CMU-CS-81-141

[illegible]

Optimality in VLSI

Bernard CHAZELLE and Louis MONIER

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

September 1981

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

Table of Contents

List of Figures

Abstract

The more realistic model of computation introduced in recent papers by Chazelle and Monier (1981a, b) has led to drastic revisions of VLSI complexity in general. Measured in terms of chip area and computation time the complexity of several problems has been shown to be much higher than previously thought. We propose here to investigate the actual performance of well-known circuits in this new model, and to suggest designs which meet criteria of optimality. We will show in particular that many complicated schemes falsely believed to be efficient can be advantageously replaced by simpler and higher performance designs.

1. Introduction

The more realistic model of computation introduced in recent papers by Chazelle and Monier (1981a, b) has led to drastic revisions of VLSI complexity in general. Measured in terms of chip area and computation time the complexity of several problems has been shown to be much higher than previously thought. We propose here to investigate the actual performance of well-known circuits in this new model, and to suggest designs which meet criteria of optimality. We will show in particular that many complicated schemes falsely believed to be efficient can be advantageously replaced by simpler and higher performance designs.

Throughout this paper we will base all analyses of circuits on the model of computation described by Chazelle and Monier (1981b). The major new assumption of this model is to require propagation times at least linear in the distance. To make the model suited for upper bounds, we will add that linear propagation time is actually realistic with current technologies, e.g., electrical (with use of repeaters) or magnetic-bubble. Note that, based on these assumptions, the concept of optimality is meaningful only for large circuits, since the actual complexity of very small chips is overshadowed by parasitic effects. However, asymptotically optimal circuits which are conceptually simple will provide useful insights and guidelines for small designs.

One major consequence of the model is that the time performance of a circuit is strongly dependent on its geometry rather than its topology. In particular, all the tree-based schemes previously proposed cease to have their claimed logarithmic complexity. Examples of such circuits can be found in Preparata and Vuillemin (1979), and in Thompson (1980a). Also, to be realistic, we must assume that clock signals follow the same law of propagation as any other signal. For example, broadcasting a control signal in constant time becomes impossible, which significantly alters the control of a device as simple as a shift register.

In this paper, we carry out these ideas and present optimal designs for the following problems: Addition, cyclic shift, integer product, matrix arithmetic, linear transforms and FFT, sorting, and searching. Although we also establish a number of lower bounds, most of those used to prove optimality can be found in Chazelle and Monier (1981b). A common point for all these problems is to have an $\Omega(N^{1/2})$ lower bound on the time T, where N is the size of the problem. This shows the importance of pipelining computations in order to increase the throughput, and leads to the introduction of the period P, defined as the minimum time elapsed between two consecutive inputs. Thus we can analyze the behavior of circuits in the light of three measures: the time of computation T, the period P, and the chip area A. Composite measures defined as products of A, T, or P will also be considered to show possible resource trade-offs.

2. Addition

For the problem of adding two N-bit integers, the following lower bounds have been shown by Chazelle and Monier (1981b): $T=\Omega(N^{1/2})$, $AT=\Omega(N)$, $ATP=\Omega(N^2)$.

The simplest adder, consisting of a full-adder cell, has unit area and runs in linear time. It is thus optimal for the measures A, AT and ATP.

In order to achieve optimal time performance, the carry-look-ahead scheme had been previously proposed. However, it is no longer optimal in our model, for its straightforward implementation yields linear time, while the H-tree layout creates enormous difficulties for ordering the inputs. Instead, we propose a new adder which can be regarded as a one-level CLA. Wlog, assume that $N=m^2$. We decompose each input number $a=a_{N-1}...a_0$ and b into m blocks of m consecutive bits. Each block is read in sequentially into a cell which computes the sum of the two *slices* as well as the last carry, and which checks whether all the bits in the sum are equal to 1. Thus for each block, the sum $(s_{i,m-1},...,s_{i,0})$, the last carry c_i, and one bit $p_i=s_{i,m-1}\wedge...\wedge s_{i,0}$ are computed. It is important to notice that $c_i=1$ implies $p_i=0$.

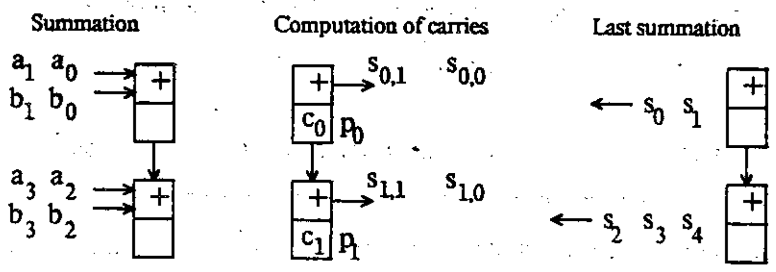

Fig. 1. *An optimal adder.*

After m steps, we can propagate each carry at the block level by computing the actual carry sequentially: if $p_i=0$ the running carry is c_i, else it is c_{i-1}. Finally we can update the partial sums in O(m) time with the value of the running carry (Fig.1). With $A=O(N)$ and $T=O(N^{1/2})$, this adder is optimal for T and ATP. Moreover, it requires very little logic since it is made of O(N) shift-register cells and $O(N^{1/2})$ full-adder-like cells. We observe that this scheme seems especially well-suited for magnetic-bubble technology.

One shortcoming of this adder, though, is its failure to allow pipelining. It is easy, however, to design a very simple adder with period $P=1$. Fig.2 illustrates this new scheme. Note that both time and area are linear, while the unit period makes this adder optimal for P and ATP.

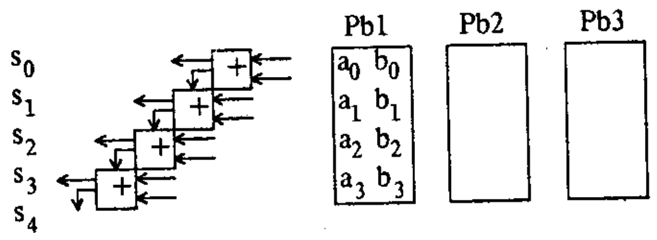

Fig. 2. *A fully-pipelined adder.*

3. Cyclic shift

The complexity of the cyclic shift has been studied by Chazelle and Monier (1981b) and Vuillemin (1980); the following lower bounds have been established: $A=\Omega(N)$, $T=\Omega(N^{1/2})$, $AP^2=\Omega(N^2)$.

The function takes for input a pair (a,p), where a is a binary sequence $a_1,...,a_N$ and p is the value of the shift, and it returns the sequence $b_1,...,b_N$ with $b_i=a_{i-p[\text{mod } N]}$. We will describe a circuit optimal for both A and T, which is based on a clock-free implementation of a shift register. For simplicity we will assume that $N=m^2$, and that we are given α and β, with $p=\alpha m+\beta$ $(\beta<m)$.

Since we assume that α and β are given in binary representation, the first task of the shifter will be to decode these integers. So, we start by describing a decoder, that is, a circuit receiving an integer $i\leq m$ as input and producing a sequence of i 1's. Let $i=i_k...i_0$ with $k=\lfloor \log_2 m \rfloor$.

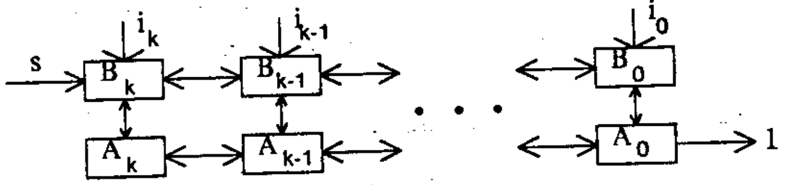

Fig. 3. *A decoder.*

The circuit is a ladder of cells as shown in Fig.3. First the cells $B_k,...,B_0$ read in the bits $i_k,...,i_0$ respectively. Then when the start signal s is activated, if $i_k=0$, B_k activates B_{k-1} and the process iterates; otherwise it activates A_k, whose task is to generate 2^k 1's on the output port of A_0. To do so, A_0 simply outputs a 1 when activated by A_1 or B_0, and it acknowledges A_1 in the former case. Recursively A_i activates A_{i-1}, waits to be acknowledged by A_{i-1}, repeats this operation, and when finished, acknowledges B_i. Finally B_i activates B_{i-1} and the process iterates. Each A_i and B_i is a very simple one- or two-state automaton whose details we can omit. Clearly the time to decode any integer $\leq m$ is $O(2^k)=O(m)$.

Next we need to describe a scheme for shifting a chain of bits to a given position without resorting to a broadcast synchronous clock. Consider a chain of m cells $A_1,...,A_m$ with the first p cells holding values $k_1,...,k_p$. We wish to transfer k_p to A_m,..., k_1 to A_{m-p+1}. To do so, we simply activate the first p cells in time O(p), then while k_p proceeds to move towards A_m, each cell A_i which is currently holding a key conditionally transfers its content to the cell A_{i+1} if it is vacant. The complete transfer is thus completed in O(m) time.

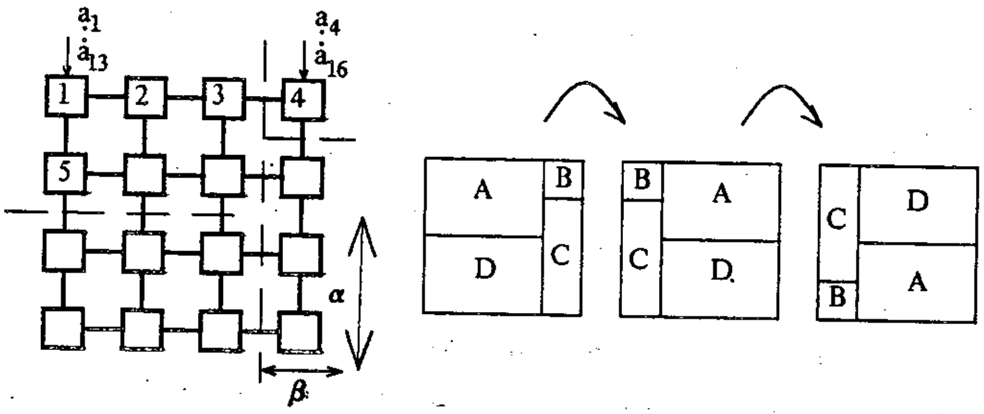

Fig. 4. *An optimal cyclic shifter.*

We are now ready to implement the cyclic shift. The circuit is an m×m array of cells as depicted in Fig.4. The input $a_1,...,a_N$ is read in by rows, and in a first stage the decoder described above permits us to delimit the areas A,B,C,D in O(m) time. Then the actual shift proceeds in two phases as indicated in the picture. This can be done by having two channels between adjacent cells, enabling 2-way communication between them. Then it is straightforward to implement the two phases with the description of the shift register given above.

4. Integer Product

The following bounds are known for the product of two N-bit integers: $A=\Omega(N)$, $T=\Omega(N^{1/2})$, $AP^2=\Omega(N^2)$. Not only tree-based schemes (Wallace trees) no longer yield logarithmic time, they cannot give better than linear time, since they all generate N^2 temporary bits. Instead, we propose a simple revised version of the shift-and-add scheme (Fig.5).

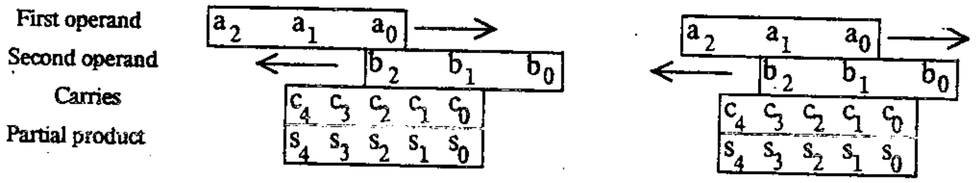

Fig. 5. *The shift-and-add multiplier revisited.*

It makes use of two clock-free shift registers meant to multiply the two numbers in carry-save representation; the carries are then released sequentially. This circuit is interesting mainly for its simplicity and the fact that it uses minimal area, $A=O(N)$. It is possible to connect N copies of this circuit in order to obtain minimum period, $P=1$; after each shift-and-add step, operands and partial results are shifted up one position, which after N steps yields the product in carry-save representation. The overall performance is: $A=O(N^2)$, $T=O(N)$ and $P=O(1)$, which makes the circuit optimal for P and AP^2.

Of all the circuits previously held as being *optimal*, e.g., Wallace tree or multiplication through DFT, none shows good time performance in our model. Recently, Preparata (1981) has given a careful description of a fast multiplier, optimal in our model: $A=O(N)$ and $T=O(N^{1/2})$. It uses a square-mesh structure to compute the convolution of the two numbers (considered as vectors) via the DFT, then releases the carries to obtain the usual product. The processors used in this mesh are very similar to those used in the sorting or DFT circuits mentioned in this paper: they mainly are small arithmetical units containing stored programs.

5. Matrix arithmetic

We next turn our attention to three matrix problems: Integer or boolean multiplication, transitive closure, and inversion. Since the first problem is reducible to the last two, as remarked by Savage (1979), the results of Chazelle and Monier (1981b) show that all three problems have the lower bounds, $A=\Omega(N)$, $T=\Omega(N^{1/2})$, with N measuring the total number of elements in the matrices.

Kung's systolic multiplier, described in Mead and Conway (1980), is thus optimal for A and T, and is the best known yet, although it cannot be pipelined. The same remark applies for the mesh-connected circuit proposed by Kung, Guibas, and Thompson (1979) to compute a transitive closure (note that for these results; we assume unit cost for element operations). Similarly Kung, in Mead and Conway (1980), describes an optimal circuit for inverting matrices which admit LU-decompositions (and Gaussian elimination without pivoting). It is easy to extend this scheme to compute the determinant of such matrices in time $O(N^{1/2})$. This matches the lower bound for arbitrary matrices, as is shown in the following.

Lemma 1: The time required to compute the determinant of an arbitrary $m \times m$ matrix is $\Omega(m)$.

Proof: Using a result from Chazelle and Monier (1981b), we simply have to show that computing a determinant involves a fan-in on $\Omega(m^2)$ elements.

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

$$A_k = \left[\begin{array}{c|c} A_{k-1} & \begin{matrix} r_1 \\ \vdots \\ r_{k-1} \end{matrix} \\ \hline a_{k,1} \ \cdots \ a_{k,k-1} & 0 \end{array}\right]$$

where the r_i's are the sums of A_{k-1}'s rows, i.e., $r_i=a_{i,1}+\ldots+a_{i,k-1}$.

Noting that we can rewrite A_k as

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

we derive the relation

$$\mathrm{Det}(A_k) = -\mathrm{Det}(A_{k-1}) \times (a_{k,1} + \dots + a_{k,k-1})$$

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

6. Linear Transforms and Discrete Fourier Transform

Vuillemin (1980) has shown that any circuit which can compute any linear transform on N k-bit elements (with $k \geq \log_2 N$) computes a transitive function of degree Nk. Using this result, lower bounds have been found for this problem by Monier and Chazelle (1981b): $A = \Omega(Nk)$ and $T = \Omega(N^{1/2}k^{1/2})$. One possible implementation requires that, in a first stage, a description of the transform be passed to the circuit as a parameter, thus enabling the circuit to compute the transform on any input vector. As yet, the best method known consists of either storing the matrix on the chip in its traditional array representation, or treating it as part of the input. In both cases, we can then use a matrix-vector multiplier, such as the linear-time, linear-area, systolic multiplier proposed by Kung in Mead and Conway (1980). This circuit is not optimal in time, but it seems hard to improve its performance as long as the N^2 elements of the matrix are to be memorized.

Note that we are often interested in computing only specific linear transforms. The previous lower bounds no longer hold, since they yield no information on the behavior of a particular transform. We choose to turn our attention to one of the most important, the discrete Fourier transform. For this problem, a lower bound is already known: $AT^2 = \Omega(N^2k^2)$. We extend this result in the following.

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

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

Noticing that the first element y_0 is the sum of all the x_i, we can prove that one of its bits is a fan-in of O(Nk) input bits, simply by exhibiting a *hard-input* (see Chazelle and Monier (1981b) for definition of fan-in). Let $x_0 = 2^j - 1$ with $j = \lfloor k/2 \rfloor$, and $x_i = 0$ for $i = 1, \dots, N-1$. The jth bit of $y_0 = 2^j - 1$ is equal to 1; however, a change in the value of any bit of order $\leq j$ of any x_i ($i > 0$) will force it to 0.

Hence, this particular bit is a fan-in of Nk/2 input bits, which yields the desired lower bound on the time.

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

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

A good survey of DFT circuits can be found in Thompson (1980b). Note that most of the schemes reviewed are optimal in a model where transmission costs are neglected. However, this ceases to be true in our model, where no logarithmic times are possible. Good examples are the straightforward implementation of the FFT network, or the CCC-scheme proposed by Preparata and Vuillemin (1979), where wires of length N contribute to the poor performance: roughly $T = O(N)$, $A = O(N^2)$.

Instead, we can see how very simple circuits are indeed optimal in our model. One simple method consists of performing a matrix-vector product, where the matrix (ω^{ij}) is input to the hexagonal systolic multiplier. We have $A = O(N^2 k)$ and $T = O(Nk^{1/2})$ provided that optimal adders and multipliers are used. This circuit can be pipelined, thus reducing the period to $O(k^{1/2})$, which is optimal for the measure AP^2.

Another solution consists of using a linear matrix-vector multiplier, generating the matrix elements *on the fly*, as described in Kung and Leiserson (1979). The circuit is more efficient, $A = O(Nk)$ and $T = O(Nk^{1/2})$, but it cannot be pipelined.

A near-optimal design is the square mesh used to simulate a FFT network, as described by Stevens (1971) and Thompson (1980a). It involves N processors, each having the complexity of a microprocessor. The size of the programs involved, however, is comparable to the length of the words processed, which makes the total area O(Nk). Note that the computation time $T = O(N^{1/2}k^{1/2})$ is optimal and matches the I/O time.

7. Sorting

From Chazelle and Monier (1981b) we know that any circuit sorting N k-bit numbers, with $k \geq 2\log_2 N$, requires $\Omega(N)$ area and $\Omega(Nk)^{1/2}$ time. Only the grid implementation of sorting networks proposed by Thompson and Kung (1977) appears to give optimal time performance. However, this scheme requires $\Omega(Nk)$ area, and thus does not match the known lower bound. This lower bound does not exclude the existence of a bit-serial sorting device, but no optimal-area scheme has yet been proposed.

8. A general data structure scheme

As of yet, only the priority queue described in Leiserson (1979) has the complexity initially claimed. All the tree-like structures fail to be logarithmic for reasons already stated. Simple geometric considerations show that it is impossible to access any element of a set of N keys in less than $N^{1/2}$ steps. However we can achieve this performance with the scheme illustrated in Fig.6. The circuit we propose can be used as a dictionary where queries are answered in $O(N^{1/2})$ time, with unit period.

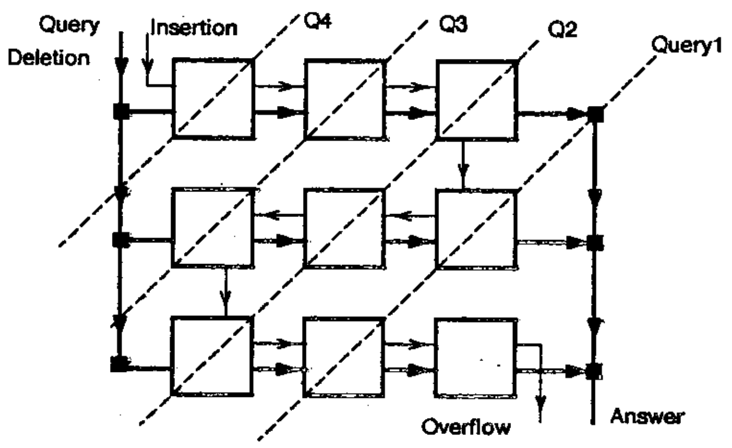

Fig. 6. *An optimal data structure.*

Insertions of keys proceed from the single input port by scanning the mesh sequentially in snake-order, and detecting the first vacant cell. Deletions and queries are handled by a downward diagonal sweep. Thus by using broadcasting, deletions are completed in $O(N^{1/2})$ time. Queries are treated in a similar fashion, running on rows in a left-to-right motion, and downwards on the rightmost column where the partial results from each row are merged. Proper synchronization is crucial; in particular the treatment of insertions and queries/deletions must occur in an alternate mode. Note that at all times we can distinguish between two kinds of keys: the keys stored at their final place, and the running keys in search of a vacancy. However, deletions and queries treat both of them in a similar way. A rigorous proof of correctness is left to the reader.

9. Conclusions

As our last application of the geometric nature of the model considered here, we can show how the idea proposed by Browning in Mead and Conway (1980) to solve NP-complete problems must now be discarded. The method is to simulate a non-deterministic Turing machine by using an exponential number of processors connected up in a tree structure. The paths between root and processors contain a polynomial number of nodes; however, a simple geometric argument shows that for any embedding of the tree, there exists paths of exponential length. This implies exponential communication times, which defeats the purpose of the scheme.

In this paper we have wished to show how the advantages of the high concurrency offered by VLSI technology should be appreciated from a realistic perspective. So far it appears that simplicity and elegance in design should win out. Complicated schemes don't seem to pay off, and even if this observation has been justified in an asymptotic -thus highly academic- framework, we still believe that it bears great significance on a more practical level. After focusing on lower bounds at length, theoretical work in VLSI should now turn decidedly to more practical aims including design and conception.

Acknowledgments

Many thanks to Mike Foster who read a preliminary version of this paper, and to Franco Preparata for communicating to us his recent results on multiplication. This research was supported in part by the Defense Advanced Project Agency under contract F33615-78-C-1551, monitored by the Air Force Office of Scientific Research.

References

Chazelle, B.M. and Monier, L.M. (1981a). *Towards More Realistic Models of Computation for VLSI*, proc. of Caltech Conference on VLSI, January 1981.

Chazelle, B.M. and Monier, L.M. (1981b). *A Model of Computation for VLSI with Related Complexity Results*, proc. 13th Annual ACM Symposium on Theory of Computing, ACM, May 1981.

Guibas, L.J., Kung, H.T. and Thompson, C.D. (1979). *Direct VLSI Implementation of Combinatorial Algorithms*, proc. Caltech Conference on VLSI, January 1979.

Kung, H.T. (1979). *Let's Design Algorithms for VLSI Systems*, proc. Caltech Conference on VLSI, January 1979.

Kung, H.T. and Leiserson, C.E. (1979). *Systolic Arrays for VLSI*, Sparse Matrix Proceedings 1978, pages 256-282. Society for Industrial and Applied Mathematics, 1979.

Mead, C. and Conway, L. (1980). *Introduction to VLSI Systems*, Addison-Wesley, 1980.

Preparata, F.P. (1981). *A Mesh-Connected Area-Time Optimal VLSI Integer Multiplier*, Technical Report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, March 1981.

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

Savage, J.E. (1979). *Area-time Tradeoffs for Matrix Multiplication and Related Problems in VLSI Models*, Proc. 17th Annual Allerton Conference on Communications, Control, and Computing, 1979.

Stevens, J.E. (1971). *A Fast Fourier Transform Subroutine for Illiac IV*, Technical Report, Center for Advanced Computation, Illinois, 1971.

Thompson, C.D. (1980a). *A Complexity Theory for VLSI*, PhD dissertation, Department of Computer Science, Carnegie-Mellon University, 1980.

Thompson, C.D. (1980b). *Fourier Transform in VLSI*, Memorandum No. UCB/ERL M80/51, University of California, Berkeley, October 1980.

Thompson, C.D. and Kung, H.T. (1977). *Sorting on a Mesh-Connected Parallel Computer*, C.A.C.M., April 1977, v.20, No.4.

Vuillemin, J. (1980). *A Combinatorial Limit to the Computing Power of VLSI Circuits*, Proc. 21st Annual Symposium on Foundations of Computer Science, Oct. 1980.