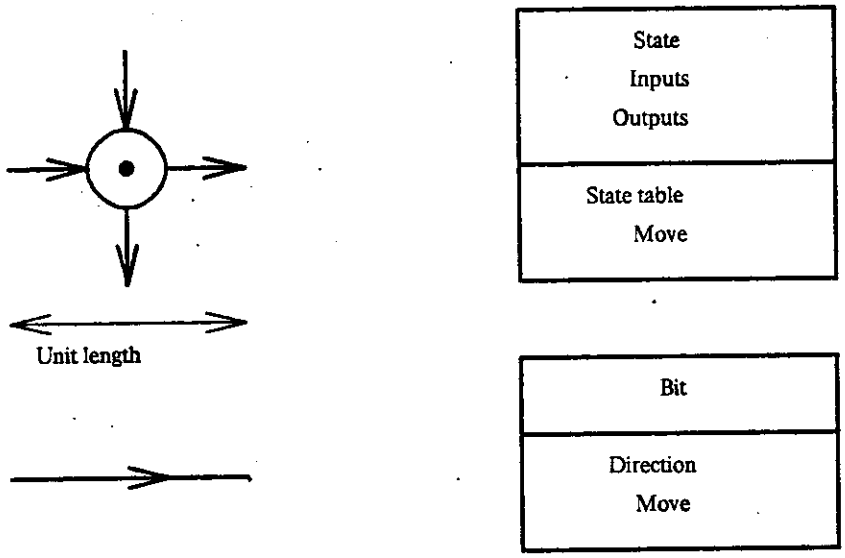

Figure 1: Encoding a circuit on a 2D Turing machine.

The only difficulty is to move the head efficiently. Since the active part of the circuit has area at most $O(NT^2)$, and since its location is known (inside circles centered at the input ports), a simple approximation of a traveling-salesman tour of all these nodes and wires can be used to route the head. So we can encode the tour on the 2-D tape of the DTM so that only local checking enables the head to update the tape and to know where to move. This simulation is shown in Figure 2.

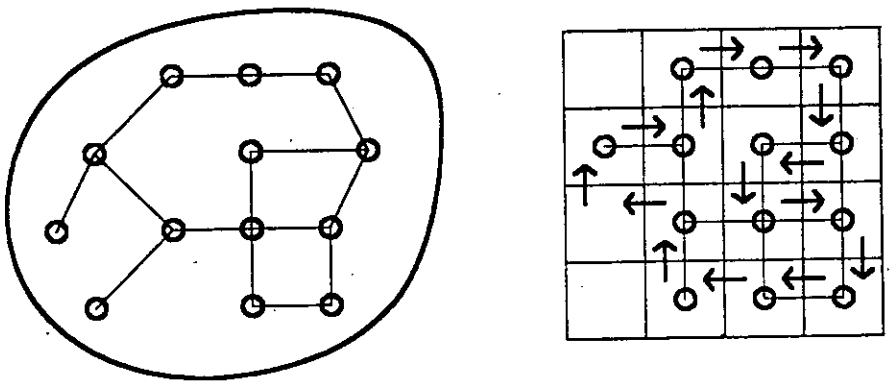

Figure 2: Simulation of a circuit on a DTM.

We conclude by noting that if the active part of a circuit is not connected (the circuit being thus artificially large), we can remove all inactive parts and move the active components close enough: since there is no empty space to traverse, the length of the tour is thus at most proportional to the number of active squares, that is, $O(NT^2)$.

We also observe that the area of the 2D tape actually used is proportional to the area A of the circuit.

Finally since every unit of parallel time can be simulated on the Turing machine in time $O(NT^2)$, the whole simulation takes at most a time $T_s = O(NT^3)$, which completes the proof. □

Next we bridge the gap between physical and functional computation by tackling the converse operation, that is, simulating a DTM on a circuit. We prove our result for one-track, one-head DTM.

Theorem 2: Any DTM with one track and one head which computes a function in time T with a tape of length L can be simulated on a circuit of area O(L) in time O(T).

Proof: In fact, we will show that a DTM is only a special instance of a circuit. To simulate a Turing machine, we can use a ladder-like circuit as shown in Figure 3, where one chain simulates the tape (memory), and the other is a duplication of the state control mechanism (processors).

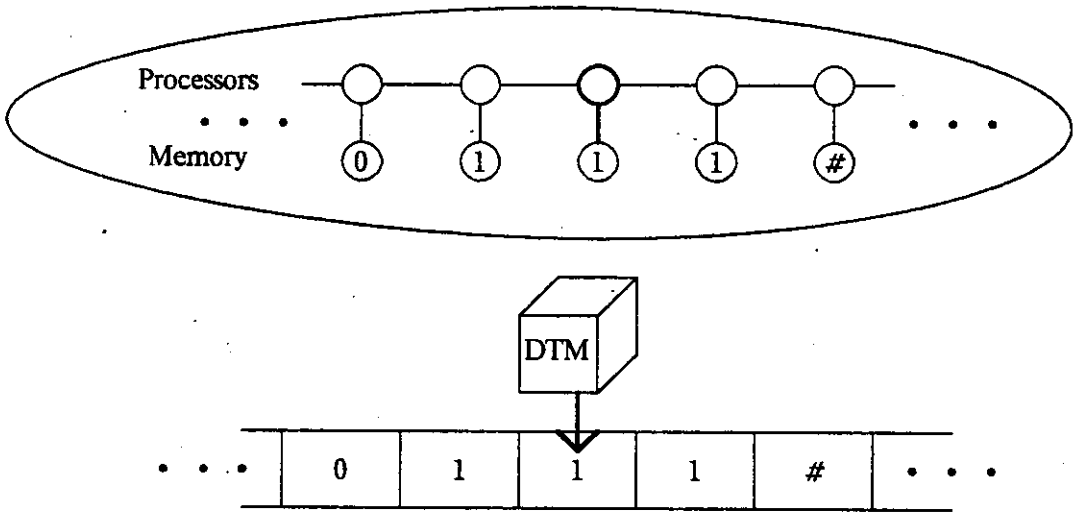

Figure 3: A circuit implementation of a DTM.

Each square of the tape is represented by a node able to memorize one symbol. Every such *square* is connected to a finite-state-automaton simulating the head of the DTM. At any moment, only one *head* (represented in bold-face on Figure 3) is active: it reads the symbol, changes its state, writes another symbol and *moves*, i.e., copies its state into the appropriate neighbor, which then becomes active. The initially non-blank portion of the tape is first loaded through input ports in unit time, and the result of the computation is output in the same way. □

It turns out that we can describe any *realistic* digital (planar) machine as a circuit: a one-dimensional or two-dimensional Turing machine, a Von Neumann machine, a vector machine or a cellular automaton are all mere instances of planar circuits. However, the model of VLSI circuit defined originally in [Thompson 80b] and used for small circuits is not equivalent to our model, since it neglects the cost of information transmission, and is thus not realistic from an asymptotic point of view.

The physical nature of a circuit is bound to frustrate many hopes and kill grandiose plans: tree-schemes for performing computations in logarithmic time, or trivial brute-force methods to attack NP-hard problems should no longer be sought. Actually, the use of high parallelism does not change the classes of complexity, and no circuit can implement a non-deterministic Turing machine, using an exponential number of processors in polynomial time.

4. Relation between area and time

A well-known paradigm concerns the area-time trade-offs: the larger the circuit, the shorter the time. We will show that this is not always true, and that for any problem there exists a maximal size of circuit: over that size, a circuit becomes slower than a sequential machine (DTM for example). For simplicity, we will omit constant factors in this section.

Theorem 3: Consider a problem of size N solvable in time T_s on a sequential machine. If a circuit using k processors is able to solve the same problem in time $T \leq T_s$, we must have $k \leq NT_s^2$, and the area of the circuit can be at most $A = N^2T_s^2$.

Proof: As a consequence of the bounded speed for propagating information, in time T, no more than NT^2 nodes can be active, which imposes a great limit to the number of processors actually used during the computation. The bound on the area is a consequence of the convexity of circuits. In order to maximize the area, the N input ports can be allowed to lie on the boundary of (say) a square, and since the distance between two consecutive ports cannot exceed T without obvious waste of space, the area is $O(NT)^2$, hence $O(N^2T_s^2)$. □

This result may seem somewhat paradoxical, but it simply states that for a physical machine, there exists a relation between the area and the computation time.

5. Extensions

The results we have shown are mostly theoretical. Although we may legitimately claim that the model used to describe a circuit is realistic and consistent, it is still a model and only a model, that is, a framework which idealizes the behavior of a computing device. We must keep in mind that all circuits and computers actually built are *small*, and therefore asymptotic analysis is not suited to give a faithful account of their behavior. Since the parameters used in practice (i.e., size of a problem, area, time) lie in a relatively small range of values, parasitic effects may become predominant: for example it is possible that tree-based schemes yield computation times proportional to the logarithm of the problem size within a certain range. The question is whether or not this range is large enough to cover all real problems, in which case our asymptotic model may become too restrictive to give good estimates of the circuit performance. If we are interested in asymptotic results, however, we must take great care in choosing the model: one suited only for approximations could lead to aberrant results.

Another point of discussion concerns the planarity assumption. We have described a model of planar circuit, mainly because present technologies restrict us to such circuits. This situation may however change in the near future. We must then be aware of one important parameter which should be included in a three-dimensional model: the energy. If we assume that a node changing its state uses one unit of energy, we must be ready to face the problem of energy dissipation. For example a mesh of nodes continually active uses an energy proportional to its volume, but can dissipate an energy which is at most proportional to its area. There follows a limit on the size of such circuits. Thus, new constraints may appear in a realistic model of three-dimensional circuits.

It is however simple to extend our model to three dimensions. It will actually give similar results. For example, a 3D-circuit could be simulated in time $O(NT^4)$ on a three-dimensional Turing machine using the same amount of space, and in fact, our main result still holds, i.e., any circuit -even three-dimensional- is equivalent to a DTM in the polynomial class.

6. Conclusions

We have designed a model of computation for digital machines that does not violate the laws of physics. Compared to previous models for unbounded hardware (e.g. VLSI), we added only the assumption that the speed of information propagation is bounded by a constant. This is sufficient to cause major modifications into previous results, since any physical computing machine actually behaves like a cellular automaton. The main contribution of this paper has been to show that any *realistic* model of digital machine is polynomially equivalent to any sequential machine (e.g., DTM). As a consequence, NP-hard problems remain intractable, even by using an unbounded amount of hardware.

References

[Aho et al. 75] Aho, A., Hopcroft, J.E. and Ullman, J.D.
The Design and Analysis of Computer Algorithms.
Addison-Wesley, Reading, Massachusetts, 1975.

[Bentley 79] Bentley, J.L.
A Parallel Algorithm for Constructing Minimum Spanning Trees.
Technical Report, Carnegie-Mellon University, Department of Computer Science, August, 1979.

[Brent and Kung 80a]

Brent, R.P. and Kung, H.T.

The Chip Complexity of Binary Arithmetic.

In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, pages 190-200. ACM, April, 1980.

[Brent and Kung 80b]

Brent, R.P. and Kung, H.T.

The Area-Time Complexity of Binary Multiplication.

Journal of the ACM (to appear):, 1980.

Also available as a CMU Computer Science Department technical report, July 1979.

[Chazelle and Monier 81]

Chazelle, B.M. and Monier, L.M.

A Model of Computation for VLSI with Related Complexity Results.

In *Proceedings of the 13th Annual ACM Sysmposium on Theory of Computing.* ACM SIGACT, May, 1981.

[Mead and Conway 80]

Mead, C.A. and Conway, L.A.

Introduction to VLSI Systems.

Addison-Wesley, Reading, Massachusetts, 1980.

[Preparata and Vuillemin 78]

Preparata, F. and Vuillemin, J.

The Cube-Connected-Cycles: A Versatile Network for Parallel Computation.

In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 140-147. IEEE, 1978.

[Savage 79] Savage, J.E.

Area-Time Tradeoffs for Matrix Multiplication and Related Problems in VLSI Models.

Technical Report CS-50, Brown University, Department of Computer Science, August, 1979.

[Thompson 80a] Thompson, C.D.

Fourier Transforms in VLSI.

In *Proceedings of the 1980 IEEE International Conference on Circuits and Computers.* IEEE, 1980.

[Thompson 80b] Thompson, C.D.
A Complexity Theory for VLSI.
PhD thesis, Carnegie-Mellon University, Department of Computer Science, 1980.

[Vuillemin 80] Vuillemin, J.
A Combinatorial Limit to the Computing Power of V.L.S.I. Circuits.
In *Proc. 21st Annual Symposium on Foundations of Computer Science*, pages 294-300. IEEE, October, 1980.