Parallelizing Time With Polynomial Circuits

Ryan Williams*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
ryanw@cs.cmu.edu

ABSTRACT

We study the relatively old problem of asymptotically reducing the runtime of serial computations with polynomial size Boolean circuits. To the best of our knowledge, no progress on this problem has been formally reported in the literature for general computational models, although we observe that early work of Chandra, Stockmeyer, and Vishkin implies the existence of non-uniform unbounded fan-in circuits of $t^{O(1)}$ size and $O(\frac{t}{\log\log n})$ depth, for time t Turing machines.

We give an algorithmic size-depth tradeoff for parallelizing time t random access Turing machines, a model at least as powerful as logarithmic cost RAMs. Our parallel simulation yields logspace-uniform $t^{O(1)}$ size, $O(t/\log t)$ depth Boolean circuits having semi-unbounded fan-in gates. In fact, for appropriate d, uniform $t^{O(1)}2^{O(t/d)}$ size circuits of depth $O(d)$ can simulate time t. One corollary is that any log-cost time t RAM can be simulated by a log-cost CRCW PRAM using $t^{O(1)}$ processors and $O(t/\log t)$ time. This is a major improvement over previous parallel speedups, which could only guarantee an $\Omega(\log t)$ speedup with an exponential number of processors.

Categories and Subject Descriptors

F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Alternation and nondeterminism, Parallelism and concurrency*; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs between Complexity Measures

General Terms

Algorithms, Theory

Keywords

circuit complexity, alternation, parallel speedup

*Supported by the NSF ALADDIN Center (NSF Grant No. CCR-0122581).

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

1. INTRODUCTION

A fundamental problem in Complexity Theory is to determine the extent to which arbitrary time-bounded computations can be parallelized. For example, the $\mathsf{NC} = \mathsf{P}$ question asks if polynomial time computations can be captured by $(\log n)^{O(1)}$ depth, bounded fan-in, polynomial size circuits. Over the years, numerous parallel simulations have been given for speeding up general (serial) computational models [13, 14, 3, 9, 10, 8], each of which can be interpreted as a Boolean circuit family, where the number of processors in the parallel algorithm is polynomially related to the circuit size of the family. However, no known parallel simulation achieves an $\omega(1)$ speedup (*i.e.* circuits of depth $o(t)$ for a time t serial algorithm, respectively) with only a polynomial number of processors (gates, respectively), cf. [4, p.79]. On the other hand, Meyer auf der Heide [11] has shown that restricted RAMs (lacking a full set of operations and indirect addressing) can indeed be sped up by a $\log t/(\log\log t)^2$ factor using $t^{O(1)}$ processors, in a similarly restricted PRAM model.

In 1985, after presenting various speedups of deterministic Turing machines on parallel machines with an exponential number of processors, Dymond and Tompa [3] concluded that it would "be interesting to know what speedups can be achieved using a number of processors that is only polynomial in the time bound." While this is certainly an interesting theoretical question, it could also be of practical import, perhaps more so than parallel simulations which obtain a faster speedup but require exponentially many processors.

The only result we could find directly addressing this question is from Mak [10] in 1997, who shows a time t RAM can be sped up to $\varepsilon t + n$ by a CREW PRAM of $t^{O(1)}$ processors, for $\varepsilon > 0$.[1] However, a scour of the literature revealed that work of Chandra, Stockmeyer, and Vishkin [2] yields non-uniform circuits of $o(t)$ depth and polysize for time t Turing machines. They prove that any bounded fan-in circuit of size $n^{O(1)}$ and depth $d \geq \log n$ has an equivalent *unbounded* fan-in circuit of $n^{O(1)}$ size and depth $O(d/\log\log n)$.

Briefly, [2] observe that a bounded fan-in circuit with depth $\varepsilon \log\log n$ and a single output gate can be seen as a Boolean function taking $O((\log n)^{\varepsilon})$ inputs, which in turn can be represented by an unbounded fan-in depth-2 circuit (in particular, a DNF) of size $2^{O((\log n)^{\varepsilon})}$. Every $\varepsilon \log\log n$ levels of the given depth d circuit can therefore be replaced with a collection of $2^{O((\log n)^{\varepsilon})}$ size, depth 2 circuits. This

[1] Note that unit cost RAMs do not have a linear speedup theorem like Turing machines.

results in an unbounded fan-in circuit of polynomial size and $O(d/\log\log n)$ depth, albeit one that is non-uniform. As it is well-known that time t Turing machines can be simulated by bounded fan-in circuits of depth $O(t)$ and size $O(t^2)$, the above transformation results in a simulation using $O(t/\log\log n)$ depth, unbounded fan-in polynomial circuits.

THEOREM 1. *Let L be a language recognized by a Turing machine using time $t(n)$. Then L is recognized by a circuit family having unbounded fan-in, $t^{O(1)}(n)$ size, and $O(t/\log\log n)$ depth.*

To obtain a more general, uniform circuit construction of smaller depth, we take a different approach. We show how to simulate time t random access Turing machines with uniform $t^{O(1)}2^{O(t/d)}$ size, $O(d)$ depth circuits of semi-unbounded[2] fan-in, where d is a parameter obeying certain constraints. Setting $d = t/\log t$ implies polynomial size circuits of depth $O(t/\log t)$. We also show how the construction implies a simulation of logarithmic cost RAMs on logarithmic cost CRCW PRAMs using $t^{O(1)}$ processors and $O(t/\log t)$ time.

2. PRELIMINARIES

We use the definition $[n] := \{1, \ldots, n\}$. We assume familiarity with basic concepts of complexity theory, though we will briefly review some notions required for this paper. Throughout, we implicitly assume all functions considered are those efficiently constructible under the appropriate resources.

Circuit Uniformity. In this paper, our circuit constructions will be *logspace-uniform*. A circuit family $\{C_n\}$ is logspace-uniform if there is a Turing machine M that, given 1^n as input, writes a description of C_n on its output tape. Furthermore, $M(1^n)$ runs in $O(\log n)$ space.

Deterministic Machine Model. Our main result simulates *random access Turing machines*. These are Turing machines with a fast access mechanism. This machine model is fairly powerful in that such machines can simulate logarithmic cost RAMs with only a constant factor overhead in runtime [15]. A random access TM has, along with the usual Turing machine equipment, $k+1$ write-only binary *index* tapes, one for the input and k for the worktapes. To access the ith cell of the input or a worktape, one writes i in binary on the respective index tape (in $O(\log i)$ steps, if the index is blank), and switches to a special "access" state which moves the respective head to the ith cell of the respective tape in one timestep.

2.1 Background on Alternation

It will be convenient to describe circuits with alternating machines, a natural model of parallelism [1].The reader may refer to Papadimitriou [12] for definitions.

$\mathsf{ATIME}[t]$ and $\mathsf{ASPACE}[s]$ denote the classes of alternating machine time t and space s, respectively; $\mathsf{ATISP}[t,s]$ denotes the sets accepted by alternating machines using both time t and space s simultaneously. $\Sigma_a\mathsf{SPACE}[s]$ denotes the class of sets accepted by alternating machines taking at most a alternations and $O(s)$ space in every branch. We use the following relation between alternating computations and circuits, sketching its proof for completeness.

[2]Confer with Section 2.1 for a definition.

THEOREM 2 (GENERALIZES [16] AND [7]). *For $a(n) \geq s(n) \geq \log n$, any language in $\Sigma_{a(n)}\mathsf{SPACE}[s(n)]$ is accepted by an unbounded fan-in, logspace-uniform circuit family of $O(a(n))$ depth and $a(n)\cdot 2^{O(s(n))}$ size.*

PROOF. The below proof is due to Immerman [7, p.87]. Let M be a $s(n)$ space machine making $a(n)$ alternations. Let C_1, C_2 denote configurations of M on an n-bit input. Define $\mathrm{E}(C_1, C_2, x)$ to be a circuit that outputs 1 iff from C_1 there is a computation on input x that reaches C_2 through existential states only. Similarly define $\mathrm{A}(C_1, C_2, x)$ for universal states. E and A accept languages in $\mathsf{NSPACE}[s(n)]$ and $\mathsf{coNSPACE}[s(n)]$, respectively, and thus can be constructed uniformly, with unbounded fan-in, $2^{O(s(n))}$ size, and $O(s(n))$ depth. Let I and Ac be the unique initial and accepting configuration for M on n-bit inputs. Now define a circuit ACCEPT via the following inductive definition:

$$\begin{aligned}\mathrm{ACCEPT}(C_1, i) \quad := \quad & ((C_1 = Ac) \wedge (i = 1)) \vee \\ & (\exists C_2)[\mathrm{ACCEPT}(C_2, i-1) \wedge (\mathrm{E}(C_1, C_2) \vee \mathrm{A}(C_1, C_2))].\end{aligned}$$

It is clear that $\mathrm{ACCEPT}(I, a(n))$ is true iff $M(x)$ accepts. Moreover, the resulting circuit is extremely regular, even logspace-uniform. Its size is dominated by the guessing of C_2, which requires $2^{O(s)}$ gates for each $i = 1, \ldots, a(n)$. □

Semi-unbounded Fan-in Circuits. A *semi-unbounded fan-in* circuit family has circuits where only OR gates are unbounded (AND gates have bounded fan-in). In the uniform setting, semi-unbounded and unbounded fan-in can make a difference; cf. Vollmer [17]. We note that the above theorem can be tweaked to get a result for semi-unbounded circuits, which we will use later.

COROLLARY 1. *For $a(n) \geq s^2(n)$ and $s(n) \geq \log n$, any language in $\Sigma_{a(n)}\mathsf{SPACE}[s(n)]$ is accepted by a uniform circuit family of semi-unbounded fan-in, $O(a(n))$ depth, and $2^{O(s(n))}$ size.*

PROOF. Circuits E and A can be made to have bounded fan-in, size $2^{O(s(n))}$, and depth $O(s^2(n))$. Then the circuit for ACCEPT is only unbounded in ORs: namely, in the choice of C_2. □

It follows that, to prove the claim of the introduction, it suffices for us to show $\mathsf{DTIME}[t] \subseteq \Sigma_{\frac{t}{\varepsilon \log t}}\mathsf{SPACE}[\varepsilon \log t]$.

2.2 Review of CRCW PRAMs

We assume a RAM model with the usual set of operations. We will work with the Concurrent Read, Concurrent Write (CRCW) PRAM. In such a model, we have a collection of RAM processors $P_1, \ldots, P_k, \ldots$ running a common program in parallel; the only difference between the processors is an instruction that allows P_i to load its own number i into a register. Each processor has its own local memory, and they all share a global memory. Global memory locations can be concurrently read and concurrently written, in that when more than one processor wants to write to the same location, we imagine a memory control that simply allows the processor of lowest index to write and ignores the rest (this is sometimes called the PRIORITY CRCW model).

Two measures of time are typically used for RAMs and PRAMs: unit cost and logarithmic cost. In the unit cost measure, every instruction takes one unit of time. Such a model can be "abused" in that it permits, for example,

the addition of arbitrarily large integers in constant time, provided they are already present in memory. We use the logarithmic cost measure, which is more realistic but context sensitive: every instruction on integers i and j takes $\lceil \log i \rceil + \lceil \log j \rceil$ units of time.

3. MAIN RESULT

We shall now prove that uniform semi-unbounded fan-in circuits of $t^{O(1)}2^{O(t/d)}$ size $O(d)$ and depth can simulate time t, for appropriately chosen d. The main idea is to generalize the well-known result for Turing machines that $\mathsf{DTIME}[t] \subseteq \mathsf{ASPACE}[\log t]$, *i.e.*, deterministic time is in alternating logspace [1]. This is a departure from all other parallel speedups of deterministic time that we are aware of, which instead build upon the proofs that $\mathsf{DTIME}[t] \subseteq \mathsf{ATIME}[o(t)]$ (cf. [14, 3]) or $\mathsf{DTIME}[t] \subseteq \mathsf{SPACE}[t/\log t]$ (cf. [6, 15, 5]), for a variety of computational models.

In the proof of $\mathsf{DTIME}[t] \subseteq \mathsf{ASPACE}[\log t]$, a time t one-tape Turing machine M is simulated by an alternating machine that runs M "backwards", one step at a time. More precisely, the simulation starts in the accepting state of M and the tape head at the leftmost tape cell (presumed to be blank) at timestep t. It then existentially guesses the content of adjacent cells and the transition taken in the previous timestep, then universally verifies each cell guess recursively. When translated to Boolean circuits, this construction yields circuits of $t^{O(1)}$ size and $O(t)$ depth.

To reduce depth, we can guess several transitions at a time, in blocks. In parallel, we can verify the correctness of transitions within the current block, as well as chronologically earlier blocks that affect the current block. By properly choosing the block size and increasing the fan-in of gates, we simultaneously reduce the circuit depth and increase the size by at most polynomial. With some care, this approach allows us to account for random accesses as well as sequential ones.

THEOREM 3. *For $t(n) \geq n$ and $a(n)$ such that $\log n \leq a(n) \leq t(n)/\log t(n)$,*

$$\mathsf{DTIME}[t(n)] \subseteq \Sigma_{O(a(n))}\mathsf{SPACE}\left[\frac{t(n)}{a(n)} + \log t(n)\right].$$

PROOF. Without loss of generality, we suppose $a(n)$ divides $t(n)$. For simplicity, assume the given random access TM M has only one read-write tape which initially contains the input and $t(n)$ blanks. The proof can easily be extended to any (finite) number of random access tapes.

Define the *local action* of M on input x at step $i \in [t]$ to be (i, r, I), where r is the transition taken and I is the index tape content of $M(x)$ at step i.[3] Clearly, a local action takes at most $O(\log t)$ bits to describe. Let $S_{M,x}$ be the unique string of the form

$$\ell_0\ \vec{r}_0\ \ell_1\ \vec{r}_1\ \cdots\ \ell_{a(n)-1}\ \vec{r}_{a(n)-1},$$

where ℓ_i is the local action of $M(x)$ at step $i \cdot t(n)/a(n) + 1$, and $\vec{r}_j$ is the vector of the next $t(n)/a(n) - 1$ transitions taken by $M(x)$ starting at step $j \cdot t(n)/a(n) + 2$. By assumption on $a(n)$, $|S_{M,x}|$ is $O(a(n)\log t(n) + t(n)) = O(t(n))$ bits. Define a *block* to be a substring of $S_{M,x}$ of the form $\ell_i\ \vec{r}_i$. Our alternating machine A to simulate M will essentially reconstruct $S_{M,x}$, one block at a time. The machine A uses two recursive procedures, VERIFY and LAST-WRITE, with the specifications:

- VERIFY(b, i) accepts iff b is the ith block of $S_{M,x}$, and
- LAST-WRITE(I, i, σ) accepts iff σ is written in the chronologically last timestep where I is the index tape, over timesteps $1, \ldots, i \cdot t/a$.

The procedures contain a number of deterministic checks, each of which are implementable in logarithmic space. We presume if any check fails that the machine *rejects* the current branch. We also presume that a recursive call erases all worktape content, except for $t(n)$ and $a(n)$ (written in binary), and the relevant arguments for the call.

Machine A on input x initially sets a variable i to $a(n)$, **existentially** writes b to tape, and calls VERIFY(b, i).

VERIFY(b, i): Let $b = \ell\ \vec{r}$.

If $i = a(n)$, **check** that the last transition of $\vec{r}$ leads to an accept state.

If $i = 1$, *accept* iff ℓ starts in the initial state, and all symbol reads in b are either blanks, appropriate bits of x, or symbols previously written in b.

Assume that the symbols read in $\vec{r}$ are correct, and that ℓ is correct. Under these assumptions, **check** that all state changes, symbols written, and index tape modifications in b are correct, *i.e.* the changes correspond to transitions of M.

Verify the pieces of the block in parallel:

Universally choose $j \in [t(n)/a(n)]$.

Verify ℓ is correct:

If $j = 1$, let q, σ, and I be the state, symbol read, and index tape of ℓ, respectively. **Universally**:

- *Check the last write to the Ith position was σ*
  Call LAST-WRITE$(I, i-1, \sigma)$ **AND**
- *Check q and I are correct*
  Existentially guess block b_{i-1}. **Check** that the last transition of b_{i-1} changes the state and index tape to the state and I in ℓ. Call VERIFY$(b_i, i-1)$.

Verify the correctness of symbols read in $\vec{r}$:

If $j > 1$, let σ be the symbol read in transition $\vec{r}[j-1]$. **Existentially** guess the index tape I' for the timestep corresponding to transition $\vec{r}[j-1]$. **Check** I' is correct by simulating the index tape starting from ℓ, using the transitions $r[1], \ldots, r[j-2]$. Along the way, **check** if I' is the index tape for more than one timestep during this simulation.

If it is, **check** that σ was written in the last such occurrence of I' prior to $\vec{r}[j-1]$.

If it is not, call LAST-WRITE$(I', i-1, \sigma)$.

That completes VERIFY. Now we describe LAST-WRITE.

LAST-WRITE(I, i, σ):

If $i = 1$, then *accept* iff σ is a blank or the appropriate input symbol of x.

Existentially guess block $b_i = \ell_i\ \vec{r}_i$.

Universally:

- Call VERIFY(b_i, i) **AND**
- **Check** if the index tape is ever equal to I, when simulating the index tape starting from ℓ_i via the transitions in $\vec{r}_i$. If so, *accept* iff σ is written in the last transition such that I is the index tape. Otherwise if the index tape is never I, call LAST-WRITE$(I, i-1, \sigma)$.

[3] In the general case where M has k tapes, a local action would contain k index tape contents, *e.g.* $I_1, \ldots, I_k$.

This completes the description of A. We now argue that A executes within the desired resources.

It is routine to verify that each deterministic **check** in VERIFY and LASTWRITE can be performed in deterministic $O(\log t(n))$ space, as each one maintains at most $O(\log t(n))$ bits of information in its simulation of a block. Therefore these checks affect neither the number of alternations nor the overall space bound of A.

The number of alternations used by A can be discerned by induction on i. Observe each call of VERIFY$(\cdot, i)$ depends only on VERIFY$(\cdot, j)$ and LAST-WRITE$(\cdot, j, \cdot)$ calls, where $j < i$. A similar observation holds for LAST-WRITE. Inspection shows that, between every two consecutive calls of VERIFY or LAST-WRITE, at most a constant number of alternations occur. It follows by induction that VERIFY(b, i) and LAST-WRITE(I, i, σ) use at most $O(i)$ alternations.

The proofs of correctness of VERIFY and LAST-WRITE are similarly straightforward, by induction on i. Recall that the string of local actions $S_{M,x}$ is unique for each x. Note when a block is guessed and applied, we recursively verify its correctness in a separate branch. It follows that every block b_i guessed in an accepting computation is the unique ith block within $S_{M,x}$. □

The following is immediate from Corollary 1 and the above theorem.

COROLLARY 2. *For $a(n)$ and $t(n)$ such that $t(n) \geq n$ and $t(n)/\log t(n) \geq a(n) \geq (t(n)/a(n)+\log t(n))^2$, deterministic time t random access Turing machines can be simulated by uniform $t^{O(1)} \cdot 2^{O(t/a)}$ size circuits of semi-unbounded fan-in and depth $O(a(n))$.*

Hence, there exists a spectrum of size-depth tradeoffs for simulating time with semi-unbounded circuits. In particular, semi-unbounded fan-in circuits of polysize and depth $O(t/\log t)$ are possible. Hence, a PRAM model where massive OR-parallelism is cheap but AND-parallelism is expensive would still be capable of performing our simulation.

3.1 Carrying out the simulation on CRCW PRAMs

COROLLARY 3. *The simulation of Theorem 3 can be performed by a logarithmic cost CRCW PRAM in $O(t/\log t)$ time. Therefore every log-cost time t RAM can be simulated by a log-cost CRCW PRAM in $O(t/\log t)$ time with only $t^{O(1)}$ processors.*

PROOF. (SKETCH) Our proof is in the spirit of Stockmeyer and Vishkin [16, Theorem 2]. Make a processor $P_{i,j}$ for each wire (i, j) in the circuit from Theorem 3 simulating a given time t RAM, where i is the gate of lesser depth (the longest path from an input bit to i is less than the longest path from an input to j).[4] Gate i in the circuit will correspond to the ith global memory location g_i. All g_is are all initially blank, except for those corresponding to input bits.

At its start, $P_{i,j}$ loads i and j into registers and determines the depth d of i. (To ensure this can be done quickly, we may encode i such that d is a prefix in i's binary encoding.) Next, $P_{i,j}$ computes $\log d$, and with it, $d/\log d$ as well. Observe these can be computed from a binary representation of d in $(\log d)^{O(1)}$ time. Then $P_{i,j}$ counts up to $d/\log d$, by repeatedly decrementing 1 from the register that initially holds $d/\log d$. The point is that this decrementing stage takes $\Theta(d)$ time on a logarithmic cost RAM[5], so that $P_{i,j}$ waits just long enough (within a constant factor) for an answer to arrive at memory location g_i.

Without loss of generality, assume gate i is an OR and gate j is an AND—the proof is symmetric for the other three cases. After the $\Theta(d)$ time delay, $P_{i,j}$ checks if a 1 was written to g_i. (If gate i was instead an AND gate, $P_{i,j}$ would have checked for a blank.) If a 1 was written, $P_{i,j}$ writes nothing to g_j. (If j was an OR, $P_{i,j}$ would have written a 1.) Arguing inductively, we find that that no write occurs to (the OR gate) g_i if and only if gate i gets set to 0 in the circuit evaluation, and no write occurs to (the AND gate) g_j if and only if gate j gets set to 1. NOT gates are handled in the obvious way.

By the uniformity of the circuit simulation, the above is doable by a CRCW PRAM of logarithmic cost where every processor runs in time $O(t/\log t)$. □

It is instructive to compare the above corollary to Mak [9], who showed that every log-cost time t RAM can be simulated by an alternating log-cost RAM in $O(t \log\log t/\log t)$ time. His proof mimics ideas from the time versus space work of Halpern *et al.* [5] on pointer machines. However, like all other work we have found in this direction, his simulation requires a superpolynomial number of processors, hence the resulting circuit requires at least this many gates.

4. CONCLUSION

We have of course given only a modest partial answer to the problem of parallelizing deterministic time with polynomial circuits. The obvious next step would be to achieve bounded fan-in circuits with similar size-depth parameters, if possible. This would imply, among other things, that the simulation can also be performed with a CREW (Concurrent Read, Exclusive Write) PRAM. As our simulation is quite different from past work, it is possible that some combination of our ideas with a simulation based on DTIME$[t] \subseteq$ ATIME$[t/\log t]$ could yield a better size-depth tradeoff.

5. ACKNOWLEDGEMENTS

The author appreciates the very useful commentary from anonymous referees for SPAA.

[4] In order to uniformly make each processor without affecting runtime, the simulating circuit needs the property that, given the pair (i, j), it can be determined if (i, j) is a wire in $O(t/\log t)$ time. Proving the circuit of Theorem 3 admits this kind of uniformity can be done with standard techniques.

[5] Subtracting a register with integer j from a register with integer i costs exactly $\log i + \log j$ on a log-cost RAM, so the decrementing takes $\sum_{i=0}^{(d/\log d)-1} \log(\frac{d}{\log d} - i)$ time, or $\log(\prod_{i=0}^{(d/\log d)-1}(\frac{d}{\log d} - i)) = \log(\frac{d}{\log d}!) = \Theta(d)$.

6. REFERENCES

[1] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM* 28(1):114–133, 1981.

[2] A. K. Chandra, L. Stockmeyer, and U. Vishkin. Constant Depth Reducibility. *SIAM J. Comput.* 13(2):423–439, 1984.

[3] P. W. Dymond and M. Tompa. Speedups of Deterministic Machines by Synchronous Parallel Machines. *J. Comput. Syst. Sci.* 30(2):149–161, 1985.

[4] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

[5] J. Y. Halpern, M. C. Loui, A. R. Meyer, and D. Weise. On Time versus Space III. *Mathematical Systems Theory* 19(1): 13–28, 1986.

[6] J. Hopcroft, W. J. Paul, and L. G. Valiant. On Time vs. Space. *J. ACM* 24(2):332–337, 1977.

[7] N. Immerman. *Descriptive Complexity.* Springer-Verlag, 1999.

[8] R. J. Lipton and A. Viglas. Non-uniform Depth of Polynomial Time and Space Simulations. In *Fundamentals of Computation Theory*, LNCS 2751:311–320, Springer, 2003.

[9] L. Mak. Are Parallel Machines Always Faster than Sequential Machines? (Preliminary Version). In *Proceedings of STACS*, LNCS 775:137–148, Springer, 1994.

[10] L. Mak. Parallelism Always Helps. *SIAM J. Comput.* 26(1):153–172, 1997.

[11] F. Meyer auf der Heide. Speeding up Random Access Machines by Few Processors. In *Proceedings of STACS*, LNCS 210:142-152, Springer, 1986.

[12] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[13] M. Paterson and L. G. Valiant. Circuit Size is Nonlinear in Depth. *Theor. Comput. Sci.* 2(3):397–400, 1976.

[14] W. J. Paul and R. Reischuk. On Alternation II. A Graph Theoretic Approach to Determinism Versus Nondeterminism. *Acta Inf.* 14:391–403, 1980.

[15] W. J. Paul and R. Reischuk. On Time versus Space II. *J. Comput. Syst. Sci.* 22(3):312–327, 1981.

[16] L. Stockmeyer and U. Vishkin. Simulation of Parallel Random Access Machines by Circuits. *SIAM J. Comput.* 13(2):409–422, 1984.

[17] H. Vollmer. *Introduction to Circuit Complexity.* Springer-Verlag, 1999.