# OPTICAL COMPUTER ARCHITECTURES

## The Application of Optical Concepts to Next Generation Computers

**Alastair D. McAulay**

NCR Distinguished Professor and Chairman
Department of Computer Science and Engineering
Wright State University

**A Wiley-Interscience Publication**

**JOHN WILEY & SONS, INC.**

**New York Chichester Brisbane Toronto Singapore**

***Library of Congress Cataloging in Publication Data:***

McAulay, Alastair D.

Optical computer architectures : the application of optical concepts to next generation computers / Alastair D. McAulay.

p. cm.

"A Wiley-Interscience publication."

Includes bibliographical references and index.

1. Computers, Optical. 2. Computer architecture. 3. Optical data processing. I. Title.

TA1630.M43 1991

621.39'1—dc20 90-42103

ISBN 0-471-63242-2 CIP

Printed in the United States of America

10 9 8 7 6 5 4 3

# Preface

Rapid advances in optical technology, including the availability of low cost laser diodes and fiber optics, create opportunities for significantly advancing the field of computing. Currently, optics is appearing in computer memories and for computer interconnections. The unique advantages of optics over conventional electronics suggests that optics will become increasingly important in computers.

This book is aimed at those interested in how to use optics for computing. It was used as a text for a first year graduate course in *Optical Computing* in the Department of Computer Science and Engineering at Wright State University. It emphasizes concepts and provides the background to understand them. Specific knowledge in optics and computer architecture is not assumed. Exercises are provided for verifying understanding.

Part I provides background for understanding the rest of the book. Chapter 1 discusses the motivation for considering optics and ways in which optics might evolve to become dominant in computing. Chapter 2 provides a review of basic concepts involving optical wave and lens systems. Chapter 3 reviews the basic principles of coherent processing. Electronic and optical addressable optical devices, suitable for optical computing, are discussed in chapters 4 and 5 respectively.

Part II describes methods of assembling the optical components into computing subsystems. Modularity in computer design is achieved by grouping together bits into words and operations into subsystems. This provides economy of design and manufacture, and permits specific computers to be designed by tailoring combinations of these subsystems. Methods of providing optical interconnections for computing are discussed in chapter 6. Chapter 7 considers optical memory. Chapter 8 shows how logic may be performed optically for use in control or arithmetic units. The synthesis of logic circuits is described in chapter 9 and involves merging parallel logic and interconnections. The construction of highly parallel arithmetic units is described in chapter 10. Matrix computations are discussed in chapter 11. Algorithms for high performance computing are described in chapter 12.

Part III describes optical computing systems constructed according to different architectural models of computation. Most computers today operate sequentially on instructions, updating only one element of the state at a time. Chapter 13 describes the history of such machine design and provides

an all optical implementation of a sequential machine. The other models of computation described are more parallel and may therefore be more suitable for optical implementations. Optical dataflow machines, chapter 14, operate as data becomes available and have no concept of state. Optical cellular automata, chapter 15, represent a model of computation in which the whole state is updated at each time step, and the interconnections are local. The important role of cellular automata in computer theory is reviewed. Optical linear neural network models of computation involve learning and widespread interconnections, chapter 16. Learning is aimed at reducing programming and providing adaptivity. Greater capability is achieved with optical nonlinear neural networks, described in chapter 17. Optical autoassociative and self organizing neural networks are discussed in chapter 18.

I wish to acknowledge the many researchers in this field whose publications I have referenced or with whom I have had discussions, including organizations such as the International Society for Optical Engineering (SPIE), the Optical Society of America (OSA), the IEEE Computer Society, and the Association for Computing Machinery (ACM). I also wish to thank the funding agencies that have supported my activities, including Air Force Office of Scientific Research, Defense Advanced Research Projects Agency, NCR, and Wright State University.

ALASTAIR D. MCAULAY

*Kettering, Ohio*

# Contents

# Chapter 1

# Why Optical Computers?

This chapter discusses the reasons for considering optics for computing and provides incentives for reading many of the subsequent chapters. In section 1.1 we will discuss future requirements for computers, difficulties of meeting them with existing technologies, and suggested future directions. The advantages and possible roles for optics in computing are discussed in section 1.2.

## 1.1 Electronic Computer Architectures

We will discuss the requirements for future computers, the limitations of currently used sequential models implemented in electronics, and new directions suggested by the human brain.

### 1.1.1 Requirements for Future Computers

A number of desirable features are identified for future computers. A future computer is expected to have the *flexibility* to run a wide range of algorithms and languages, including ones not yet developed. The user requires *extendability* so that the machine can be extended as his or her needs grow rather than having to purchase a new, larger, and possibly incompatible machine. Similarly, the manufacturer would like *scalability* so that one product line using the same software has machines from small to large. *Reliability* is critical because down-time may have a high cost to the user. *Fault tolerance* permits the system to continue operating at a reduced capability after a part of the system fails. The *software environment* must be efficient and easy to use for the appropriate level of skill of the user: the system software developer, the application software developer, and application user. Also, *cost effectiveness* is clearly critical in the present competitive marketplace.

These desired features influence computer architecture as follows. *Reconfigurability* in parallel machines is required for fault tolerance, flexibility,

and extendability. Such machines suggest complex interconnection networks. These are difficult to construct and expensive with electronic systems due to interference. ***Massive parallelism*** is required for extendability, scalability, fault tolerance, and high performance. The latter suggests that the interconnection networks have a high bandwidth and are fine grain. *Symbolic* computation capability for string manipulation, reasoning, and expert systems is required to provide good software environments and for some applications. Consequently, there is a move to parallelism and symbolic computation.

### 1.1.2 Limitations of Current Computer Technologies

Limitations are due to electronics, the architectural model of computation, and software approaches.

#### Electronic Limitations

Computers have increased their capabilities by approximately an order of magnitude every five years for the same cost since the 1950s. This was accomplished by shrinking the size of active electronic elements such as transistors on semiconductor chips. The cost of manufacturing a chip does not change much, so the cost per element decreases with increasing density. Further, smaller elements have increased speed. It is getting increasingly difficult to shrink the elements further because the switching speed is now similar to the communication speed between elements, and the latter does not decrease with size. This is because the resistance-capacitance (RC) time-constant of the interconnecting conductor remains constant. As the link shrinks, its capacitance decreases but the resistance increases; thinner wire presents greater resistance to current flow. Progress in shrinking electronics further is hampered because the number of pins for accessing the device and the bandwidth per pin are severely limited. Optical interconnections to chips are a method of overcoming these limitations.

#### Architectural Limitations

Another serious limitation with current technology is the sequential computer architecture in which instructions are implemented in sequence (figure 1.1). The concept involves separating memory, computation, interconnection, and control. The high cost of early electronic arithmetic units favored channeling everything through a single unit. Further, the sequential architecture uses random access memory (RAM) to store information and to save interconnections. Electronic interconnections are expensive in terms of reliability, cost, and power because electrons are charged particles which interfere with one another. In RAM, each item to be remembered is stored in an address represented in binary form. Hence, $n$ bits can be selected to reference $2^n$ addresses.

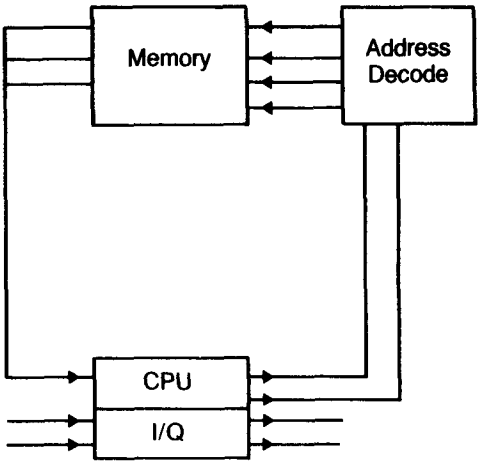


Figure 1.1: Sequential Computer Architecture

A decoder is used to translate the bits to permit selection of the address. This means (figure 1.1) that the total number of connections to memory is only $n+1$ rather than $2^n+1$ for addressing $2^n$ memory cells. In the figure $n=2$, only one of the $2^n$ memory cells is accessible at a time. The serial nature of RAM is a disadvantage for parallel computing.

Proposed approaches to parallelism with sequential architectures depend on the application. For example, in transaction systems, such as airline reservation systems, parallel access to memory is critical, but the transactions are otherwise uncoupled. The processors can be loosely coupled to each other. In contrast, tightly coupled processors are needed for a large scientific computation. Unfortunately, if a machine is too specific to an application, the cost of software and hardware development may not be justified by the market size. A massively parallel, tightly coupled machine could handle a wider range of applications, making the development of extensive software and hardware more attractive.

Massively parallel tightly coupled machines are difficult to construct with current electronic sequential architectures. Figure 1.2 shows that even using all the techniques of pipelining and overlap, the improvement in supercomputer performance with time was flattening out until Cray moved to replication of the system to two processors in 1982. The computation rates shown are for a linear equation solving software package LINPAK. Careful vectorizing was used to optimize performance on the Cray to achieve the faster rate. The techniques for increasing performance on early supercomputers, such as instruction pipelining and overlap, are now universal in workstations.

Figure 1.3 [122], shows the difficulty of increasing performance on a single

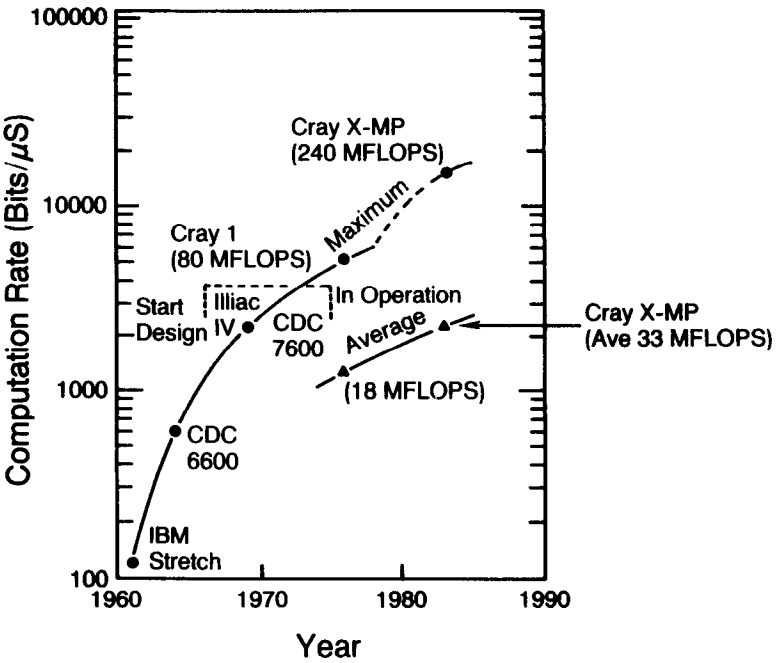


Figure 1.2: Performance of Supercomputers with Time

problem as more processors are used. Speed-up $S$ for a computation (y-axis) is defined as the ratio of the shortest time to perform the computation on a uniprocessor to the shortest time to perform the computation on the parallel processor. Note that different algorithms may be selected to provide the shortest times for the two architectures and that equivalence of accuracy and robustness must be considered. Minsky's conjecture, a maximum speed-up of $log_2 n$, has proven conservative. For example, for a given computation, a parallel algorithm is often possible that provides greater than eight speed up with 256 processors. The $n/ln\ n$ bound is derived in reference [122] and is achievable for some computations. In this case, 32 processors will provide a speed up of nine. Higher performance is rare, because as more processors are used, more time is lost in communication and overhead. Also, long computations, for which high performance machines are needed, often have sections of code that are not very parallelizable because they involve gathering data together from the parts, as in normalization.