

DTIC FILE COPY

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AD-A221 443

average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering the collection of information. Send comments regarding this burden estimate or any other aspect of this form to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20543.

ATE

3. REPORT TYPE AND DATES COVERED

FINAL REPORT 15 Jan 86 -15 May 89

Optics and Symbolic Computing

5. FUNDING NUMBERS

DARPA

2

6. AUTHOR(S)

Dr Brian G. Kushner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BDM Corporation
7915 Jones Branch Drive
McLean, VA 22102-3396

AFOSR-TR-

90-0509

8. PERFORMING ORGANIZATION
REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NE
Building 410, Bolling AFB DC
20332-644810. SPONSORING/MONITORING
AGENCY REPORT NUMBER

F49620-86-C-0030

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION STATEMENT

DTIC
ELECTE
APR 30 1990
S
3 D

13. ABSTRACT (Maximum 200 words)

Optical and optoelectronic technologies have been developing at a very rapid pace over the past decade. This growth has been largely fueled by the fiber optic telecommunications, novel (flat panel) optical scanning and recording systems. These technological advances have also lead to research in insertion of optics into computing systems. The role of optics can be in providing interconnections between electronic subsystems, providing largely capacity data storage or leading to a complete system capable of implementing the necessary operations. The exact role of optics depends on the specific domain of the computing system (signal/image processing, numerical computing or symbolic manipulations) and the nature of application in addresses.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION
OF REPORT

UNCLASSIFIED

18. SECURITY CLASSIFICATION
OF THIS PAGE

UNCLASSIFIED

19. SECURITY CLASSIFICATION
OF ABSTRACT

UNCLASSIFIED

20. LIMITATION OF ABSTRACT



**A Ford Aerospace
Company**

OPTICS AND SYMBOLIC COMPUTING

FINAL TECHNICAL REPORT

FEBRUARY 15, 1990

Approved for public release
distribution unlimited

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSO)
ATTENTION: TECHNICAL INFORMATION DIVISION
 This document has been reviewed and is
 approved for release IAW AFR 190-12.
 Distribution is unlimited.
WILLIAM J. KERPER
 Technical Information Division

90 04 27 089



A Ford Aerospace
Company

BDM INTERNATIONAL, INC.
7915 JONES BRANCH DRIVE
McLEAN, VIRGINIA 22102-3396
(703) 848-5000

OPTICS AND SYMBOLIC COMPUTING

FINAL TECHNICAL REPORT

FEBRUARY 15, 1990

BDM/MCL-90-03363-TR

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

Sponsored by
Defense Advance Research Projects Agency
Monitored by AFOSR Under Contract #F49620-86-C-0030

Prepared for Air Force Office of Scientific Research and
Defense Advanced Research Projects Agency.

EXECUTIVE SUMMARY

Optical and optoelectronic technologies have been developing at a very rapid pace over the past decade. This growth has been largely fueled by the fiber optic telecommunications, novel (flat panel) optical displays, high capacity optical storage (optical disks) systems and optical scanning and recording systems. These technological advances have also led to research in insertion of optics into computing systems. The role of optics can be in providing interconnections between electronic subsystems, providing large capacity data storage or leading to a complete system capable of implementing the necessary operations. The exact role of optics depends on the specific domain of the computing system (signal/image processing, numerical computing or symbolic manipulations) and the nature of applications it addresses.

BDM International, Inc. has been investigating the subject of "Applications of Optical Techniques to problems in Symbolic Computation" under a contract from Defense Advanced Research Projects Agency, monitored by Air Force Office of Scientific Research (Contract Number F49620-86-C-0030) from January 1986 through October 1989. The following document describes the results of research performed at BDMI under this contract. This Final Technical Report is divided into five self-contained parts representing different stages of research in this program. The Executive Summary contains a brief introduction to each of the parts.

Part I consists of a book chapter coauthored by Dr. B. G. Kushner of BDMI and Dr. J. A. Neff of Du Pont (formerly at DARPA) titled "Optics and Symbolic Computing". Written in the early stages of this program, this paper contains a basic exposition of the domain of symbolic computation, contrasting it with numerical computing. The primary difference between symbolic and numerical computation resides in the representation and the primitive processing operations. The paper further elaborates on the different application domains of symbolic computation (speech understanding, vision, natural language understanding, expert

systems) emphasizing the functional capabilities required by each domain. The implications of the functional requirements to parallel architectures is explored in the last section. Particular emphasis is placed on the dynamic and global interconnect capabilities provided by free space optical systems. Special purpose analog optical processors are also discussed in implementing accelerators for commonly encountered operations in symbolic computing.

Part II contains detailed discussions of two specific aspects of symbolic computation. An introductory discussion outlines the program philosophy by putting the optical symbolic computing effort in proper historical perspective within the optical computing research. It is concluded that symbolic computing is based on computational primitives that are different than those encountered in other optical computing areas. Special purpose relational data base machines were explored to identify these primitives. Batcher's odd-even network for sorting was identified as a widely useful operation in relational database machines and other symbolic computing applications. The network consisted of active planes containing processing modules performing the "compare-and-exchange" operations on two bit-serial data streams. These planes were interconnected via a permutation network. A perfect shuffle permutation network was identified as being particularly useful. Since these sorting algorithms are fully pipelined, the latency of interconnects is not important and it is important to achieve negligible clock skew - exactly the characteristics of an optical permutation network. The appendix to Part II contains the reprint of a paper that described different optical designs for a compare-and-exchange module.

The pattern matching operation commonly encountered in symbolic computation that has a superficial similarity to the matched filter correlation operation performed by an analog optical processor. A detailed investigation into the nature of pattern matching for symbolic computation is reported in Part II. The conclusion was that the dynamic and flexible nature of the

data representation (a list) precludes the application of parallel analog optical techniques to this operation. The attention was therefore turned to more deterministic data representations, such as binary matrices, for optical computing applications.

Part III of the report contains two reprints describing further development of the sorting network concepts outlined in Part II. The first paper, titled "Sorting with Optical Compare-and exchange modules", deals with the incorporation of the designs for the compare-and-exchange modules described earlier into sorting networks. The particular emphasis was on matching the characteristics of the all-optical or hybrid designs with specific application requirements. The second paper represents a collaborative effort with researchers at University of Arizona in demonstrating a compare-and-exchange module built with nonlinear interference filters exhibiting optical bistability. The the optical circuit design work was performed at BDM and the layout and construction of the optical system was performed at the University of Arizona. The main emphasis of the project was on demonstrating the use of optical bistability in a useful digital optical circuit. The performance obtained (kHz data rates) was limited by the nonlinear interference filters and higher speeds were projected for GaAs etalon devices.

The perfect shuffle permutation required for the sorting network can be implemented by using simple optical imaging systems and prisms/mirrors. An optical system demonstrating perfect shuffle permutation on 1-D data was demonstrated by researchers at University of Erlangen in FRG and AT&T Bell Labs. The size of data arrays that can be accomodated by a free-space optical imaging system increases significantly if the long 1-D array is "folded" into a 2-D matrix. A simple optical system can then perform a 2-D permutation on the matrix which is equivalent to a 1-D perfect shuffle permutation on the original data sequence. Part IV contains a reprint from Applied Optics Rapid Communications that describes the formulation and implementation of such a folded perfect shuffle optical processor. The paper

also describes experimental results obtained on a 64 element data array and the size was extended to a 1024 data array in our laboratory.

The binary matrix data representation discussed earlier was found to be the basis of a heuristic search procedure known as the consistent labeling. The second section of Part IV contains a preprint of a publication describing the use of optical processors implementing Boolean matrix operations in implementing the heuristic search procedure. Such optical processors due their inherent parallelism and ability to interface directly with parallel access optical data storage are particularly attractive to speeding up the forward checking operation that can reduce the time required for subsequent search operation.

The last part of this Technical Report (Part V) describes further analysis of the folded optical interconnection systems that includes engineering issues such as cascadability and power efficiency. Modifications to the basic folded perfect shuffle architecture is described that improves the net power efficiency by minimizing the losses and makes the input-output arrays totally compatible to ensure direct cascadability. Part V also describes an extension of the folded interconnection principle to another useful permutation, namely hypercube.

In summary, the research project undertaken has led to the following advancements:

- (1) Identification of the sorting network as a widely useful computational subsystem in symbolic computation.

- (2) Design and experimental demonstration (in collaboration with University of Arizona) of a digital optical module performing compare-and-exchange operation on two bit-serial (most significant bit first) data streams.

- (3) Invention of the folded optical permutation network concept; design and demonstration of a folded optical perfect shuffle system on a 1024 data array.

- (4) Extension of the folded optical system concept to hypercube connections and an engineering refinement of the folded

optical perfect shuffle system for improved efficiency and cascadability.

(5) Highlighting the difficulties in applying standard optical correlator techniques to the problem of pattern matching in symbolic computing.

(6) Identifying the forward checking and tree pruning operations in combination with binary matrix representation of the constraints as very promising candidates for optical symbolic computing approaches and preliminary design of an optical boolean matrix processor to speed up the forward checking procedure.



Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

PART I

TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
	TABLE OF CONTENTS	iii
	LISTING OF FIGURES	v
I	INTRODUCTION	I-1
II	WHAT IS SYMBOLIC COMPUTING?	II-1
	A. KNOWLEDGE CHARACTERISTICS	II-1
	B. KNOWLEDGE REPRESENTATION	II-6
	C. SEARCH	II-13
	D. ATTRIBUTES OF SYMBOLIC COMPUTING AS COMPARED TO NUMERIC COMPUTING	II-19
III	SYMBOLIC COMPUTATION: FUNCTIONAL CAPABILITIES	III-1
	A. OVERVIEW OF SYMBOLIC COMPUTING DOMAINS	III-1
	B. SPEECH UNDERSTANDING	III-6
	C. VISION	III-16
	D. NATURAL LANGUAGE UNDERSTANDING	III-29
	E. EXPERT SYSTEMS	III-37
	F. CONCLUSION	III-50
IV	SYMBOLIC COMPUTING ARCHITECTURES	IV-1
	A. INTRODUCTION TO SYMBOLIC ARCHITECTURES	IV-1
	B. ARCHITECTURES FOR PARALLEL PROCESSING	IV-2
	C. OPTICAL IMPLEMENTATION OF MULTIPROCESSOR ARCHITECTURES	IV-14
	D. ALL OPTICAL ARCHITECTURES	IV-19
	E. HYBRID OPTICAL-ELECTRONIC SYSTEMS	IV-27
	F. ACKNOWLEDGEMENTS	IV-32
V	GLOSSARY	V-1
VI	REFERENCES	VI-1

LISTING OF FIGURES

	<u>Page</u>
SECTION I:	
1. Trends in Numeric and Symbolic Computing Technology	I-2
SECTION II:	
2. Commonly Used Methods of Acquiring Information	II-3
3. Simplistic Example of Semantic Network	II-8
4. Frame of Knowledge Representing a 2D Spatial Light Modulator	II-9
5. Elements of Predicate Calculus	II-12
6. Hierarchy of Search Strategies	II-14
7a Directed-Graph Search	II-16
7b Tree Search	II-16
8. Best-First Search	II-20
9. Attributes of Symbolic and Numeric Computation	II-21
SECTION III:	
10. A Natural Language Understanding System	III-3
11. A LISP Machine	III-5
12. Speech Understanding Paradigm	III-8
13. Low-level Speech Processing	III-9
14. Semantic Net for Speech Recognition	III-11
15. Architecture of the Hearsay III Continuous Speech Understanding System	III-15
16. Low-level Vision Processing--Pattern Matching Approach	III-19
17. DOG Operator	III-21
18-1. Int-level Vision Processing - An Example	III-22
18-2. Int-level Vision Processing - An Example	III-23
19. Image Processing Paradigm	III-25
20. Prototypical Frame for a Tank	III-26
21. Possible Parsings of the Phrase "Optics is light work"	III-33
22. Elements of a Natural Language System	III-35
23. Natural Language, a Blackboard Model	III-36
24. The Elements of an Expert System	III-39
25. Applications of Expert Systems	III-43
26. Prototype Optical System Used in Expert System Example	III-44
27. Frame-based Representation of Optical System in Figure 26	III-45
28. A Modular, Parallel Expert System	III-51

LISTING OF FIGURES (CONTINUED)

	<u>Page</u>
SECTION IV:	
29. Block Diagram of a General Parallel Computer	IV-4
30. Classification of Parallel Processors by Nodal Complexity	IV-5
31. Microcomputer Array	IV-6
32. Enhancement of Interconnection via Multi-port Memories	IV-10
33. Processory/Memory Interconnect Employing a 3x3 Generalized Crossbar	IV-12
34. A Multi-Stage Switching Network	IV-13
35. Electronic Versus Optical Computing	IV-16
36. Hybrid/Optical Electronic Multiprocessor Architecture	IV-17
37. All-Optical Multiprocessor Architecture	IV-20
38. An All-Optical Processing Element	IV-21
39. Multi-Stage Optical Interconnect Network	IV-23
40. Optical Disk Interface to All-Optical System	IV-25
41. Pipelined Optical Gate Arrays	IV-26
42. Hierarchy of Functions in Hybrid Systems	IV-28
43. Parallel Processing of a Semantic Net	IV-33

SECTION I
INTRODUCTION

The incorporation of intelligence into computational systems is rapidly gaining momentum with both the computer science community and with a large segment of computer system users. The field has been traditionally labeled "artificial intelligence," or "AI" for short. It is difficult to exactly specify what is meant by AI, however, since intelligence is a relative merit which cannot be precisely quantified or defined. Some aspects of intelligence have been in computers from the beginning, even though AI conjures up an image of very advanced computer science. Afterall, memory capabilities are usually associated with intelligence and even the earliest computers incorporated some form of memory. A working understanding of what is meant by AI might succinctly be stated as imparting to computers those attributes which we associate with the human thought processes that are notably different from the way in which conventional computers operate.

An important characteristic of AI systems is their incorporation and utilization of knowledge in each computation. Computer scientists have been struggling for the last two decades to determine how best to represent knowledge in computing machines. Numerous techniques have evolved for creating, manipulating, and storing collections of symbolic structures. These structures, or knowledge elements, can be used to represent objects, events, knowledge about how to do things, and knowledge related to what is known (or meta-knowledge). Collectively, this set of symbolic structures is referred to as the knowledge base of the AI system.

Why does optical processing look attractive for performing symbolic manipulations or computing? This chapter is designed to answer this question in some detail, but on the surface one could readily cite computational throughput and operation compatibility. The need for computational throughput can be appreciated by comparing the throughput performance of numeric computers against that of dedicated LISP machines (LISP, or LISt Processing, being the symbolic computing language of choice in the US AI community). Figure 1 is representative of system performances, depicting a

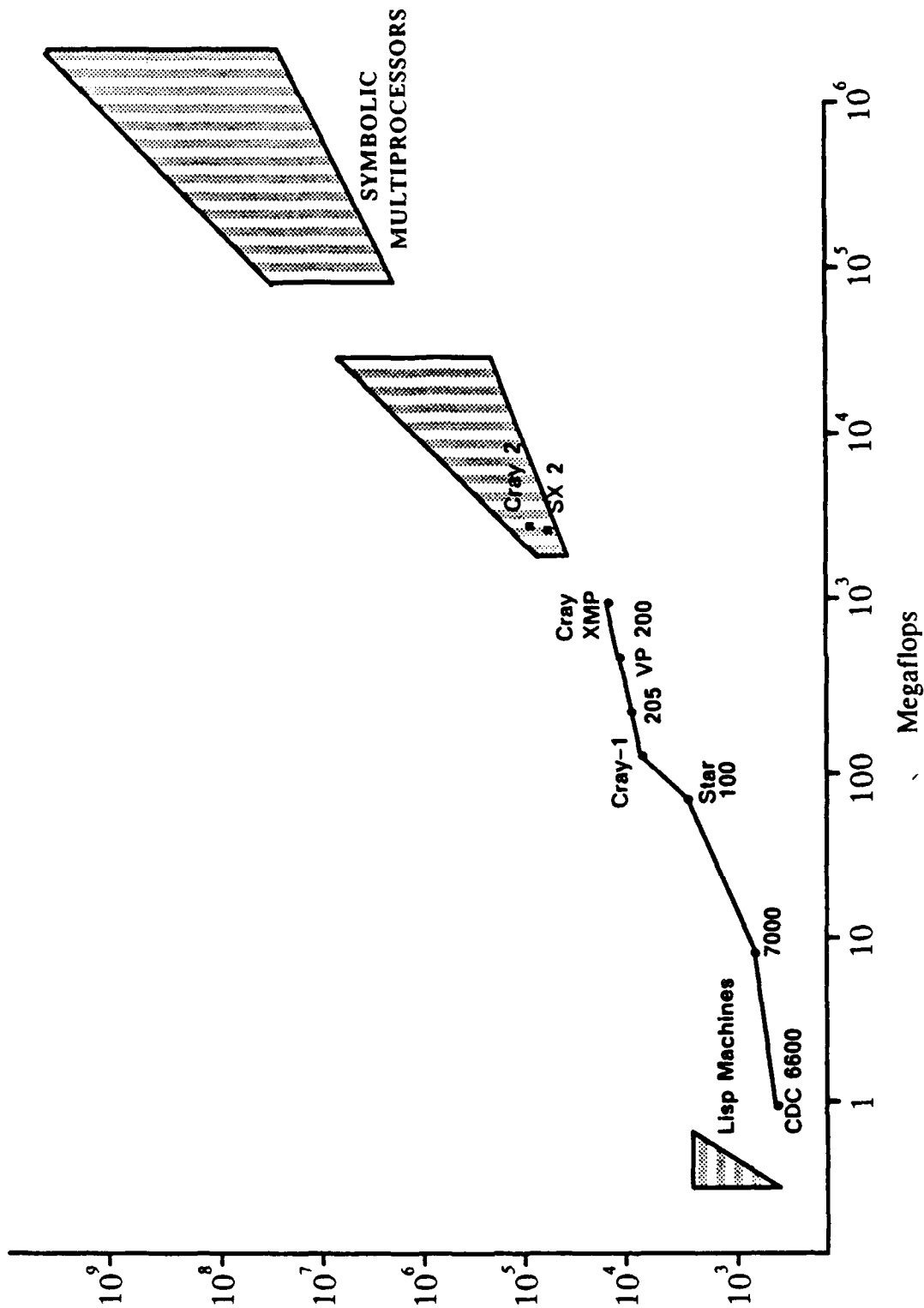


Figure 1
Trends in Numeric and Symbolic Computation Technology
(Source : IEEE)

typical measure of AI computing power on the vertical axis and operational speed on the horizontal axis.^{1,2} LIPS, or Logical Inferences Per Second, is used here as a measure of intelligence, since no functional equivalent of an "IQ test" currently exists for AI systems. The figure illustrates the slow speed of these LISP machines, relative to current generation "supercomputers" and next generation multiprocessors. The need to drastically increase the processing rate of AI systems, coupled with the limitations of current uniprocessor architectures, has resulted in a major research impetus to explore parallelism to enhance the speed of these machines and render them far more useful as modern computing systems. This utilization of parallelism indicates a potential synergism between symbolic and optical processing.

Interestingly enough, the numeric "supercomputers" execute AI functions at a greater rate than dedicated AI machines. This may imply that raw speed is an important component in overcoming existing AI computational bottlenecks. It also may indicate that architectures designed to improve numeric throughput will be useful in symbolic computation, and vice versa. The issue of mapping the computational structure onto desired functionality is currently a global issue of concern, and crosses all boundaries of computer science and mathematics.

The second factor potentially linking optical and symbolic processing, that of operation compatibility, derives from the need to perform correlation, searching, and matching types of operations on symbolic data. Many of these operations do not require high computational accuracy. The application of optics to symbolic computing will likely avoid what has traditionally been the "Achilles heel" of optical computing - the difficulty in achieving more than a few bits of accuracy. In addition, the dependence of symbolic computing on correlation functions (and their isomorphs) may provide an excellent opportunity to enhance symbolic computing performance with the use of optical correlators.

The next section will describe the fundamental attributes of symbolic computing, and will compare it with techniques commonly utilized in numeric computing. Following this introductory discussion, we will present a

description of the most important functional capabilities that are based on symbolic computing. This will lead to a discussion of problem areas that are likely to be faced in achieving these capabilities. These sections do not address optical symbolic computing, but the reader will likely find the information contained therein to be important to gaining an understanding of the synergism between optical and symbolic forms of computing. In fact, these sections are intended as an introduction to the subject of artificial intelligence, and, because of the breadth of the topic, we have had to limit our discussions to only the major aspects. Section IV will then lay the framework for symbolic computing with optics. Fundamental architectural concepts will be described with an emphasis on how they differ from the more conventional computer architectures. This will be followed by the authors' concepts of how optics could enhance the performance of symbolic processors.

In trying to cover such a broad topic, we have attempted to provide an introduction to each of the main disciplines within symbolic computing. However, in many cases we have had to sacrifice the details of a particular topic in order to provide a balanced presentation. In other cases, the material is simply beyond the scope of this book. For additional information, the interested reader can consult literature such as The Handbook of Artificial Intelligence,⁶ a three volume treatise which provides an excellent overview of AI technology.

SECTION II
WHAT IS SYMBOLIC COMPUTING?

In order to realize computers that are more capable of simulating human thought processes than is possible with today's numeric computers, the bit patterns within the computers must be made to represent arbitrary symbols in addition to arithmetic ones. For example, the computer should be able to provide an answer to the question "What path should I take to reach my destination (given all I know about my environment and my capabilities)?" as well as to an arithmetic question such as "What is the sum of 2 plus 4?" Humans routinely make routing decisions, but numeric computation was not developed to handle such problems. First of all, encoding the question itself into the computer offers a formidable problem. Second, if the destination and alternative path information is provided by a visual scene (i.e., visual navigation), the resulting object recognition and image understanding tasks can be immeasurably enhanced by symbolic manipulation. Finally, the decision process will likely draw on one's knowledge of the world, and this also relies on symbolic constructs.

A. KNOWLEDGE CHARACTERISTICS

A system that we would describe as knowledgeable has three main attributes: capability of acquiring additional knowledge, ability to retrieve appropriate information from a knowledge base, and the power of reasoning with the retrieved information to solve problems. In defining knowledge, we recognize that several other interpretations are possible, each with an independent set of characteristics. We believe that our set of attributes comprise a convenient definition for knowledgeable systems and one which will be used to provide a format for the following discussion; however, it suffers a malady common to any attempt to neatly dissect and categorize a complex phenomena - that of an inability to ascertain the independence of the attributes. This will become evident in the discussion

of acquisition in which reasoning (the third attribute) is described as a method of acquisition (first attribute) - this is what we know as learning.

Acquisition of knowledge can occur via three different routes as shown in Figure 2. First of all, knowledge can be placed into the computer by a programmer working in conjunction with a knowledge engineer. In the case of expert systems (to be discussed in Section III.E), the knowledge engineer acquires or extracts knowledge from an expert, and passes it onto the programmer, who embeds it within the computer. This process is referred to as knowledge engineering. Secondly, in the near future we expect more and more knowledge to be automatically acquired via sensors. This will become an increasingly popular technique with the advent of speech recognition and vision systems (to be discussed in Sections III.B and III.C). Both programmed and sensed acquisition may be accomplished by just rote techniques, but most often the information is classified in some way to facilitate the retrieval process. Classification is a process of associating each input with related items to form classes which aid significantly in identifying relevant data during knowledge base searches. Classification is also commonly known as linking and lumping. If one knows, upon acquiring a given piece of information, that it will be associated with an entity already in the knowledge base, then a link is specified between the two, and if many entities are likely to be used together, they are lumped into a larger structure. The reader should have a good appreciation and understanding of classification following the discussions on retrieval, knowledge representation, and search.

The third category of acquisition is learned knowledge. The field of machine learning is still in its infancy, but three areas that have shown recent progress are parameter adjustment, discovery, and analogical reasoning (ref. 18). The variation of parameters and stimuli is a standard scientific technique for learning. Two important areas of application for AI are in variation of classification parameters for knowledge acquisition (changing of classes into which objects are placed), and in adjusting heuristic function parameters for improved problem solving (to be discussed in Section II.C on Search). Discovery usually entails problem solving, and

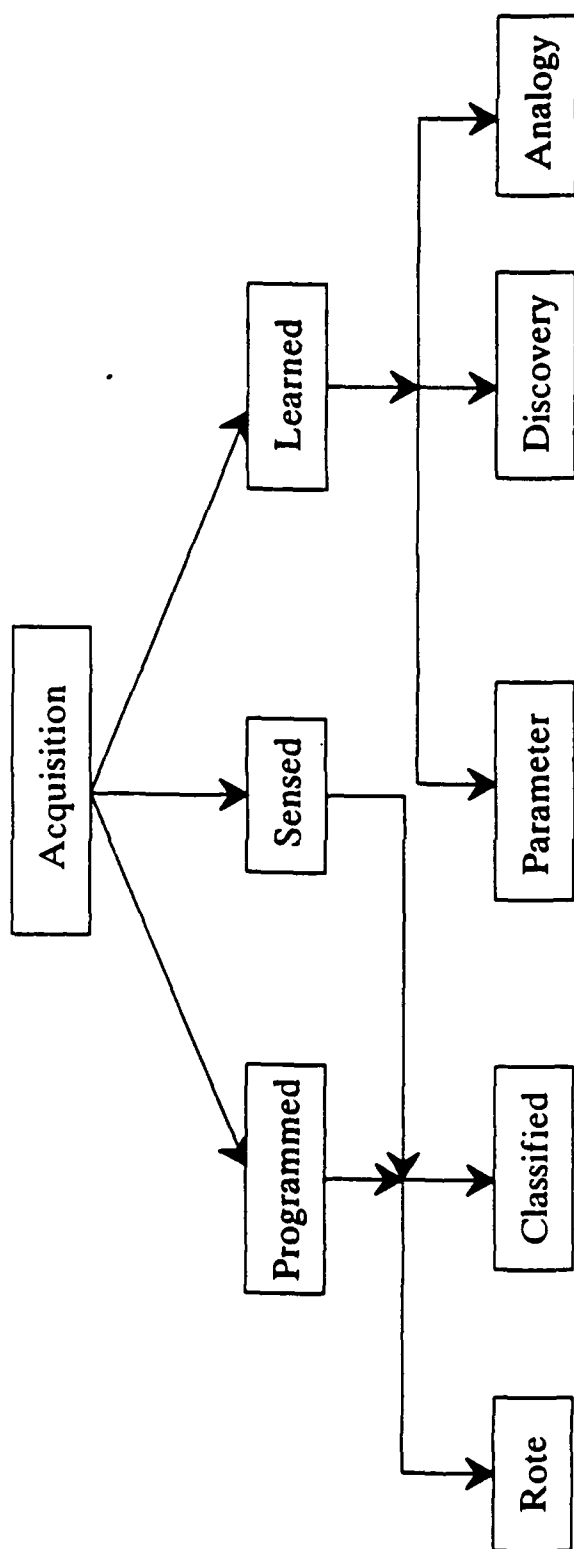


Figure 2
Commonly Used Methods of Acquiring Information

therefore relies heavily on reasoning. Reasoning also underlies learning-by-analogy which involves filling in missing information about some entity if it is known that the partially defined entity is "like" some already known entity. The representation of knowledge via frames and scripts (discussed in Section II.C) facilitates analogical reasoning since one can readily pass attributes between two frames or scripts that are said to be alike.

Retrieval is a very real problem for AI systems for several reasons, not the least of which is due to the large sizes of the data bases. Not only do these data bases contain the collection of facts that are relevant to the problem domain of a particular system, but they contain rules that enable the intelligent manipulation of the facts. One common technique of retrieval utilizes multiple indices which tag facts by one or more of their attributes. For example, a holographic lens could be indexed by such attributes as "optical", "diffraction grating", "conformal", "narrowband", and "lightweight". The LISP programming language, which is the most widely used language in the AI community, facilitates the assignment of attributes to objects in that it centers around lists of related symbols. The above example could be encoded in LISP using a "property list" as:

```
(holographiclens optical diffractiongrating conformal  
narrowband lightweight).
```

Two other retrieval schemes are based on pattern matching and contexts. The pattern matching scheme retrieves data according to some pattern which is related to data categories. A more advanced scheme is contextual storage, in which data are retrieved according to meanings. As an example of pattern matching, consider a data base containing the following lists:

```
(lightsource laser heliumneon wavelength(x) . . .)  
(lightsource laser NdYag . . . )  
(lightsource laser diode . . . )  
(lightsource arclamp mercury . . . )
```


THE BDM CORPORATION

```
(lightsource arclamp xenon . . . )  
(powersource 110Vinput 12Voutput . . . )
```

In order to retrieve those elements representing light sources, one could query the data base with a pattern denoted as:

```
(lightsource ?x)
```

To find laser entries, the pattern matcher would be specified by:

```
(lightsource laser ?x)
```

All of these retrieval schemes differ significantly from those used in numeric computers which store data according to memory addresses. Numeric information, being a subset of symbolic information, can be stored and retrieved via the schemes mentioned above, although likely not as efficiently. For example, the simple list (+ 2 4 9) associates the numeric symbols 2, 4, and 9 with the summation operation, and the nested lists (+ (* 3 4) (* 6 3)) associates 3 and 4 with one product operation, 6 and 3 with the other product operation, and associates 12 and 18 with the summation operation. This example has used basic LISP notation in which the first element of the list represents the operation to be performed while the remaining elements are the arguments to be operated upon.

In any discussion of retrieval, one must go far beyond the fundamental techniques for recognizing relevant data in the knowledge base to a discussion of how one searches for the set or sets of data that can lead to a defined goal. However, a discussion of search will be delayed until after knowledge representation is discussed since the two are closely related; i.e., knowledge is usually represented in such a way as to facilitate the search process.

Reasoning, which is the third capability of knowledge systems, is required when the system needs information that cannot be retrieved directly from the knowledge base. AI systems can tradeoff between large

knowledge bases and complex reasoning procedures. Systems must either have a high degree of reasoning power or be able to store and retrieve all relevant information; however, a system that spends too much time searching for reasoning strategies does not have enough knowledge.

Reasoning may be viewed in terms of a movement in a state space in which the states represent all possible situations and the movement is from an initial state(s), representing the current situation(s), to a goal state(s). The reasoning process in solving practical problems typically involves passing through many intermediate states. The allowable transitions between the states are specified either by rules, such as "if..., then..." statements, or via a linking of facts, such as a directed graph. The discussion in the next section on knowledge representation will present various techniques used for state specifications and interstate transitions. A classic example of the state space concept is the game of chess, in which the initial state would be the starting locations of all pieces, and the goal states would be any configuration of the pieces in which all possible moves by the opponent are illegal and the opponent's king is in check. The state transitions would be the rules governing the allowable moves of each piece in each state (each state would have different allowable moves depending on the locations of the other pieces and the proximity of pieces to the edge of the board).

Residence in a given state will most likely present the problem solver with a multitude of possible transitions to other states enroute to a solution. Therefore, search operations also play a major role in the reasoning process. Here, the search is for the path or paths through the state space of a given problem domain, whereas in retrieval it was a search for relevant data in the knowledge base. As stated above, search operations will be discussed following the next section on knowledge representation.

B. KNOWLEDGE REPRESENTATION

Fundamental to each of the types of knowledge discussed above is the problem of how to represent them in a computer in such a way as to

facilitate their interaction, thus producing useful systems. Numerous representations have evolved but most are variations or combinations of the following four: semantic nets, production systems, frames, and logic systems. Semantic networks are very diverse in nature but are generally characterized as being graphical representation schemes in which the graph nodes represent objects or concepts and the links represent inference procedures that relate the nodes. Figure 3 illustrates a very simplistic semantic network that could facilitate arriving at the inference that a Bragg cell has both transverse and longitudinal wave propagation. This type of knowledge is often referred to as declarative, since it is usually derived from factual statements of specific knowledge or relationships.

In their most elementary form, production systems represent knowledge by rules (productions) formulated as "pattern/action" pairs expressed as "If/then" statements. If the "pattern" segment of the statement is true (also known as the antecedent), then the action segment is "fired"; i.e., the state of the machine is modified according to the action specified. Examples of such statements would be: "If the light source is a laser, then it emits coherent illumination," and "If low cost is important and if coherency is not required, then use LEDs instead of laser diodes." Production systems are popular as knowledge representation schemes for expert systems (to be covered in Section II.E), and are in many cases referred to as procedural knowledge, since some action typically results from a rule firing. It should be noted that pattern matching (correlation) plays an important role in production systems in that rule firings are based on "matches" between the rule antecedents (the "if" parts) and the problem states; that is, the matching process determines whether the antecedent is true or false.

Representation via frames involves organizing data by functional groups of hierarchically linked attribute-value pairs. Such a representation is advantageous when dealing with stereotypical concepts such as illustrated in Figure 4. This frame, representing knowledge of a two-dimensional light modulator, might have been referenced by one of several other frames such as ones for optical processing, input/output devices, or

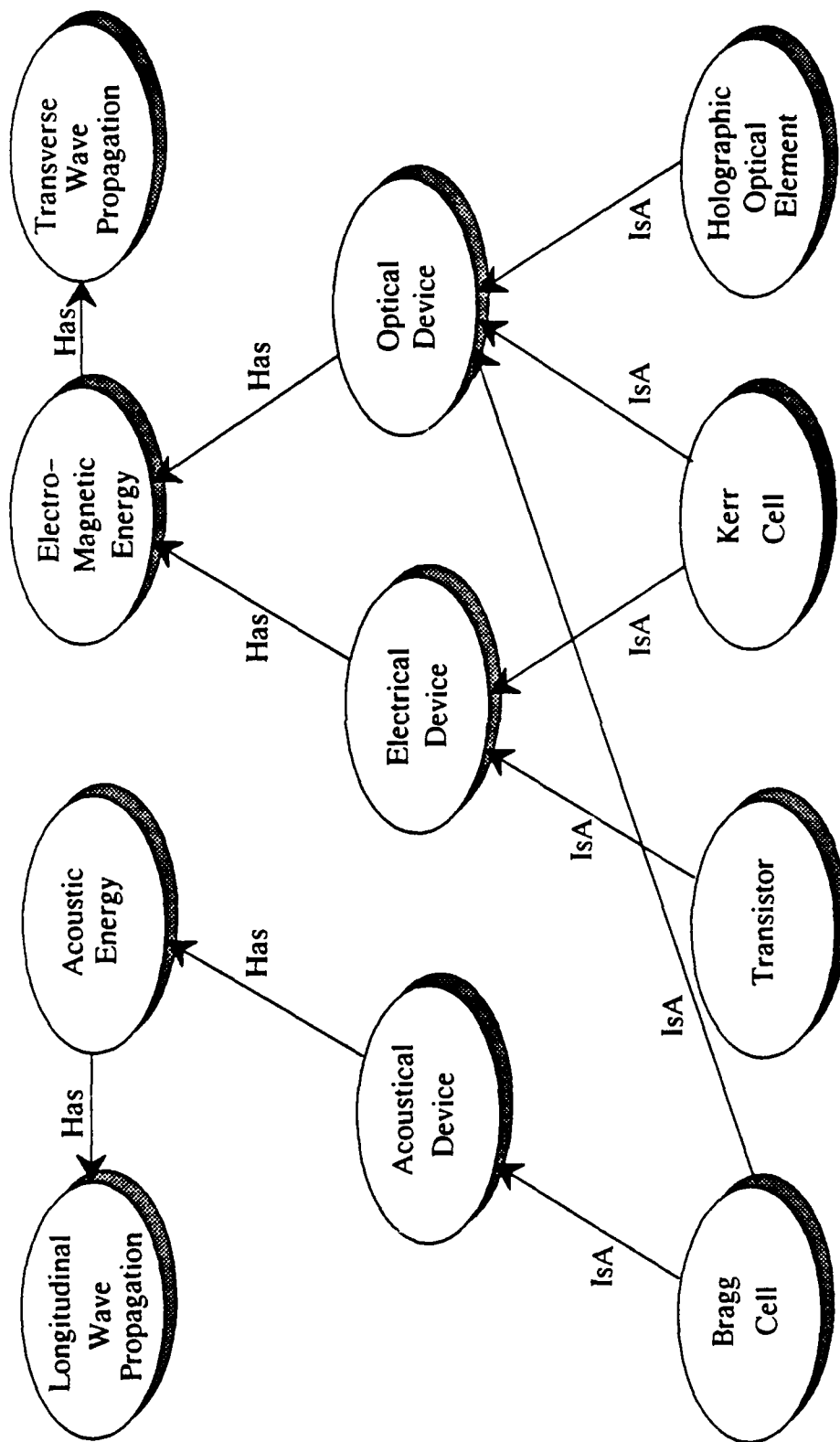


Figure 3
Simplistic Example of Semantic Network

2D Spatial Light Modulator

Specific attributed of 2D SLM's that do not lead to additional frames

Photoconductor:	Silicon
Read/Write Mechanisms:	Optical
Nonlinear Optical Materials:	BSO
Available Sources:	Diode Laser
Frame Rate:	1 kHz
Number of Pixels:	$10^3 \times 10^3$

Figure 4
Frame of Knowledge Representing a 2D Spatial Light Modulator

optical devices. In turn, it references other dependent frames corresponding to each entry in the light modulator frame that has external linkage (shown in the figure by arrows). For example, the non-linear optical material attribute would lead to a frame of information on attributes of such materials. Related to the concept of frames is that of scripts which involve a collection of common sequences of events just as frames involve a collection of related objects and attributes. For example, one could formulate a script for fabricating a light modulator from its component parts.

Logic representation schemes attempt to construct knowledge by the syntactic manipulation of formulas to arrive at TRUE or FALSE premises. The propositional calculus of numerical computing has been extended to provide the formalism known as predicate calculus. Logic is an appealing representation for systems that frequently encounter knowledge base expansion; e.g., such as in theorem proving. The ability to expand the knowledge base results from the power of mathematical deduction to derive new facts from old ones.

Real world relationships are expressed as predicates and their arguments. Predicates represent relationships between things, and predicate symbols are used to stand for these relationships. The predicate applied to its argument(s) returns either a true or false response. The IS-A link between the Kerr Cell node and the optical device node in the semantic net representation shown in Figure 3 would be represented by an ISOPTICALDEVICE predicate in logic representation. The predicate expression ISOPTICALDEVICE (KERRCELL) would return TRUE while ISOPTICALDEVICE (MAGNET) would return FALSE. Other examples of predicates which could be defined (written with variables as arguments for the purpose of generality) are:

WOMAN (x)	TRUE if x is a woman
EQUALS (x,y)	TRUE if x=y
PRESIDENT (x,y)	TRUE if x is president of y

In predicate calculus, it is possible to combine predicates together with their arguments (rather than propositions such as "it is illuminated") using the connectives of propositional calculus including AND ($\wedge$), OR ($\vee$), NOT ($\sim$), EQUIVALENCE ($=$), and IMPLIES ($\rightarrow$ or $\Rightarrow$). In addition, one must introduce quantifiers that assign ranges to variables. For example, the statement EQUALS (x,y) could mean any one of the following four relationships: all x equals all y , a given x equals a given y , all x equals some value of y , or some value of x is equal to any value of y . The universal quantifier $\forall$ means that all values of the variable x are to be considered while the existential quantifier $\exists$ means that some particular value of x is to be considered. For example, if the truth of EQUALS (x,y) is to be based on all values of x being equal to all values of y , then one would express it as $\forall x \forall y \text{ EQUALS } (x,y)$ whereas if the meaning were that the statement is to be TRUE if a specific value of x equals a specific value of y , then one would write $\exists x \exists y \text{ EQUALS } (x,y)$. Examples of expressions written in predicate calculus notation are:

Light is coherent if it comes from a laser.

$$\forall x (\text{LASERLIGHT } (x) \rightarrow \text{COHERENTLIGHT } (x))$$

Some laser illumination is visible.

$$\exists x (\text{LASERLIGHT } (x) \wedge \text{VISIBLELIGHT } (x))$$

Before introducing the final fundamental element of logic, that of functions, let us review the previously introduced elements. These are summarized in Figure 5. Note that the accepted convention is to express variables in lower case alphanumerics and to express the constants and predicate symbols in the upper case. Functions will also be expressed in the lower case.

Predicates are somewhat limiting since their evaluations return only TRUE or FALSE values. For example, the predicates WOMAN (WIFE), WOMAN

DESCRIPTION	EXAMPLES	COMMENTS
variables	x,y	used for general arguments
constants	KERRCELL, MAGNET	specific assignments to arguments
predicate symbols	ISOPTICALDEVICE ()	defined so as to represent relationships
connectives	$\wedge, \vee$,	link symbolic expressions
quantifiers	$\forall, \exists$,	quantify range of variables

Figure 5
Elements of Predicate Calculus

(FLORENCENIGHTENGAL), and PRESIDENT (UNITEDSTATES, GEORGEWASHINGTON) would return TRUE provided that in the latter example PRESIDENT (x,y) was assigned the meaning "y is president of x". WOMAN (GEORGEWASHINGTON) would, of course, return FALSE. Functions, on the other hand, can return objects; therefore, they are used as arguments of predicates. An example would be "wavelength (x)" being defined to retrieve the numeric value of the wavelength associated with x; i.e., "wavelength (REDLIGHT)" would return 0.6 microns and "wavelength (CO2LASER)" would return 10.6 microns. An example of a function as part of a predicate would be VISIBLE (wavelength (CO2LASER)) which would return FALSE since the CO2 laser lases in the near infrared region of the spectrum instead of the visible region.

C. SEARCH

The extensive sizes of the knowledge bases and the state spaces required for solving practical symbolic problems dictate the use of some kind of control strategy for searching either for relevant facts in the knowledge base or for solution paths through the problem state spaces. Figure 6 illustrates the various categories of search, from purely random searching to very domain specific heuristic searching (heuristic implying the use of problem domain knowledge to guide the search). Although this simplified block diagram fits search strategies into specific categories, in actuality one encounters almost a continuum of strategies. The more random that a search is, the longer the time needed to reach the goal. On the other hand, the more heuristic, the more complex are the control procedures. In the limit, the most complex controls would incorporate so much knowledge about the search space that the need to search would be eliminated; i.e., the controls would guide the solution directly toward the goal. Since both extremes are unrealistic, one attempts to find a compromise strategy that is best for the problem at hand. In fact, many AI systems use a combination of general purpose and specific purpose schemes that adapt the overall search toward a solution can adapt to a varying structure of the state space as the search proceeds.

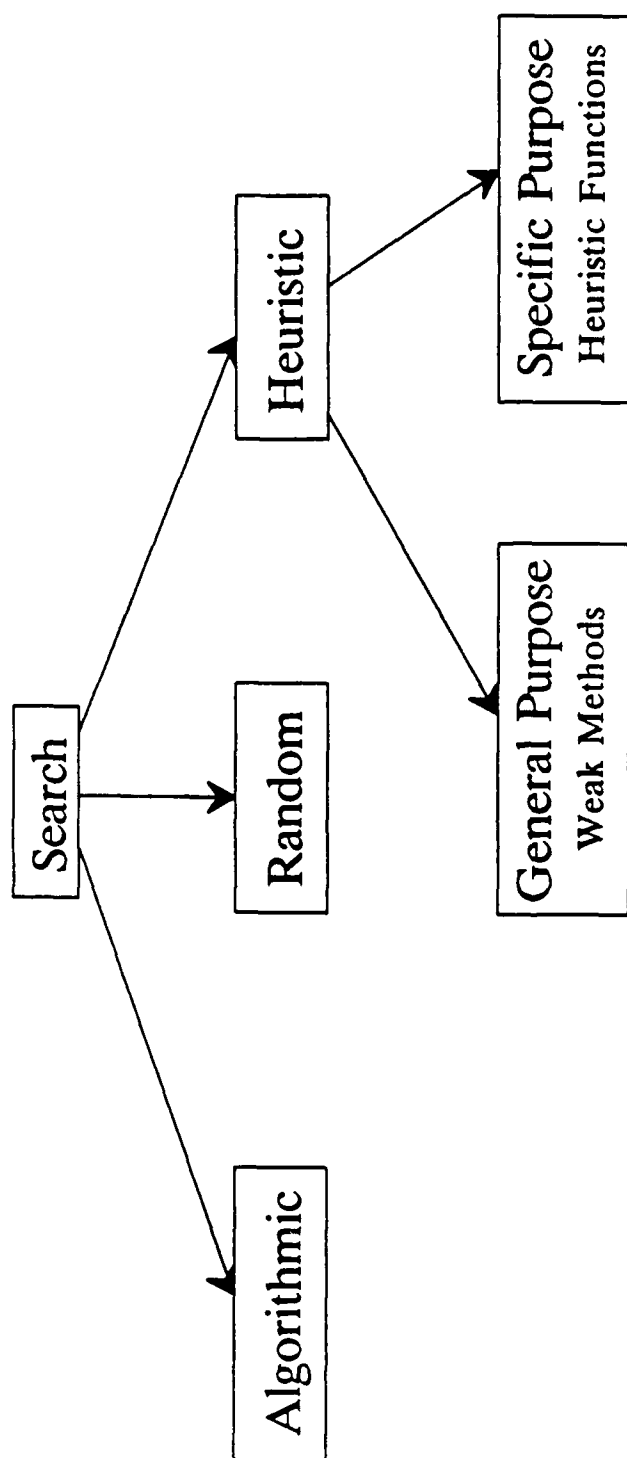


Figure 6
Hierarchy of Search Strategies

Underlying the techniques of search are several general strategies that can be rendered more or less heuristic depending on the degree to which problem domain and goal information are used in selecting various search paths to be tried. Some of these strategies which will be discussed are: tree versus directed-graph, depth-first versus breadth-first, and forward versus backward chaining.

The search process may be accomplished by following either a directed-graph structure, such as illustrated in Figure 7a, or the corresponding tree structure shown in Figure 7b. The directed-graph search remembers all tried strategies so that the search can return to an earlier problem state (graph node) and continue the search along another path. The tree search, on the other hand, may duplicate efforts in pursuing strategies that were already tried. For example, in the push toward a viable solution, the "C" node might have to be expanded a second time (let us say via A C F ...) if the first attempt (via A B E C) did not reach a goal. The obvious disadvantage of the tree is the possibility for redundant effort, but the advantages are the use of much less memory in not having to store the previous strategies and the freedom from running a test on every generated node to determine if it matches a previously generated node. The choice of tree versus graph is often decided by the nature of the problem being solved; i.e., how frequently are repeated states likely to occur. The availability of memory will also be a deciding factor.

Search schemes may also differ with respect to the order in which the nodes are searched. A strictly serial search, known as depth-first, pursues a given strategy (e.g., branch of a tree) until the strategy has either succeeded in reaching a goal or has been shown to terminate in an unsuccessful search. In the latter case, one backtracks to the most recently traversed node that possesses a yet untried branch. This search procedure saves on memory since unsuccessful branches can be discarded; however, if the problem domain is characterized by long search paths, the depth-first search could be exceedingly costly in terms of time. An example of a depth-first search through the tree illustrated in Figure 7b might be A B D B E C A C F G ...

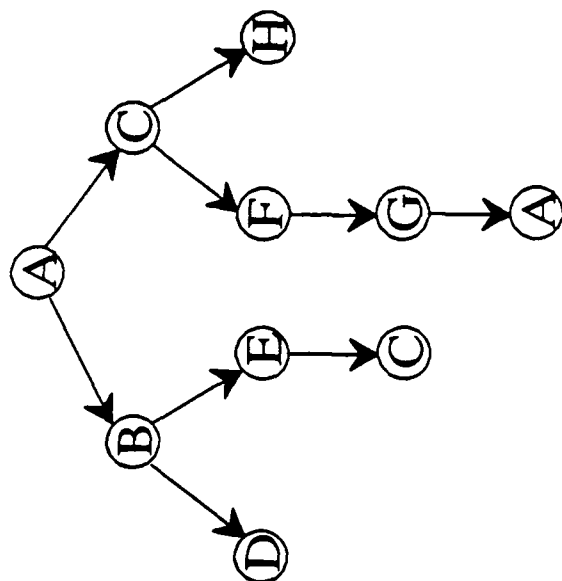


Figure 7b
Tree Search

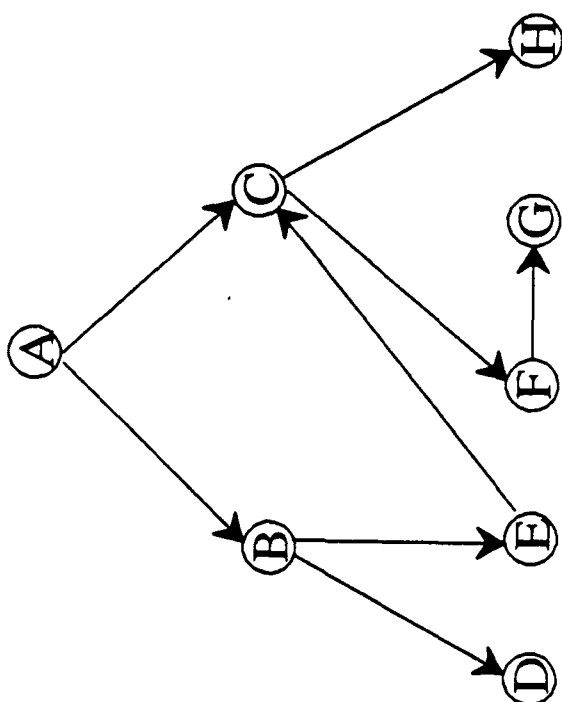


Figure 7a
Directed Graph Search

In the breadth-first search, on the other hand, all states linked to the starting state are tested to see if they match the goal state before proceeding any deeper into the tree or graph. If a goal has not been reached, then all of these first level states (or nodes) are expanded; i.e., all of the linked states are generated and tested. For example, the breadth-first search through the tree of Figure 6b would proceed via the nodal order A B C D E F H C ... until a goal state is reached. This type of search could have great utility in parallel processing systems, such as the types that will be discussed in Section IV.

Up to this point, problem solution and reasoning have been presented as a procession from a starting state or states toward the goal state(s). Another option that we as humans sometimes employ is to start with the goal and proceed backwards in an attempt to satisfy the initial conditions. In backward reasoning, or backward chaining as it is known in the AI community, one first generates one or more states that could produce the goal state and tests to see if a match exists with the initial state(s). If not, the search continues on backwards. In production systems, this means that the "then" parts of the rules are matched and the "if" parts are fired (i.e., each "if" part is used to generate a more forward node). There are two factors which would strongly influence the choice of backward chaining over forward chaining - the branching factor going backward versus going forward, and the number of goal states versus the number of initial states. That is, if the generated search tree branches out significantly more in the forward search than for the backward search for a given problem domain, then backward chaining would be preferable for such problems. If the branching is approximately the same in both directions, the number of goal states versus the number of initial states becomes a deciding factor. Backward chaining looks more appealing in solving synthesis-type problems for which there exists a broad spectrum of objects from which to synthesize. An example would be the determination of which material characteristics are needed to optimally realize a specific device. Here it would be better to start with the goal state (the device requirements) than to start with the sets of characteristics for all possible materials.

Chess, on the other hand, could never be reasoned through via backward chaining due to the extremely large number of ways to attain checkmate.

For many practical problems, the state spaces are so large that searches for guaranteed optimum solutions are sacrificed in favor of incorporating strategy and tactics into constraint of the search process and being satisfied with a good solution rather than necessarily the best. Search processes involved with the playing of chess are an excellent example of this; otherwise, the chess problem could not realistically be solved. As previously mentioned, varying amounts of information can be provided in order to judiciously guide the various search schemes just discussed. Techniques such as indexing, factorization, and template matching fall under the category of general purpose heuristics; however, their effect in constraining complex searches is characteristically weak, often necessitating the use of more complex heuristics in conjunction with these more general purpose ones. Indexing involves using a predetermined scheme to assign indices to problem states and storing the rules applicable to each problem state in such a way that they can be associated with the index for that state. Residence in a particular state can then invoke the index for that state, which, in turn, can call up all of the rules that could apply. More appealing is the operation of factorization. If the knowledge base is divisible into broad sets which have small cross-correlations, entire sections can be ignored by considering appropriate problem domain information. For example, if the question at hand deals with diagnostics for lung diseases, the system does not need to search any part of its knowledge base dealing with procedures for use in actual lung operations. Search constraint can also be achieved via template matching, analogous to classical pattern matching where the matching is used to verify that the correct shape, word, etc. has been found. Template matching is frequently used in speech recognition, natural language understanding, and image understanding, all of which will be discussed in Section III.

One can get more quantitative in search constraint by utilizing some kind of heuristic (or evaluation) function. This falls under the class of specific purpose heuristics as denoted in Figure 6. The heuristic function

generates a weighting or cost factor to be associated with each node that is some measure of the "goodness" of the solution path to that point. Different types of problem domains have different opportunities for defining such functions (thus the notation "specific purpose"); however, frequently used measurement concepts are: a metric representing the length or difficulty of the search to the node in question, or a metric representing the distance to the goal node or difference between the current node and the goal node. As mentioned earlier, there will always be a tradeoff between the search time saved by the heuristic functions versus the time needed to compute the functions themselves; i.e., complex functions may provide excellent guidance for the search, but the time needed to compute them may be more than the time that would have been used by a more random search.

Once a measure function has been defined, the search process can proceed along what is known as a best-first search. Using this technique, the nodes to be expanded at any given point in the search are the ones with the best "goodness" measure. For example, consider the number beside each node in the tree of Figure 8 to be the value of the heuristic function such that the lower the number, the better to expand that node. Then the best-first search would proceed as follows: A C G H B D K E F I J L.

D. ATTRIBUTES OF SYMBOLIC COMPUTING AS COMPARED TO NUMERIC COMPUTING

At this point, the reader should be gaining a cursory view of what constitutes symbolic computation. This understanding can be enhanced further by making direct comparisons between symbolic computing and the more familiar numeric computing. Figure 9 lists many of the attributes of the two computational methods in such a way that a comparison can be made between corresponding attributes in each column. The comparisons are discussed below.

The logical inference is the analogue of the floating point operation. Whereas the floating point operation is basic to the manipulation of numeric symbols, the inference operation is used to manipulate the much

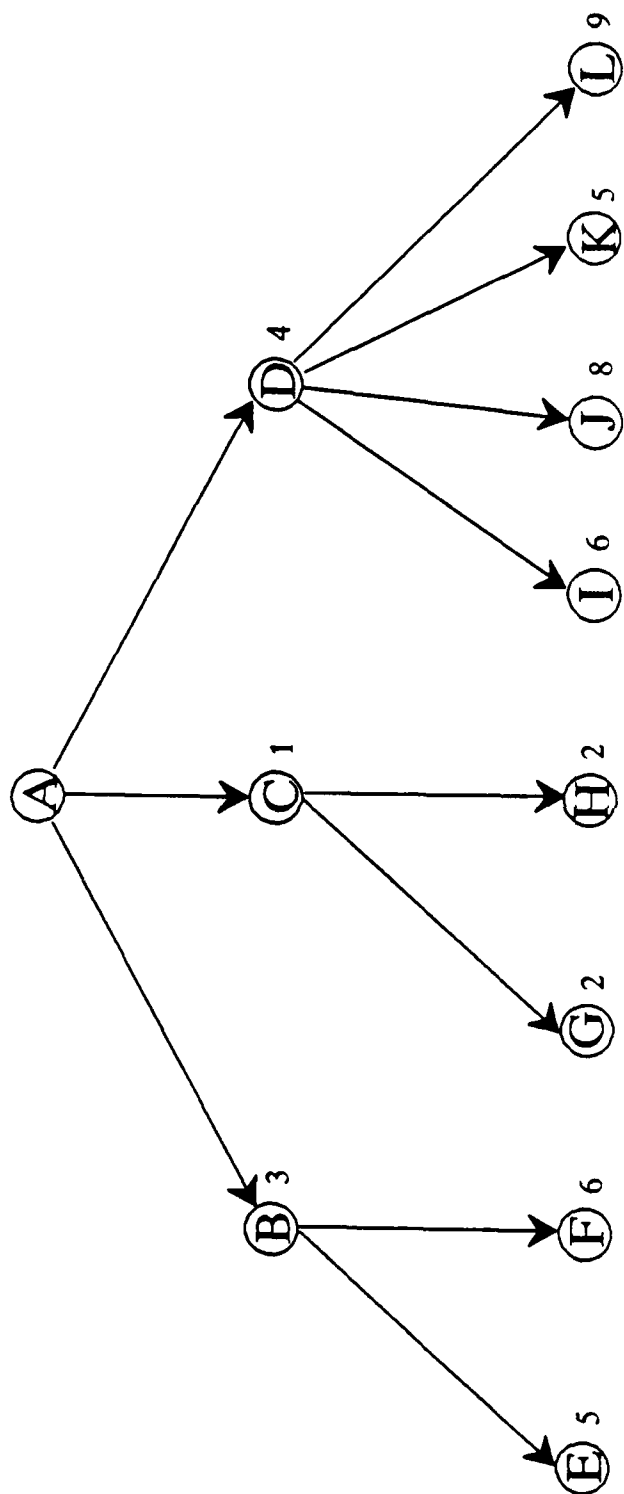


Figure 8
Best-first Search

<u>SYMBOLIC</u>	<u>NUMERIC</u>
Logical Inferences	Floating Point Operations
Dynamic Data Structures	Static Data Structures
Heuristic Search	Algorithmic
Inexact, Incomplete Knowledge Acceptable	Correct Complete Data Required
LISP, PROLOG, ...	FORTRAN, PASCAL, COBOL,...
Independence of Control and Problem Knowledge	Integration of Control and Information
High Level Modification	Macro Level Modification
Explanatory Capabilities	Limited Explanation

Figure 9
Attributes of Symbolic and Numeric Computation

broader class of symbols encountered in symbolic computations. A logical inference is generated by combining knowledge elements or groups of objects to reach a conclusion. Whether the logical inference is done using syllogisms (cascaded "if/then" statements), graphical linking, frame matching, or logical inference techniques depends considerably on the knowledge representation used (production system, semantic network, frames, or first-order logic, respectively). But it is the combination and manipulation of symbolic data structures (e.g., objects and their attributes) by logical inferencing that is the basis for most reasoning techniques in symbolic computation.

The well-structured data formats of vectors, matrices, etc. used in numeric computing give way to data structures that can change their shapes in symbolic processing. In performing tasks such as navigating over unknown terrain, playing a game, or carrying on a dialogue, one knows prior to performance of the task only the general form the data must take to represent the route planning, the responses to the opponents' actions, or the meaning of the spoken phrases, respectively. Therefore, symbolic processors use lists of objects connected by pointers, such as those discussed in previous sections of this chapter. The input data, the data manipulated, and the end point in the manipulation of that data are usually a phrase, a concept, or some other symbolic structure of unconstrained size and/or shape, rather than data structures with specified dimensions as in numeric processing. Further, dynamic data structures allow the machine to deal with inexact information or to arrive at uncertain or inexact conclusions.

This use of inexact or incomplete information gets to the heart of the differences in using symbolic versus numeric data. The algorithms used on numeric computers have little ability to generate correct numerical output from qualitative descriptors or from incomplete numerical input data. They depend upon exact data since all fundamental operations are combinations of numeric entities. On the other hand, it is a feature of most symbolic representations that conclusions can be reached in a qualitative sense by generating relations among the functional attributes. This is possible

even when the input information is incomplete, or, in some cases, inexact. It should be noted that such techniques are at the forefront of today's research in symbolic computation, and comprise a discipline called reasoning with uncertainty (ref. 6).

It should be obvious that in order to take advantage of the power of symbolic computing, such as the use of inexact or incomplete knowledge in reasoning, specialized programming languages are required. We should note, however, that the numeric programming languages, such as FORTRAN, could, in the hands of clever programmers, be used in symbolic computing. However, they have not been optimized for such use and would make symbolic programming a very cumbersome task. The LISP language, and its major derivatives of INTERLISP and MACLISP, create a programming environment that facilitates the manipulation of symbolic expressions characterized by flexible data structures. The semantic meanings of objects are readily changed by adding and deleting the variable lists of attributes. Another notable characteristic of LISP is its recursive nature - any LISP function can call itself and any program can be defined in terms of itself. Such a capability facilitates the search of lists of indefinite length in the search for certain elements (attributes) in the lists.

A very different language is PROLOG (PROgramming LOGic). Based on production rules, it uses pattern matching techniques to prove or disprove program statements. The language relies heavily on predicate calculus in establishing relationships between objects. The fundamental operation in PROLOG is the logical proof of some condition or relationship starting with a set of more primitive conditions.

The sixth element in Figure 9, the independence of control and problem knowledge, represents a drastic difference in the way symbolic systems process information. In symbolic computation, the control refers to any process, explicit or implicit, which governs the order of problem solving activities.^{3,4} A key aspect of this occurs in expert systems (discussed in Section III.E) where the actual structuring of the solution strategy can be changed recursively based upon changes in the program's evolution. This is very different from a numeric computation, where changes, even conditional

branches within a program, are input "a priori" to the system. It is this independence of the knowledge base from the control activities that allows an expert system "shell" to be robust enough to be applied to more than one problem domain. As an example, although the operating program MYCIN was originally developed to aid in the medical diagnosis of bacterial infections, it may be applied to a knowledge base of crystallographic information to aid crystal growers. Instead of containing rules relating symptoms, bacteria, and remedies, the knowledge base would contain rules relating measurements, crystallographic structure, and recommendations for regrowth. In the numeric domain, one cannot take a program written for, say, VLSI design and adapt it for lens design with only a change in the input data.

Even the way one makes changes to symbolic programs differs from techniques in the numeric domain. If a change is required in the program of a numeric computer, the entire program, or at least the macro in which the change is to be made, must be pulled up, edited, and returned to the system, at which time the macro or the entire code must be recompiled. In programming environments built upon LISP one can modify the program without such activity. LISP programs and data sets are both written in the same syntax and form. As a result, a LISP program can manipulate (alter) another LISP program or data base. It can automatically modify faulty rules or knowledge, or it can call suspect rules or knowledge to the attention of the operator who can then make changes interactively without having to recompile the entire program. These environments are very powerful, allowing the operator to slide a window through the program which allows one to see in real time the outcome of a change in the knowledge base or rule structure anywhere in that window. At the present time, this form of software engineering is not available in numerical computers, for which the time-consuming debugging cycle is: find bugs, rewrite, recompile, rerun, look for new bugs, rewrite, recompile, rerun, etc.

If making changes to symbolic programs differs from numeric practice, then it is not altogether unexpected that the power of symbolic computing can be used in the debugging phase of program development as well. It is

certainly possible to build into a numeric computer program the ability to print out messages indicating which "if/then" paths were taken during execution of the program. In a symbolic computer, however, not only can the machine trace for the operator the path taken to the solution, but also it can tell the operator why it took each path and not others. This feature is useful for several reasons, only two of which are stated here. First of all, it is a valuable diagnostic tool; in fact, it is an integral part of most development environments designed for use with LISP machines. Second, human beings want to know "why?"; that is, we tend to ask how some conclusion was reached by another human in order to form some opinion about its validity; and one would not expect a machine replacing a human expert to be exempt from having to justify its own conclusions. This leads the user to a feeling of confidence in the machine.

Having looked at the similarities and differences between symbolic and numeric computing, the reader can begin to understand some of the ways in which symbolic computing can be applied to real-world problems. The earlier discussions on knowledge representation and search strategies will be built upon in the next section, where the reader will see both how these techniques are specialized and modified as well as how they can lead to performance problems. This is particularly true with the search process, which is very often the cause for the computational bottlenecks in AI systems, and the discovery of ways in which optics can impact these operations may be a key to optical symbolic computing. Before discussing these opportunities, let us take a look at the symbolic computing domains of speech recognition, vision/image understanding, natural language processing, and expert systems.

SECTION III
SYMBOLIC COMPUTATION: FUNCTIONAL CAPABILITIES

The previous section highlighted the underlying principles of symbolic computation, providing both an introduction to the subject and a discussion of the types of representations employed and search strategies utilized. This section will extend that discussion by first presenting an overview of the four main disciplines which comprise symbolic computation: speech understanding, vision, natural language understanding, and expert systems. These applications can take several forms, each of which draws upon aspects of knowledge acquisition, reasoning, and retrieval. If they have one aspect in common, though, it is the use of knowledge, about the system or domain under consideration, to improve the machine's understanding of input information.

A. OVERVIEW OF SYMBOLIC COMPUTING DOMAINS

As in any new field, there is considerable debate as to what constitutes AI and what the appropriate taxonomy of AI disciplines should be. Complicating matters is the fact that the overall nature of the programming tasks and the optimal computing structures for it are still not well understood.¹⁻³ Nevertheless, major advances have been made in applying the knowledge-based techniques presented in the previous section to a wide variety of problems. As a result, symbolic computing research is currently concentrated in four areas which form the basis of generic machine understanding capabilities: speech recognition and understanding, vision or image understanding, natural language understanding, and expert systems and reasoning.

To enable a machine to respond and identify spoken language, we can apply advanced pattern recognition techniques to process the input signal and recognize the words. This aspect of speech research is termed speech recognition. Language is typically entered into a speech recognition system by means of a microphone.⁴ Recognition can occur in any of several

ways, but each typically involves performing a digital comparison of the input phrase or sentence with elements stored in the computer's memory. Speech understanding, on the other hand, focuses on the use of knowledge to attach meaning to a series of spoken inputs. Although no real boundaries exist, in most cases speech recognition is referred to as low-level speech, emphasizing signal processing and template matching; speech understanding, by virtue of its reliance upon knowledge processing, is termed high-level speech. The goal of all speech research, obviously, is to achieve totally speaker independent, high accuracy recognition, over a large vocabulary base.

Vision, or image understanding, as it is more commonly known in computer science, refers to the ability of a machine or computer to understand scenes utilizing a visual input. Such systems have the goal⁵ of pattern and image understanding with a degree of accuracy that parallels human vision systems. Recognition can be accomplished in many ways, but most involve matching elements of an observed scene to objects represented in the system's knowledge base. This is similar to the problem for speech understanding systems, except that the information is comprised of input images rather than waveforms, and the objects themselves are three dimensional. Since most visual input devices are two-dimensional imaging arrays, such as solid-state TV cameras, much emphasis has been placed on achieving some level of full three-dimensional information from the input scene. By analogy with speech, image processing functions such as preprocessing, image restoration, or gradient calculations are termed low-level vision. Any processing requiring interactions with the knowledge base are known as high-level vision.

Natural language understanding focuses on the ability of a machine to attach meaning to English language (or some other written language) phrases input to it through a peripheral device, typically a keyboard (see Figure 10). To be effective, therefore, natural language understanding systems should incorporate knowledge of linguistic theory, such as sentence decomposition according to the syntax of the language (parsing) and assigning meaning to words or phrases (semantics).⁶ Thus natural language



Figure 10
A Natural Language Understanding System
(Courtesy of BBN)

processing is a very knowledge intensive activity, relying upon knowledge of the grammar and the context to attach meaning to the input sentence or phrase. Knowledge of these and other linguistic attributes must be included within the system's knowledge base, and the challenge, as in other AI disciplines, lies in resolving and implementing one of many different strategies for interpreting input, such as English sentences.

Expert systems are a discipline within applied artificial intelligence which seek to emulate human expertise in specialized areas, known in AI parlance as domains. Examples include the interpretation of spectral data (the DENDRAL project),⁷ the configuration of minicomputers from components (the RI project),⁸ and the diagnosis of diseases in internal medicine (the MYCIN project).⁹ These computer systems "achieve high levels of performance in task areas that, for human beings, require years of special education and training."¹⁰ Here, expertise is defined as the "set of capabilities that underlies the high performance of human experts, including extensive domain knowledge, heuristic rules that simplify and improve approaches to problem solving, metaknowledge and metacognition, and compiled forms of behavior that afford great economy in skilled performance."¹⁰ (The prefix meta- refers to knowledge about the root word.) These systems have been very successful recently¹⁰ in solving problems and tasks which are knowledge and heuristic intensive, those that would typically take a human expert between 8 and 40 hours to accomplish. Using the capabilities of LISP machines (see Figure 11) and specialized software development tools, expert systems research seeks to capture human expertise accurately enough to apply it to more complex, demanding, and diverse tasks.

These functional capabilities are presented in this order for several reasons. Historically, the first successful applications of AI research centered on the human/computer interface, either spoken or visual, and on game playing by machines. (For a nice review of early AI research, the reader is referred to reference 6, Vol. 1) This has evolved into what we now know as low-level processing, comprised mainly of signal and image processing, with knowledge based techniques naturally assigned the role of

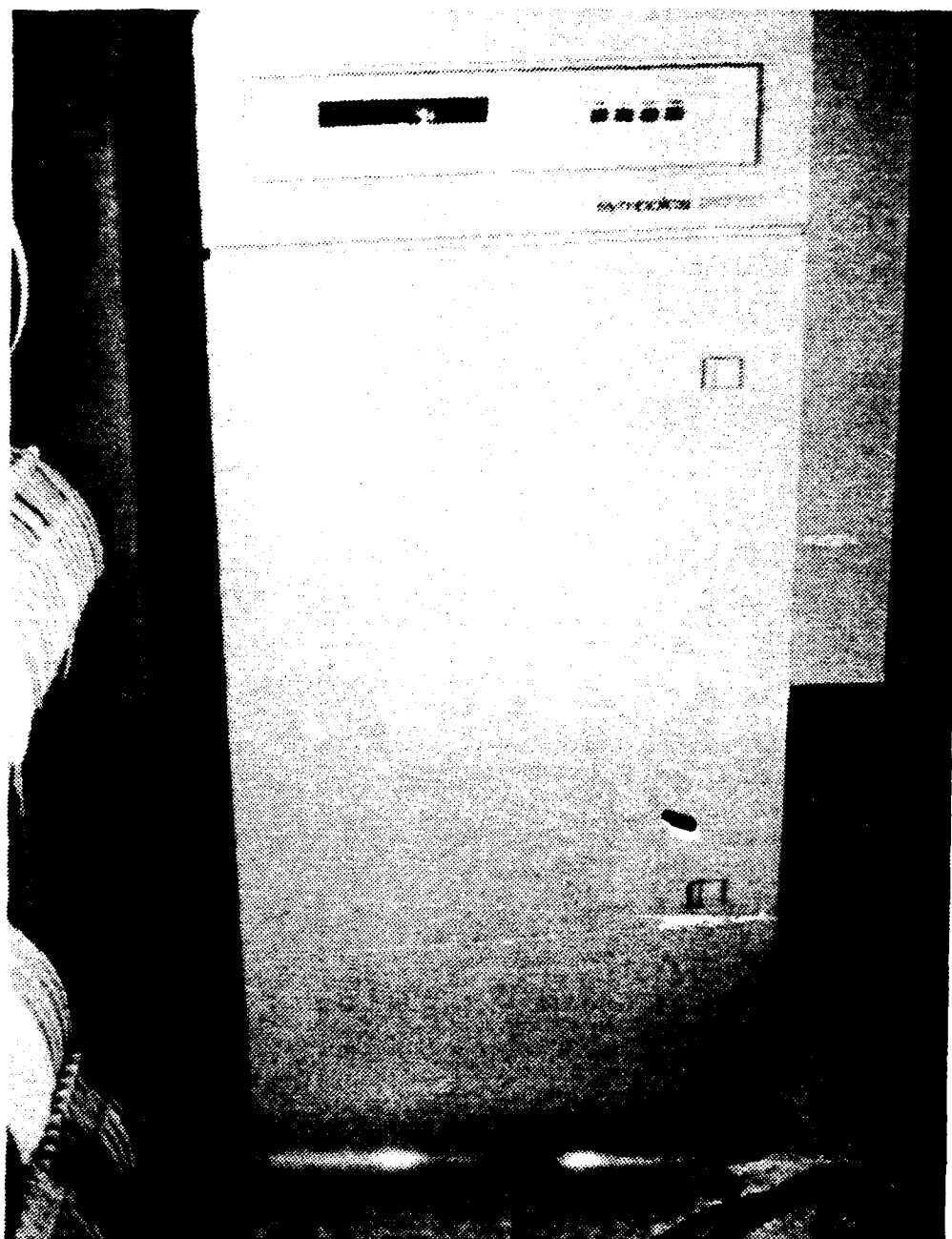


Figure 11
A LISP Machine
(Courtesy of Symbolics, Inc)

higher level processing. By virtue of their long history, speech and vision research are thought to be more mature discipline than natural language or expert systems. As an example, many of the concepts required for natural language processing had their genesis in linguistic theory and in higher level speech research. And the explanation facilities in expert systems have in turn developed out of natural language research.

Second, there is a natural evolution from capabilities which primarily focus on the man-machine interface to those which emphasize computational reasoning with minimal human interaction. Finally, this will allow us to minimize the amount of overlap and repetition of concepts. In the ensuing discussions, we will see a great deal of commonality between each of these disciplines, and in many cases this is a result of critical ideas transitioning from one area to another. The use of the blackboard architecture (to be discussed in Section III.B) is a case in point. Originally developed for the Hearsay speech understanding system,⁴ it has now been successfully applied in natural language and expert systems.

In what follows, each of these symbolic computing capabilities will be discussed in greater detail, with an eye towards identifying the underlying technology. In doing this, the major challenges to current research in each field will be highlighted. Each subsection will conclude with a review of current and projected machine intelligence capabilities, and an analysis of the problems and computational bottlenecks impeding the progress of each AI research area.

B. SPEECH UNDERSTANDING

Most speech understanding activities attempt to endow computer systems with the required knowledge and auditory discrimination to achieve real-time machine understanding of continuous spoken input. Unfortunately for computer systems, humans have not standardized on any particular form of word pronunciation. Thus, even systems which seek to recognize speech input (but not necessarily understand it) are faced with the problem of distinguishing words independent of speaker, the rate of speech, any

dialects or accents, vocabulary, dropped syllables, and last, but not least, background noise environment.

This is obviously a formidable problem, since humans themselves do not yet have a reliable, 100 percent speaker independent speech understanding system. We have problems understanding speech in which adjacent words are run together, influencing the pronunciation, as is typical with geographic references such as Long Island, which is often spoken as Longoilan. We still have trouble with different dialects, e.g., the word oil in various sections of the country is pronounced as all or as ole, and very often we have to infer words or missing phrases from context. This last point embodies the critical challenge for knowledge retrieval and reasoning in speech systems - the use of pattern matching techniques for word identification, and knowledge for understanding meaning and for interpretation. The question is how to apply the knowledge, and when (see Figure 12).

Many techniques have been developed for knowledge-based interaction in speech, but the two most developed are the isolated word recognition (IWR) systems and the continuous speech understanding (CSU) systems.¹¹ The isolated word systems focus in on word identification, using segmented spoken input, such as:

"helium-neon...laser...broken"

or:

"weather...today...rain"

to decrease problems associated with identifying the beginning and ends of words. The problem of identifying the word from the transformed acoustic input still remains, but is handled by template matching. This process, shown in Figure 13, involves taking the input signal and identifying key features, such as the major sound variations and the beginnings and ends of words. The next step is dynamic time warping, which seeks to align the

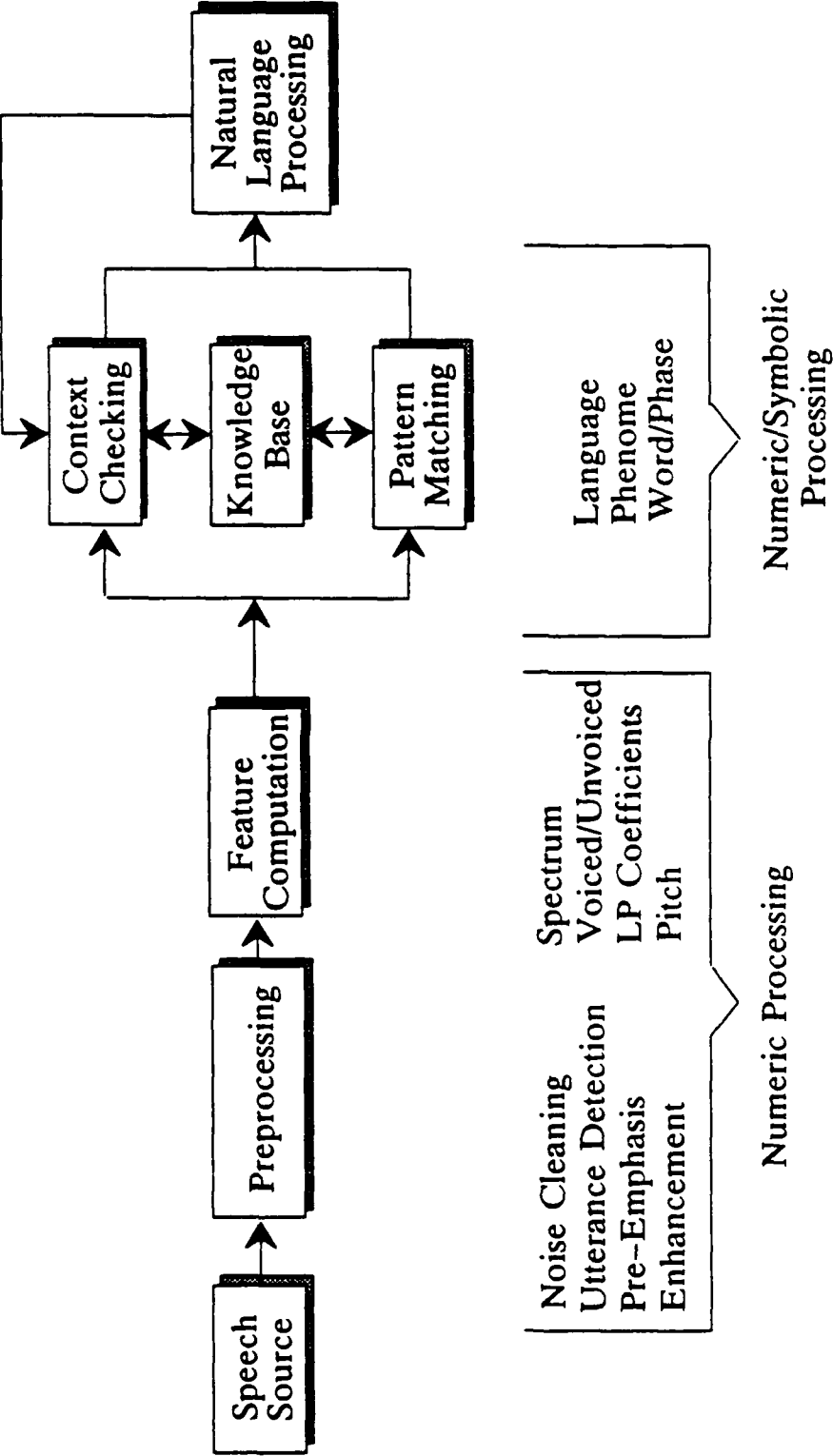
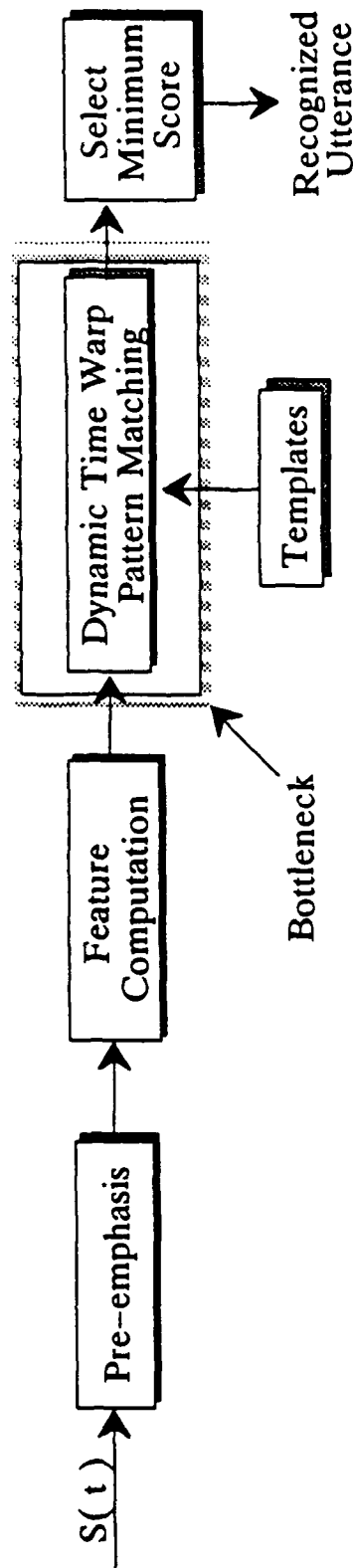


Figure 12
Speech Understanding Paradigm



(Weighted Euclidian
for Filter Bank Features)

$$\sum_{i=1}^{N_F} (D_i - T_i)^2 W_i$$

(Itakura Metric)

$$\frac{\sum_{j=1}^{N_F} \sum_{i=1}^{N_F} D_i R_{ij} D_j}{\sum_{j=1}^{N_F} \sum_{i=1}^{N_F} T_i R_{ij} T_j}$$

Number of Features
= $N_F \sim 8-15$

Pattern Matching Metrics:

Figure 13
Low-Level Speech Processing

length of the spoken word with some internal reference length. As an example, the words helium-neon and Caribbean are often pronounced as:

"helium-neon"	"Care-ih-be-yan"
"heel-e-yum nee-un"	"Cah-rib-e-an"
"heelyum-neun"	"Cah-rib-be-yun"

each of which is spoken at a different rate. To standardize on the lengths of words, the signal, as a function of time, is stretched or compressed, in order to make it compatible with an internal reference length. The features of this "standardized word" are then compared with templates stored in memory, and weighted metrics, such as those shown in the figure, are used to match the word with its counterpart in memory.

It is here that we see the greatest bottlenecks in the low-level process. To give an idea of the computation rate, there are approximately 500 metric computations typically performed per vocabulary word using dynamic time warping during a 500 msec utterance.¹³ Using this as a basis, the total number of multiplies (or inner product computations) per word ranges from 5 K to 10 K, using the weighted Euclidean metric for filter bank features¹⁴ and the Itakura metric for linear predictive coding (LPC) features.¹⁵ Therefore, an optimal inner product processing environment, such as optics, should have great utility in speech recognition.

The template match is not the only computational bottleneck, unfortunately. The application of high-level knowledge, which is a retrieval and reasoning problem, also limits the accuracy and timeliness of the system, preventing real-time operation. Figure 14 gives an example of how knowledge is used to obtain meaning from the input phrase. Here, successive matchings with words in the network allow the machine to interpret the phrase. The reader should note that this is just one of a number of possible representation techniques, and speech systems have evolved which have used frames, scripts, and rules as well as semantic networks.

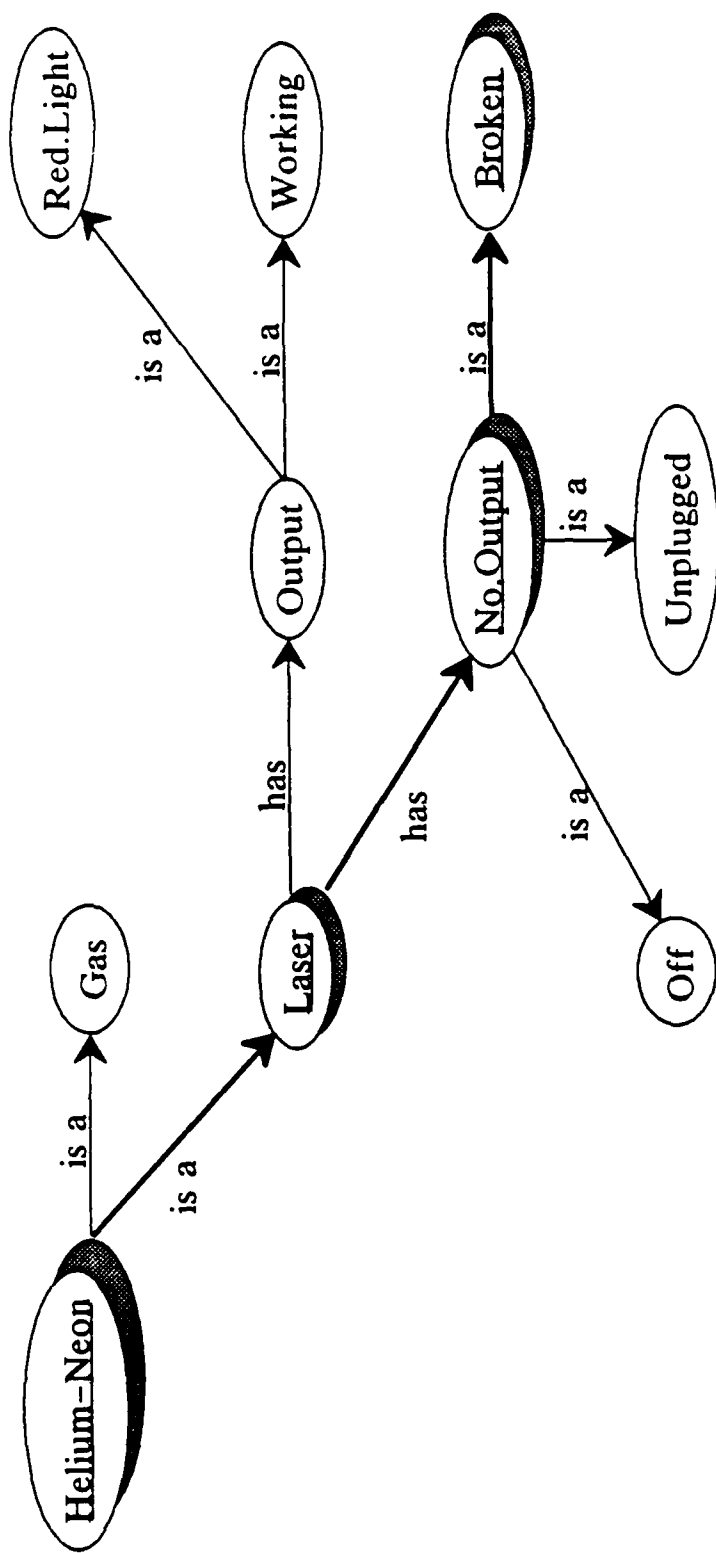


Figure 14
Semantic Net for Speech Recognition

There are many types of knowledge that can be used in understanding speech. Many of these are derived from linguistic considerations, and we can therefore expect that they will be important in the discussion of natural language as well. Each of these is specific to the vocabulary under consideration, and is heavily dependent upon the rules associated with the grammatical structure of the language. Thus it is important for the machine to understand that adjectives modify nouns, adverbs modify verbs, ...etc.

The most obvious, and probably the most familiar type of knowledge, is phonetics, which refers to the physical characteristics of the sounds of each word in the vocabulary; it is the acoustic signature of the words. Another important type of knowledge is morphology, which refers to the ways in which the basic units of words (basic morphemes) can be combined to form new words, plurals, tenses, etc. Using these types of knowledge as a basis, higher level knowledge attributes can be used to determine meaning from spoken input. In the introduction to this section, we introduced the concepts of syntax (the sentence structure and grammar) and semantics (the ways in which word meanings are combined to form the meanings of sentences and phrases) in the discussion on natural language. It should be clear that these types of knowledge are necessary for any speech recognition system as well.

The last type of knowledge, and possibly the most important for high-level speech, is pragmatics. Pragmatics refers to the rules of conversation and dialogue, which allows the system to distinguish questions and queries from factual statements. In the following examples, the statements all have the same syntactic and semantic meaning. yet each represents a different type of interaction with the machine:

- (1) "The part in the helium-neon laser is broken."
- (2) "Is the part in the helium-neon laser broken?"
- (3) "What part of the helium-neon laser is broken?"

The first is a simple factual statement, whereas the second is a rearrangement of the first, requiring a yes or no response from the system. In the third, however, a yes or no response is insufficient; another level of

response is called for from the system. It is the pragmatic knowledge of the vocabulary that allows the system to recognize and understand differences in sentence and phrase meanings.

These types of knowledge could have been incorporated into our earlier IWR example, but they were not required. The scheme for using knowledge in the IWR example is very general and is applicable to most types of knowledge, and is referred to as the bottom-up hierarchical paradigm. Here, bottom-up implies the use of numerical techniques at the start of the algorithm to improve the signal and perform all feature extractions, which collectively are known as low-level speech processing (see Figure 12). The high-level processing is the use of knowledge, and in this case occurs after initial low-level processing has been completed. Hierarchical refers to the passage of system control from the low-level, through a series of intermediate, sequential steps, to the point where the knowledge interactions occur.

This paradigm is not unique, and as we shall see in our discussion of continuous speech (below), can be structured as a top-down strategy, or even a middle-out scheme. In each case, however, the overriding consideration is where and when the knowledge about the problem is retrieved and utilized. If it is at the beginning, chances are that we are dealing with a top-down paradigm; if it is at the end, as in the case of the isolated word example, it is a bottom-up scheme.

In continuous speech understanding, grammatic or linguistic models are used to constrain the knowledge retrieval process, which means limiting the active vocabulary or the number of words possible at any instant in time in a CSU system. Not only is processing time saved by limiting the size of the search space for each word, but the potential for confusion is lowered and a higher recognition rate results.

The basic CSU system consists of an acoustic processor, which carries out the initial signal to symbol transformations, and a linguistic decoder, which applies knowledge to understand the spoken input. Typical CSU systems take two forms: recognizing individual words in random arrangements, such as for data base retrieval, and understanding meaningful,

continuously spoken sentences. The two have some features in common with IWR, as well as a few differences, but the principal difference is in the area of word segmentation.

Word segmentation, or determining the beginning and end points of individual words, is a critical problem in CSU. Since words spoken in continuous speech tend to run together, the end-point detection algorithms used in IWR will not work. Several methods exist for overcoming this problem, and all are variations on the dynamic time warping algorithm discussed earlier. This still remains as a processing bottleneck, although not as severe as the template match or the knowledge retrieval process.

Most continuous speech understanding systems are top-down, knowledge directed systems, using knowledge about the speaker's problem to limit or constrain the expected content of the input.¹¹ Such a technique has been successfully implemented in the Hearsay-III system,¹² where each type of knowledge can interact with the partially processed input signal independently of the others, as shown in Figure 15. The novel feature of this system's architecture is the blackboard, an area in memory allocated for communicating between the various knowledge "experts." In fact, by allowing each type of knowledge to operate on the input signal, we have in fact developed individual speech "expert systems" (see Section III.E). This blackboard concept, which allows cooperation among multiple knowledge bases, has been successfully applied to both natural language understanding systems and to expert systems.

The use of heuristic search to constrain the input is not without its price, and that is the tradeoff between generality and performance. This will be a theme that we will see throughout this section, and one which underlies all AI research at the present time. To appreciate this tradeoff, the reader should recognize that the vocabulary of the English language is very large - the 24 volumes of the Oxford English Dictionary are a testament to that. Yet most human beings have a usable vocabulary much smaller than that, but even 20,000 words are staggering when one looks at the number of possible combinations and pronunciations associated with each word. Early speech systems, which sought to include most words within

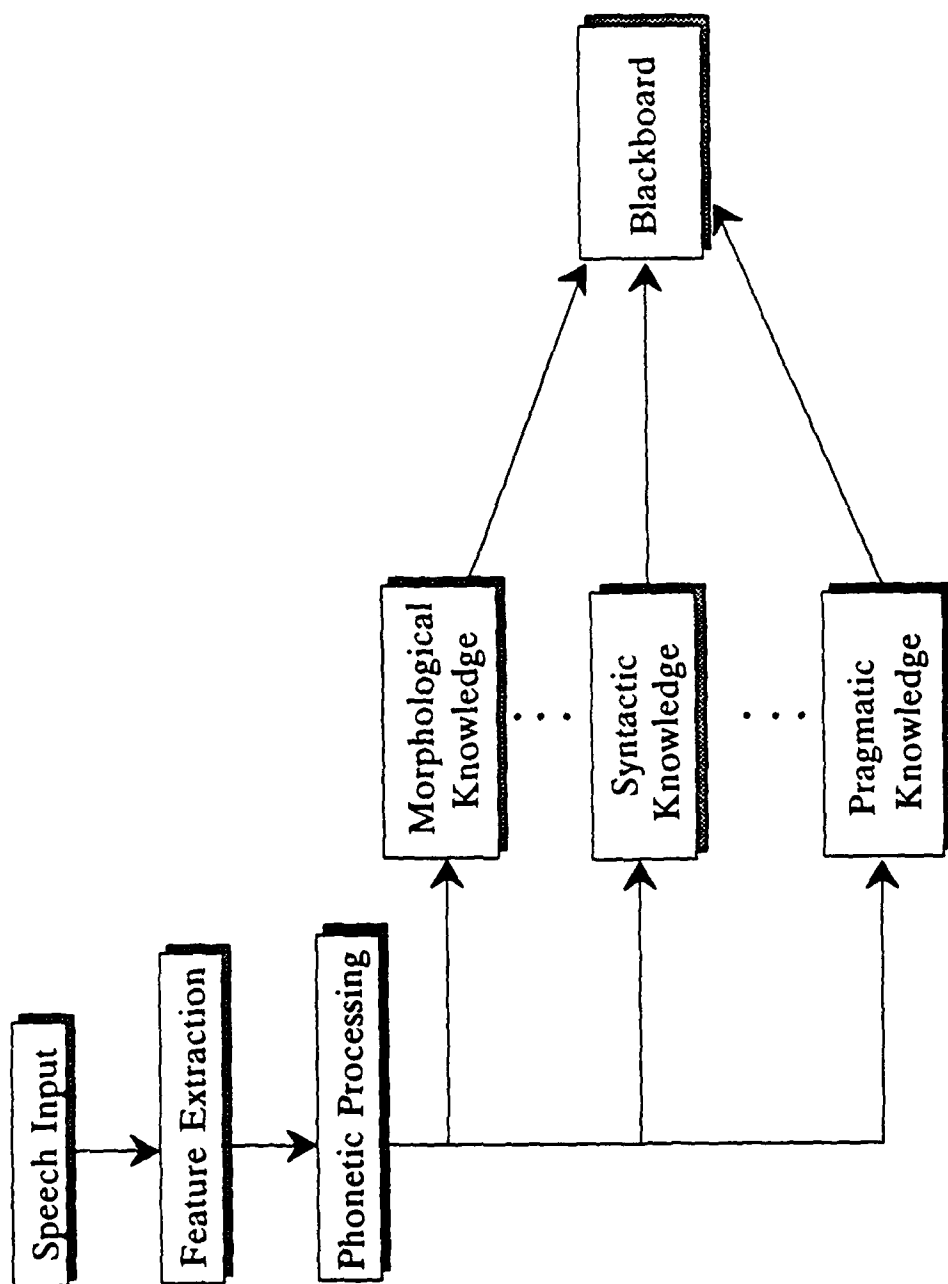


Figure 15
Architecture of the Hearsay III Continuous Speech Understanding System

the typical human vocabulary, were never able to even approximate real time operation, and spent far too much time searching memory. The solution to this bottleneck, borrowing from cognitive psychology, is the use of knowledge about the problem or domain to constrain the search. Once this was realized, near-real-time speech systems became possible, as well as the recent successes of vision, natural language and expert systems.

The reader should note that much of the recent improvement in the performance of both IWR and CSU systems is due to developments in special purpose signal processors. These include special purpose IC's, as well as systolic structures for processing lower level speech. The remaining critical bottleneck, that of knowledge retrieval and processing, requires additional research into symbolic computing, in order to structure an optimal high-level speech processing environment. (Several such structures are suggested in Section IV, where the relationship between processing structure and function will be examined more closely.) Special purpose processors are also under consideration for vision systems, as we shall see in the next section.

C. VISION

Vision, as its name implies, involves providing a machine with the capability to understand visual input in real time. As in human vision systems, machine vision includes the identification of what is in the image or scene, and how the various elements are related to one another, both spatially and temporally. Such systems have broad application, both commercially and militarily, beyond their use as an input device to ease man-machine communications. Many of the recent advances in process automation and robotics (e.g., bin-picking robots) have become possible because of machine vision research. Other commercial sector applications include: sensors for automated welding, handling hazardous materials, VLSI manufacturing, computer aided design and manufacturing (CAD/CAM); inspection of manufactured goods; medical imaging; remote sensing for cartography, traffic monitoring, drilling aids, land use management, and

exploration for oil and minerals. Potential military applications are just as diverse, such as autonomous vehicle navigation, photointerpretation, reconnaissance, target acquisition and range finding, terminal homing, and tracking of moving objects. We also see it as the most obvious application for optics, since visual information is inherently optical in nature and can be processed two-dimensionally.

In many ways, the techniques used in machine vision research build on the knowledge-based systems presented in the previous section on speech understanding. As in speech, high-level knowledge about the scene or image under consideration is effectively use to constrain the object identification and understanding process. Similarly, in vision, this high-level knowledge specific to the scene can be effectively used in guiding lower level operations. Later on in this section we will see examples of how knowledge can be applied to constrain the processing.

Another parallel between speech and vision is the approach to use of knowledge and control. Early work in vision was carried out using the bottom-up hierarchical paradigm, similar to the case in isolated word recognition systems. With this approach, preliminary processing performs edge detection, feature extraction, and linking without the benefit of knowledge-based inferencing. As in the case of speech, this preliminary processing is referred to low-level vision, and the subsequent interaction with the knowledge base is termed high-level vision processing.

Currently, developers are exploring advantages of mixing the high- and low-level processing in paradigms of broader scope. For example, high-level reasoning about expected features and their relationships is useful in tasking and guiding the lower level processing. The utility of this stems from the efficiency of focusing on specific regions of an image for a specific purpose, such as resolving an edge and following a lineal feature. At other stages in the vision process, the low-level processing may be involved in generating hypotheses to be evaluated by the top level processes. In this case, the efficiency of the process is improved by avoiding hypotheses which are inconsistent with the image.

Finally, just as speech systems incorporate pattern recognition techniques in the low-level processing, vision research has also developed as an outgrowth of early pattern recognition work. However, treating vision as mere pattern recognition is inaccurate; pattern recognition is strictly numerical in its approach, operating directly on the image. Vision, in contrast, uses knowledge about the scene to categorize the image, and operates on the symbolic representation of the image rather than the image itself. This is a major and important distinction, and is analogous to the use of syntactic knowledge in understanding language (see natural language understanding in Section III.D).

Having introduced the concept of machine vision, we would like to describe, in an elementary fashion, the types of processes and computations associated with vision. Detailed discussions are beyond the scope of this text, but the reader is referred to excellent books on the subject by Marr⁵ and by Ballard and Brown¹³ for additional information.

For simplicity, in the following discussion of vision, the processing will be considered to originate with the pixels on the visual detector, which is usually some form of two dimensional imaging array, such as a TV camera or vidicon. As in speech, low-level processing focuses in on the transformation from signal-level to symbolic information. This information is in the form of intensity variations across a two dimensional array, and the relative positioning of those variations. Color and texture information can also be utilized, since both can be extracted from the input image. The processing procedure begins with extraction of features via the processing of pixels to find edges and regions, as shown in Figure 16.

The first step in this is the the preprocessing stage, which prepares an image for actual image processing. For example, image restoration may be required to remove the effects of noise and optical or motion blur. Geometric distortion correction can be required to remove the effects of angle of view, to correct for common lens distortions, or other types of distortions introduced in the process of presenting the image to the computer. More than one sensor may be used in order to obtain, for instance, a 3D image, or to take advantage of information in several

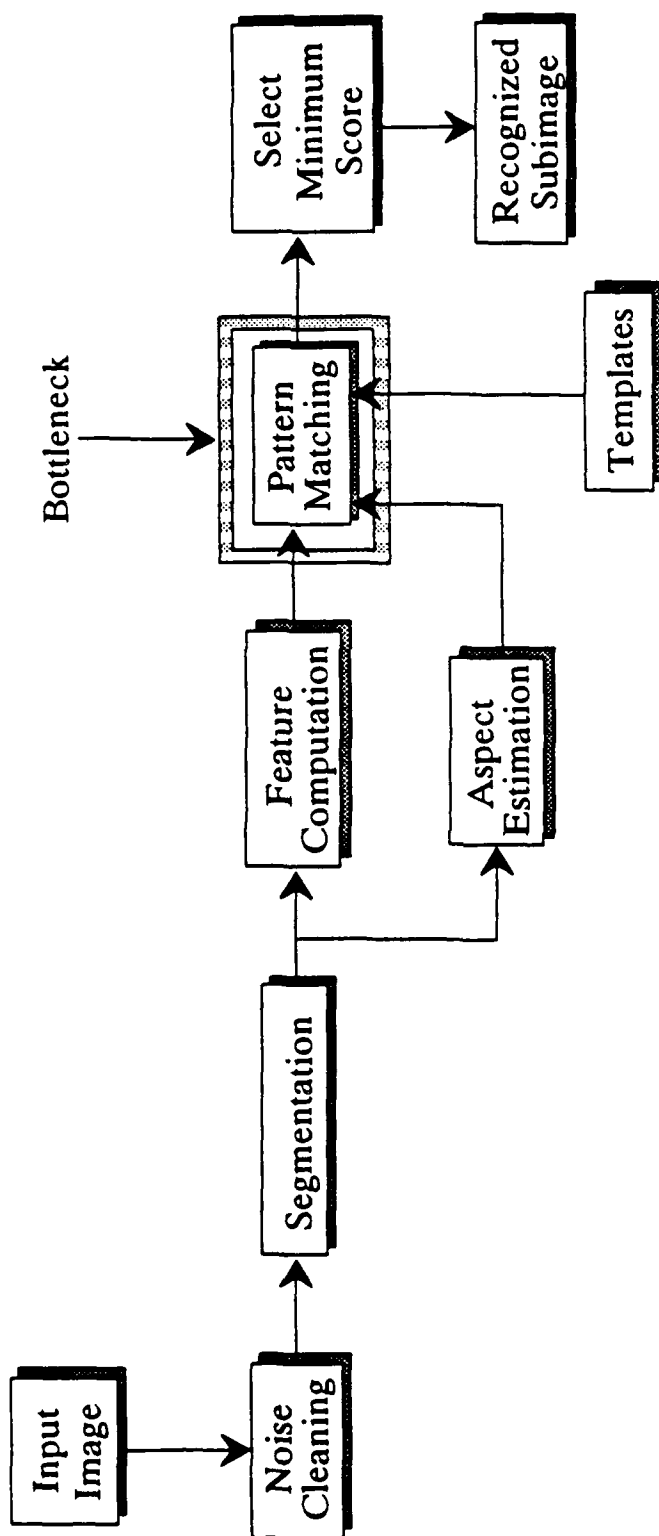


Figure 16
Low-Level Vision Processing – Pattern Matching Approach

wavelength ranges. In such a case, the inputs of the several sensors can be combined in the preprocessing step in an attempt to achieve maximum fidelity.

Typically, the first operation after preprocessing is feature extraction. This is usually accomplished by identifying gradients within the image, using the "DOG" (difference of Gaussians) operator. This operator, shown in Figure 17, is created, as its name implies, by subtracting two Gaussian operators to create an operator with two zero crossings. Application of this operator to the image creates a distribution of intensity gradients, and in particular, identifies the edges of objects.

The next step in the processing is termed the primal sketch and first-order segmentation phase. By deciding such things as where the edges of the objects contained in the scene are, which lines and edges at one part of the scene are continuations of lines and edges from other parts of the scene, and what regions of the scene belong to individual objects, a stereotypical "sketch" can be generated from which recognizable features can be extracted. The idea here is to remove the computation from pixel level processing as quickly as possible.

As an example, a tank is represented as moving on a roadway in Figures 18a-d. The "DOG" operator extracts the edges of the tank, which can then be linked during the primal sketch phase. For the tank example, the image has now been segmented into differing regions, corresponding to the turret, the base, the road, and the wheels.

Some method must be devised to recognize actual objects in the primal sketch (for the tank, these are the wheels, cannon, road,...) and relationships among those objects. This is done during the iconic feature extraction and grouping step. Currently, there are several competing ways to do this, and simply stated, they involve intensive, complex numerical and symbolic computations. Symbolic representation of the segmented image occurs at the next level, and finally the resultant symbol set of edges and regions is semantically interpreted in combination with models generated at

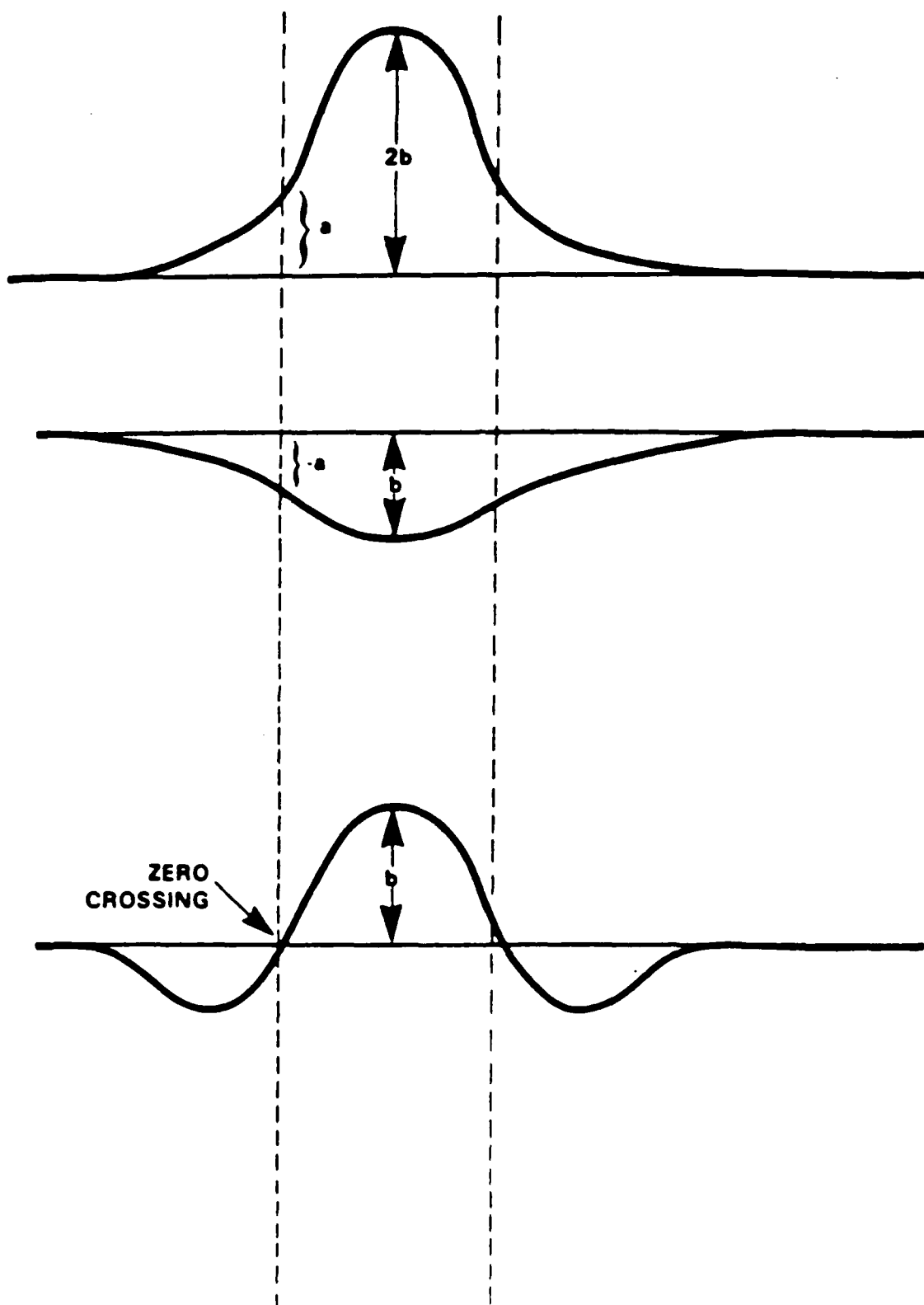
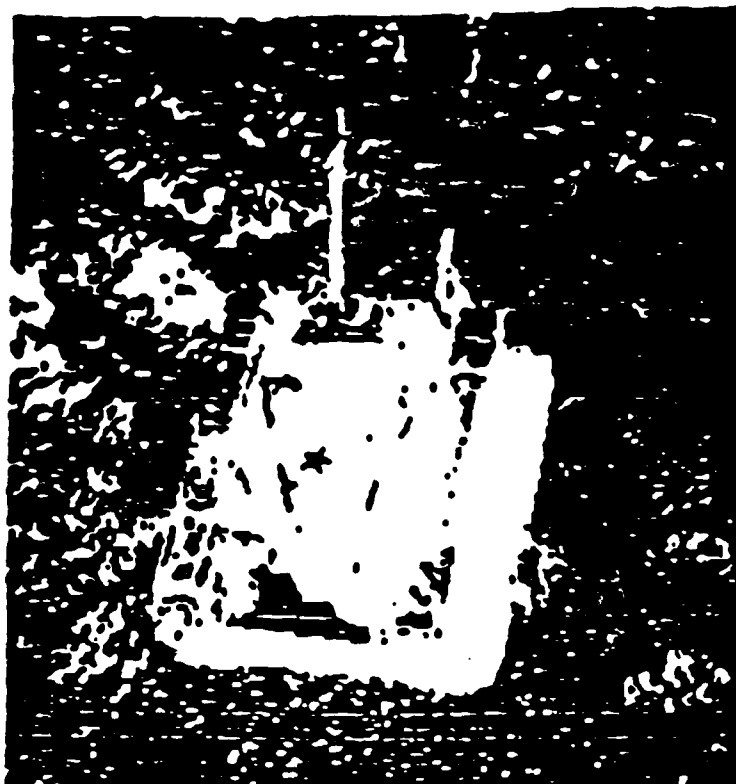
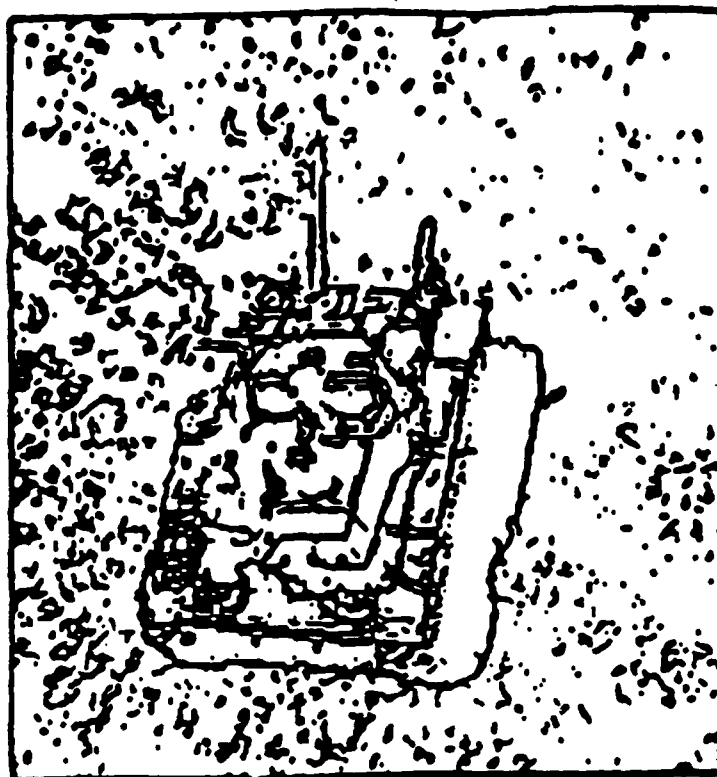


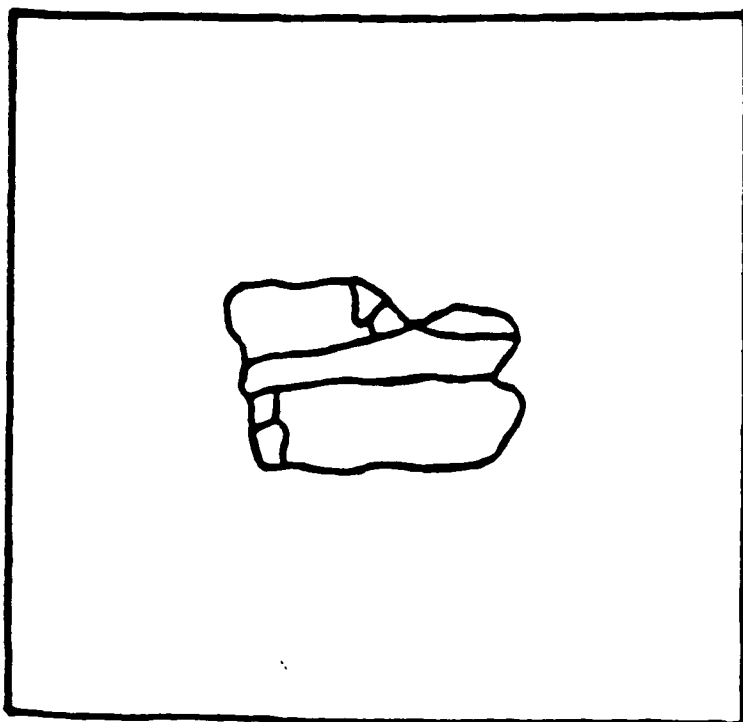
Figure 17
DOG Operator



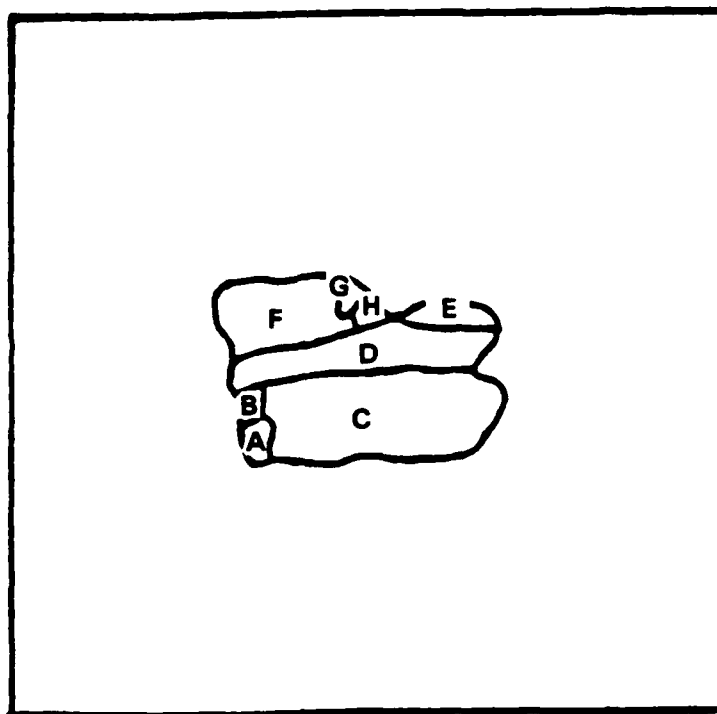
(a)



(b)

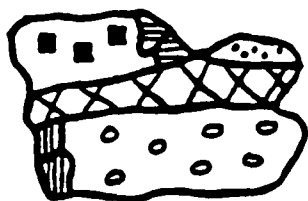


(c)

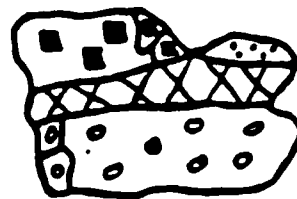


(d)

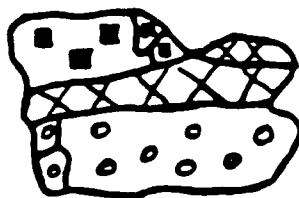
Figure 18-1
Int-level Vision Processing - An Example



(e)



(f)



(g)

Figure 18-2
Int-level Vision Processing - An Example

the highest level. This overall system architecture is presented graphically in Figure 19.

As an example of this process, we can look at the problem of identifying that the image is in fact a tank. One aspect of this is determining the identity and relation of the segments from the image in Figure 18. A technique for doing this, based on the use of the "frame" (see Section II.B), is shown in Figure 20. Here, the shape of the turret, and its position relative to the base, match descriptions of similar objects in the systems knowledge base. Within the frame, the turret is seen as deriving from the class gun, and having two specific incarnations: the tank turret and the gunboat turret. Within memory, the frame can call a model of the turret up for symbolic comparison to the segmented image. Then, by recursively repeating this process, the system can potentially identify the tank base. The system can then hypothesize that the segments of the image are elements of a tank. But this hypothesis is not validated until the other elements in the scene have been identified.

As one might expect, this process is a significant bottleneck in machine vision systems, even with the use of domain knowledge to constrain the search space. Multiple hypotheses could be formed, and comparisons made until one inference emerges without contradiction. This offers an opportunity for parallelism in this computation, but at present remains a critical process limiting real time image understanding. However, once object identification is made, the remainder of the computation is handled symbolically.

This leads to the scene understanding phase of the computation, which, unfortunately, is the least understood area of the enterprise. To say that we are looking at an object with a turret or with a top cannon or with tracks is far from understanding that it is a tank as opposed to a tracked troop carrier or some other vehicle. To say that it is a tank is far from understanding what type of tank it is or what it is doing. To do even the simplest of these things will require application of sophisticated modeling methods coupled with concise representations of these objects in the knowledge base of the system. A commonly employed technique uses multiple

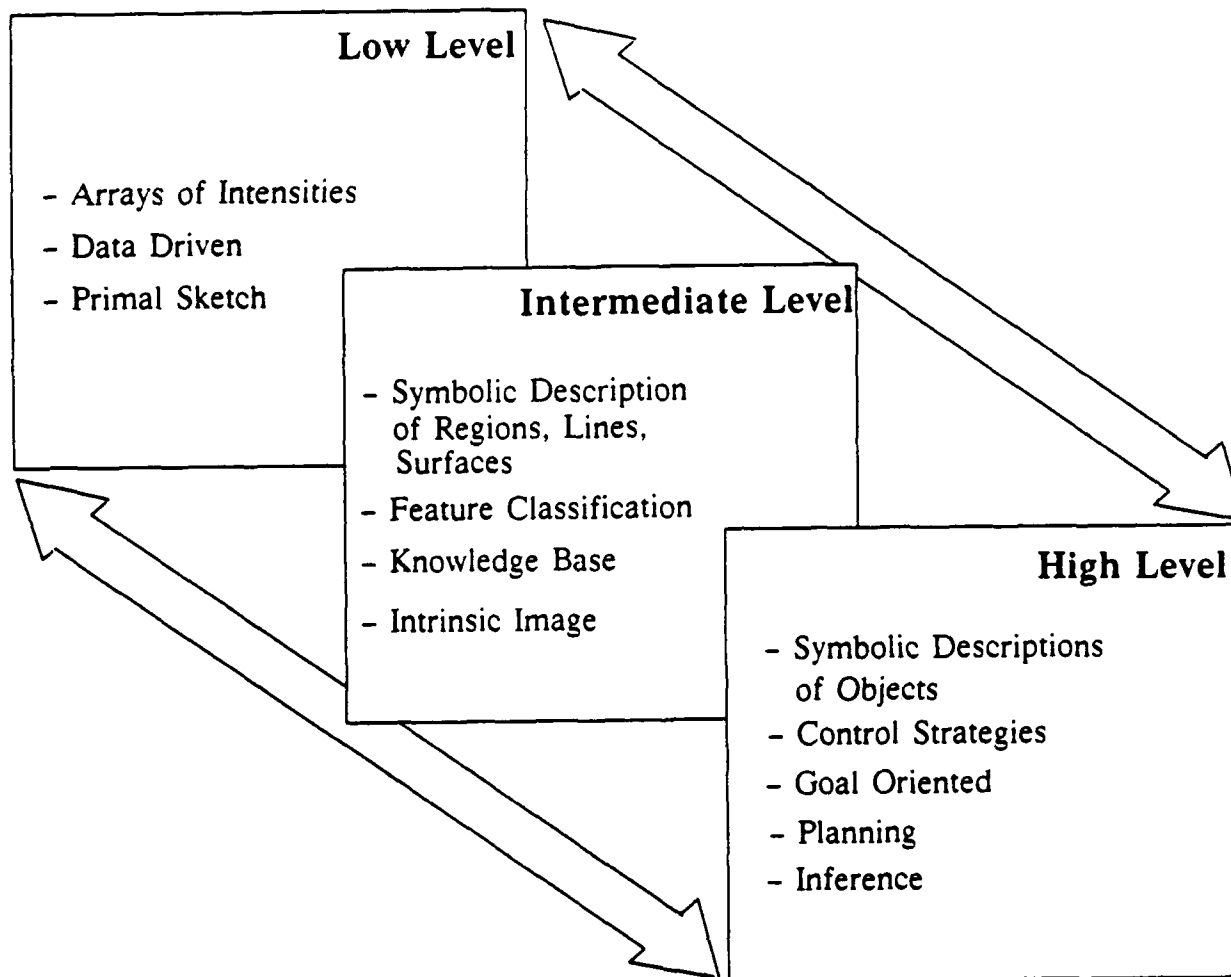
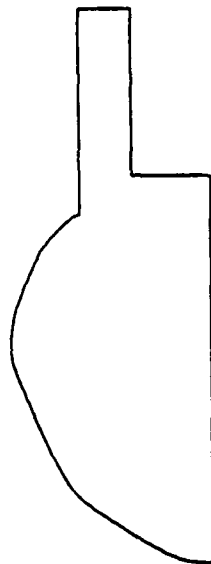


Figure 19
Image Processing Paradigm

Frame:	Turret
From:	Gun
Class1:	Tank
Class2:	Gunboat
Shape:	Oval
Slot1:	Gun
Next.to1:	Tank.Base
Next.to2:	Gunnery.Base
Display:	Model.Turret



Tank Display

Figure 20
Prototypical Frame for a Tank

images to analyze movement or change, and then inferences are made concerning these actions. However, this use of the time domain is not always possible, and, therefore, techniques for scene understanding from context are under development.

In vision processing, knowledge about the scene can be used at all levels to constrain the processing. Initially, geometric inferencing could use knowledge of occluded sides of a three-dimensional geometric object, such as a tank turret, to project the appropriate 2-D model. Domain-specific knowledge will distinguish a road from an airport runway, the difference being the environments around roads versus runways. Finally, identification of the road would increase the likelihood that the scene was that of a tank, rather than a gunboat.

If we had implemented a different control strategy, rather than a bottom-up one, knowledge about the scene would have constrained the system in other ways. As in the case of speech, many types of image understanding control strategies are possible, including the top-down hierarchical approach, mixed top-down and bottom-up, heterarchical approach, and variants of the blackboard approach. In the top-down approach, predictions made from high-level models in the knowledge base are tested and verified, such as the assumption that there is a road in the image. In this case, templates are commonly used to validate higher level hypotheses. In the blackboard approach, the feature extraction, symbolic, and semantic processors operate in parallel and communicate with each other via a common working data storage or "blackboard." The important point here is that knowledge influences all levels of the computation, whatever the control strategy, and very often requires iterative processing at lower levels to test any hypothesis.

From the above discussion, the reader should appreciate that the heart of the vision problem is the recognition of complex objects. Eliminating current computational bottlenecks involves developing new, more efficient algorithms and interfacing techniques to allow flexibility in transitioning between the basic component detection phase, the extraction/feature grouping process, and the high-level interaction with some knowledge base.

The problem is that currently, most vision algorithms run very slowly, and the individual processing stages are not integrated, very often requiring different hardware to achieve the desired functionality. Coupled with this is the need for efficient representations of the knowledge in memory, in a form conducive not only to some variant of template matching, but also to symbolic labelling and context identification. Underlying all of these problems is the drive to develop an efficient architecture for vision/image understanding processing. The belief is that by exploiting parallelism in vision computations, and by developing appropriate implementations of vision algorithms, many of these problems can be overcome. Optical solutions to this problem are also promising, as we will discuss in Section IV.

In fact, the authors believe that vision systems using high level modeling techniques could benefit from optical techniques, since they combine elements of numeric (e.g., edge detection/enhancement for segmentation, Fourier descriptors for feature computation) and symbolic (template matching, object recognition,...) computation.¹⁴ Since such systems are knowledge base intensive, higher memory bandwidth systems, such as those to be discussed in Section IV, may alleviate some of the problems associated with iterative identification processes. Most advanced image understanding systems^{15,16} require a smooth transition between numeric manipulations at the pixel and first segmentation levels and the symbolic computations at the higher, object classification and recognition levels.¹⁷ Coupling numerical and symbolic computations in ultimately solving vision problems, such as optical flow, may provide optical computing its most significant role in AI.

In this section we presented an introduction to machine vision, and identified, by way of example, some of the more stringent computational bottlenecks associated with the vision process. At the same time, we drew parallels between the processing associated with images and that conducted with languages, using the speech understanding discussion of the previous section as a reference point. Many of these same issues, such as

hierarchical control and hypothesis testing, will be revisited in the next section, on natural language understanding.

D. NATURAL LANGUAGE UNDERSTANDING

A commonly asked question is:

"What is natural language processing, and how does it differ from speech understanding?"

To answer this question, we will first define natural language understanding, and then explore its relationship with the types of knowledge used in speech understanding. Using an example from a well known piece of optics literature, we will then discuss some of the techniques utilized in natural language processing. This will lead directly into our discussion of expert systems technology in Section III.E.

The most popular definition of natural language, and one that is as accurate as any other, is that it is a language used in spoken or written form as the primary means of communication by a community of people. Hence, in some countries, and within most portions of the US, natural language most often refers to English; in other countries, it is Spanish, Chinese, or one of several other international languages. It should be clear, therefore, that linguistic attributes play a large role in the knowledge processing associated with natural language, and that it is closely allied with the field of computational linguistics.

As was the case with speech and vision, natural language understanding (NLU) had its genesis in man-machine interface research. Here, the desire was to have the machine understand English phrases or sentences input to it through some peripheral device, which was typically a keyboard. At that time, the principal application, in addition to language translation, was the querying of large data bases. The structuring and interpretation of the query was very important, and the user had to be careful that the input was accurately translated into the language of the data base. Drawing upon an

earlier example, a query to find articles in a data base on nonlinear optical materials for 2D Spatial Light Modulators could be structured using natural language input as:

"Find all articles on nonlinear optical materials for 2-D Spatial Light Modulators."

Early on, the system would have been able to interpret this input only because all words within the vocabulary of the system. Using an appropriate indexing scheme, this input would be transformed to:

Class: 2-D(w)Spatial(w)Light(w)Modulators
Subindex: nonlinear(w)optical(w)materials

where the (w) refers to the linking of the words. Even today, many data base query systems still function in this manner. To illustrate the point further, say it was desired to obtain articles on performance parameters of 2D Spatial Light Modulators. The query:

"Now find all articles on their performance parameters"

would only lead to an error message from the system, even if this statement directly followed the first. This is because the second phrase contained an indirect reference to an element of the first sentence, so that the subject of the second phrase could not be unambiguously identified. This application of natural language, termed interactive discourse, looks at pragmatic knowledge (see Section III.B) to understand references in conversations, alleviating the problems cited in the example.

In discussing natural language understanding, therefore, we are treating a discipline which is analogous to the human abilities to both read and comprehend texts as well as carry on dialogues. There are no direct parallels between natural language and the signal/image processing common to lower level speech and vision systems. There is a signal to

symbol transformation, but it occurs directly at the input stage, and from that point on, all processing is symbolic in nature. However, one saving point is that much of the same knowledge used in natural language understanding is directly applicable to high-level speech understanding.

In order to get a better appreciation of natural language processing, it is worthwhile to identify some other applications of NLU. While we will address some of the main applications, the interested reader is referred to the book by Rich¹⁸ for others. Aside from input interfaces, a principal application of NLU is in the area of computer programming. Here, the objective is to replace expressions such as:

```
DO 100      I = 1,50
      J = J + DATA (I)
100 Continue
AVE = J/50
```

with:

"Calculate the average of the 50 pieces of data."

Languages are being developed that are more "English-like," especially in the field of AI, as we will see in the discussion on expert systems in the following section.

Another aspect of NLU is text processing, by which we mean the processing of multiple sentences or paragraphs to extract critical information. This specialized extraction of information is very useful in literature analysis, such as the assimilation of information on optical computing from all of the journals of the IEEE and the AIP. Mechanical translation of texts, say from English into Spanish, is another application of text processing. Finally, the generation of natural language output, as in the explanation facilities of expert systems, is another critical application of NLU. The reduction of reasoning processes to a series of simple sentences, however, is a staggering challenge, even for humans.

There are several elements to any natural language processing system. The input is typically derived from a keyboard, although pointing devices or microphones are also sometimes used. In this case, however, the microphone is not responding to words or sentences, but is used instead to input letters in conjunction with a menu-driven software routine.

Following input, the NLU system decomposes or parses the sentence to identify word relationships and dependencies. The parser relies on knowledge about the grammar of the language to identify the subject of the phrase and relate nouns and verbs to their modifiers. For example, the well known phrase from the journal Applied Optics, "Optics is light work," can have a couple of different parsings, examples of which are shown in Figure 21.

Parsing decomposes the input phrase, identifying the relationships between the words and storing them symbolically. Following the parsing, a semantic interpreter takes this information, and, using information from the knowledge base, attaches meaning to each of the words in the phrase. This is accomplished either by lookup, or by conversion to an intermediate format known as the "meaning representation language." This language preserves the meaning and symbolic relationships of the words, but is designed to have a more direct mapping onto the vocabulary of the system than a random input may have. From the above example, the system may determine that either of the following are true:

Optics	=	Not_Work (N)	
Optics	=	Easy (Adj)	+ Work (N)
Optics	=	Work (N)	on Light (N).

The relative merits of each of these interpretations is determined by higher level processing. In this case, the representation is then passed to the domain and discourse processors, which use pragmatic knowledge to generate hypotheses about the meaning of the input. These hypotheses are either verified by comparison with the knowledge base, or lead to additional semantic, domain, and contextual processing.

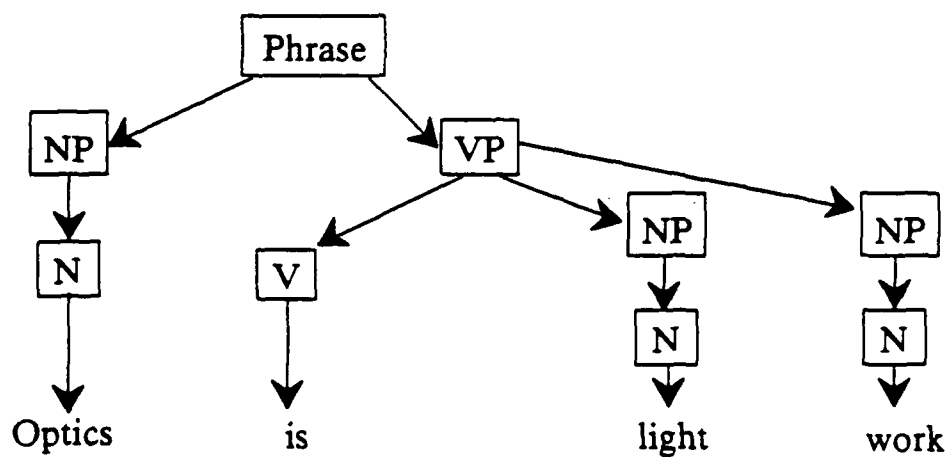
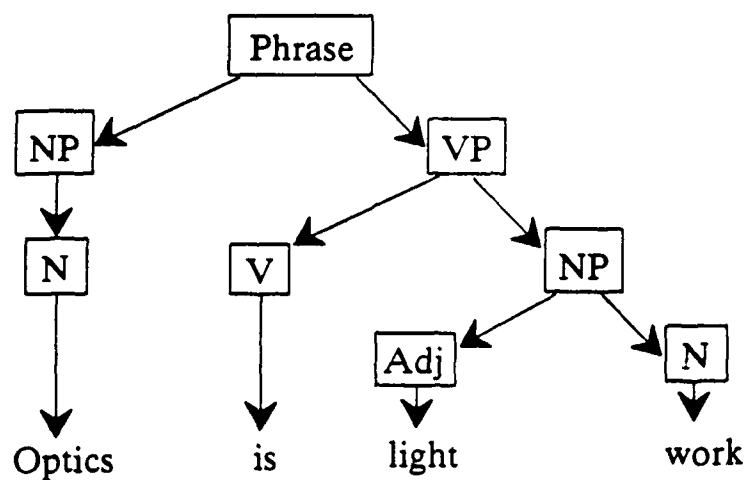


Figure 21
Possible Parsings of the Phrase "Optics is light work"

In the above example, a bottom-up paradigm was used to explain the knowledge processing, just as we applied it in the cases of speech understanding and vision. The bottom-up approach, shown graphically in Figure 22, can be used to move the processing up from the parser to the semantic interpreter, domain and discourse processor, response planner, and data and knowledge base translators. However, other approaches are equally plausible. For example, a system architecture based on the blackboard model (Figure 23) may be useful to combine partial understandings by the processors and achieve full understanding more rapidly and in parallel. For example, a parser may find two different parsings of the sentence "Optics is light work," but other processors will have to determine the most likely interpretation by using other knowledge.

Unfortunately, natural language processing is at present unable to unambiguously determine the meaning of input sentences, or use contextual or pragmatic knowledge effectively. Each phase of the processing is a computational bottleneck in its own right, particularly when dealing with input information which is flawed (e.g., wrong punctuation), or has errors (e.g., misspellings). In the understanding of raw text, natural language systems need to be more broadly applicable, more robust. They are currently slow and limited to understanding text only in very narrow areas, with a low accuracy of interpretation. As was the case with speech and vision, there has been a tradeoff between generality and performance in natural language processing systems.

An overall goal of natural language research is the development of a sufficiently robust system (necessarily domain portable) which can achieve high levels of accuracy in interpretation. As before, the desire to optimize performance has initiated the development of parallel algorithm research, with an eye towards achieving optical or multiprocessor implementations. But, as in the case of vision, our understanding of parallel tasking is not sufficiently developed to understand the implications of this research.

From the preceding discussion, we can begin to see how a multi-processor or optical processor could be used to achieve understanding of

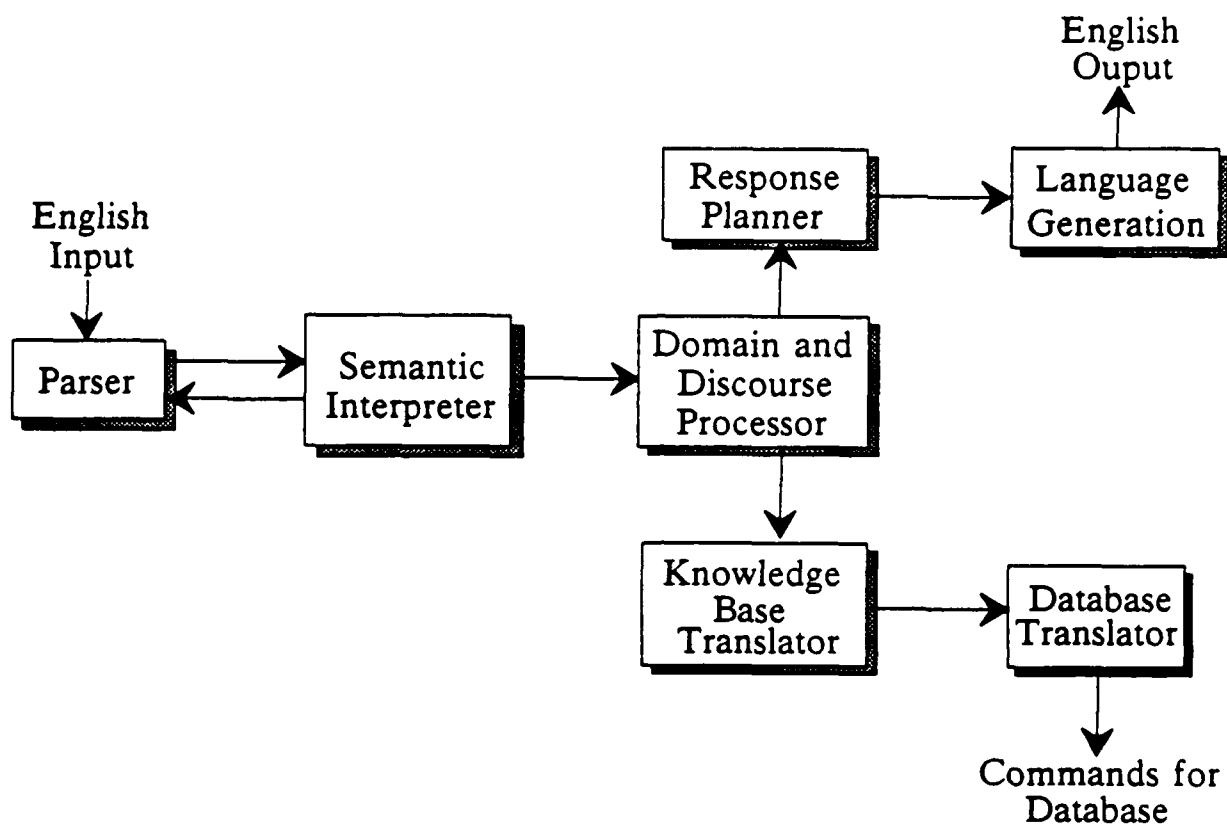


Figure 22
Elements of a Natural Language System

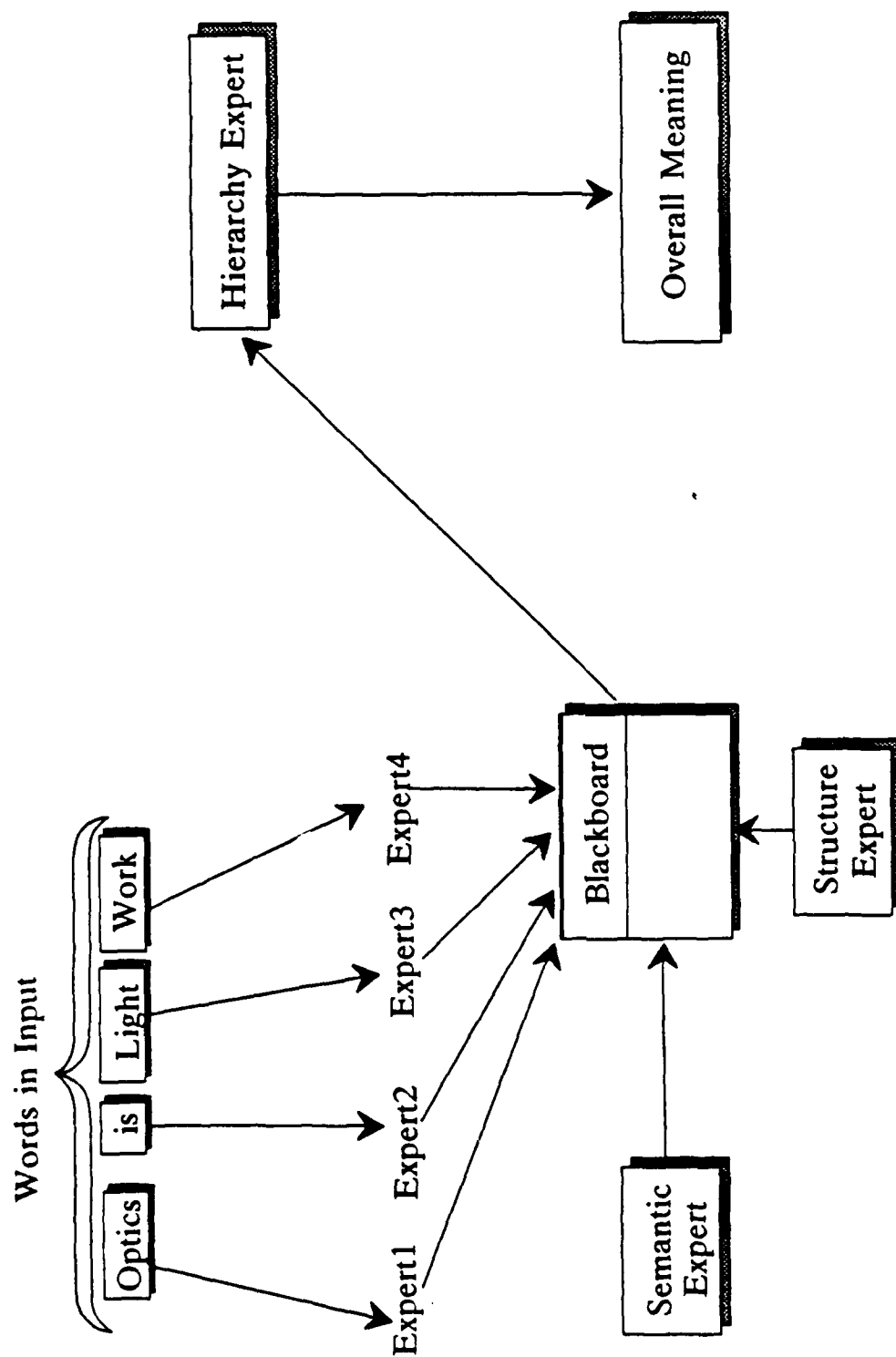


Figure 23
Natural Language, a Blackboard Model

natural language in real time. A parser itself could be parallelized, since many different morphemes, lexical variations, and syntactical structures can be analyzed simultaneously. A blackboard approach to the main processors would be immediately parallelizable.

The final point to be made is that, in describing natural language processing, we are in fact discussing a behavior of computer systems. We have crossed into the area where we are looking at systems that can interpret and reason about inputs, hopefully even learn from their mistakes. The issue of interpretation expands into the problem of where does natural language processing end and where does expert system processing begin? Knowledge representation becomes more and more common to both, since stored knowledge will have to be used in order to understand new natural language input. General knowledge about the world, i.e., common sense, as well as specialized knowledge about a problem domain must be used to determine if the new input is literally plausible and corresponds to physical reality, or if it is erroneous as opposed to metaphorical, humorous, or sarcastic. These questions will surface again in the next section, where we will look at the discipline known as expert systems.

E. EXPERT SYSTEMS

The final capability of symbolic computing that we will discuss is expert systems. These systems are characterized by their almost total reliance upon manipulations of symbolic information, in marked contrast to the systems presented previously. In speech, vision, and natural language understanding, the emphasis was on some form of signal-to-symbol transformation, coupled with the use of high-level knowledge. These are disciplines which are dependent upon knowledge retrieval and reasoning processes. Expert systems, on the other hand, involve the process of knowledge acquisition in addition to the retrieval and reasoning process. Advances in the other functional capabilities would be of great benefit to

expert systems technology, since it draws heavily on the interfaces provided by these disciplines.

But what is an expert system? In its broadest sense, an expert system is a machine which mimics or emulates the thought and reasoning processes of a human expert. It seeks to utilize the solution techniques utilized by the human expert to solve a particular problem. These techniques are, in many cases, just the rules of thumb or heuristics described in Section II. The way experts look at a particular problem, the information they look at, the data they require, their knowledge about the knowledge they possess (what we term metaknowledge), what they ignore - these are the elements we call expertise. Expert systems attempt to capture this expertise, and apply it to a particular problem area or domain. The price to be paid, as we have seen in earlier sections, is one of generality; to date, expert systems have only been able to function in very specialized, narrow areas of expertise. Some of these successes were cited earlier in this section: R1,⁸ the system that configures VAX and PDP series minicomputers; DENDRAL,⁷ the system developed to interpret spectroscopic data; and MYCIN,⁹ the system which aids physicians in making diagnoses in internal medicine.

In each of these systems, knowledge was placed into the machine which enabled it to "understand" the problem, in much the same way as was done for speech, vision, and natural language systems. It was then possible for the machine to function as a decision aid to the user. The system used knowledge retrieval and reasoning to be expert, generating inferences about the problem based on data supplied by the user. The power of these expert systems is the amount of symbolic information which they can store in their knowledge base, and their ability to rapidly process it.

There are three components to any expert system (see Figure 24) - the knowledge base, the inference engine, and the explanation facility. This expertise is stored in the knowledge base, using one or more of the representation types described in Section II.B. The knowledge base includes the heuristics as well as any elements of metaknowledge required for the task. The inference engine gives the expert system its reasoning capability, allowing the system to combine rules or frames to generate a conclusion.

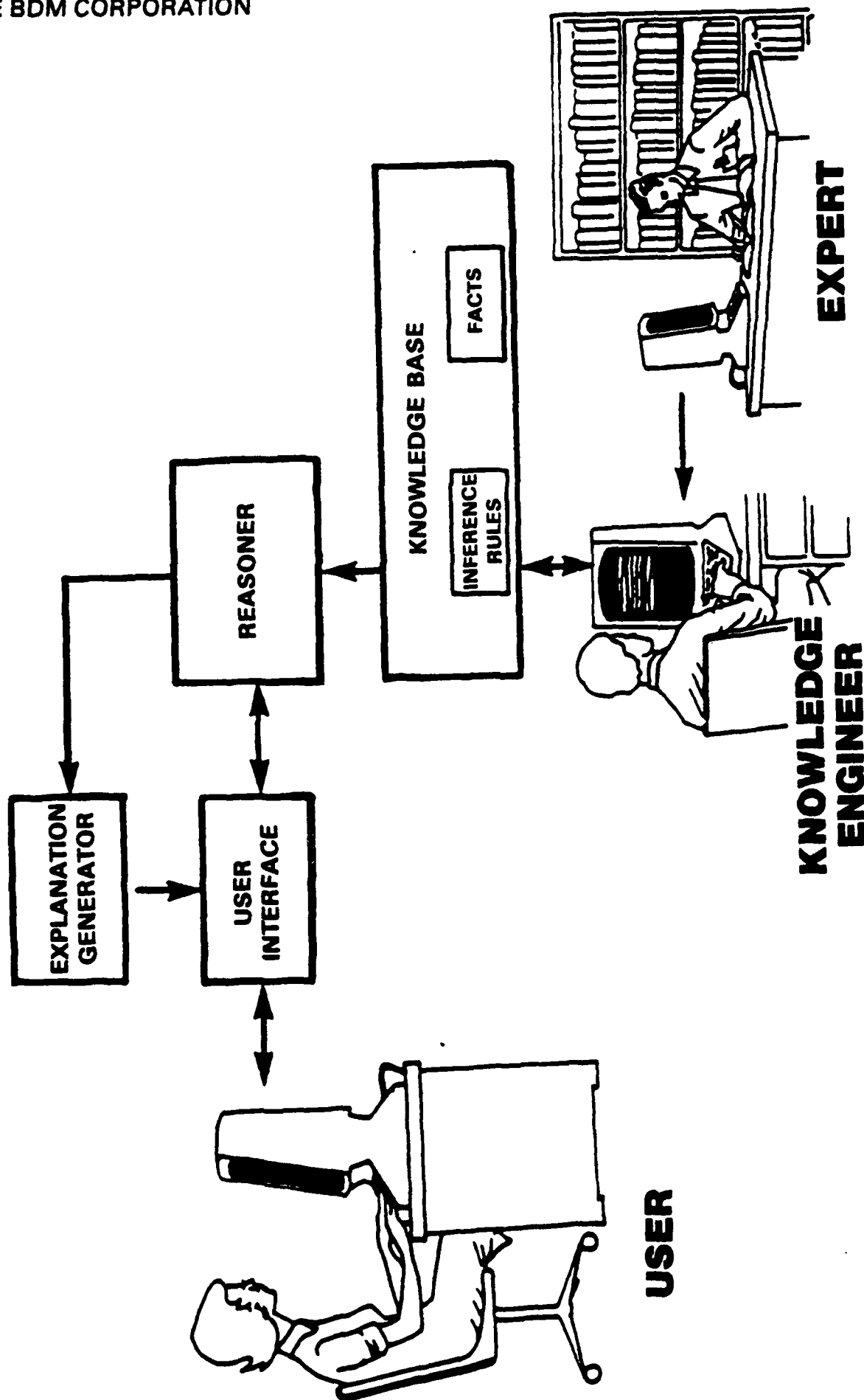


Figure 24
The Elements of an Expert System

This usually includes backward chaining, or goal-directed reasoning, as well as forward chaining and expectation-driven reasoning. The algorithms that underlie these reasoning processes are very often matching procedures, comparing the truth of one statement relative to that of another. As an example, the Rete algorithm, used in the popular OPS-based expert systems (such as the R1 system cited above), matches antecedents of production rules to the global memory in conducting a backward chaining operation.

The explanation facility is what allows the user to understand the machine's solution strategy, to determine why particular conclusions were reached. Very often, the explanation facility is a modified natural language interface, allowing the user to ask the machine:

"How did you arrive at this conclusion?"

and receive an answer which shows the evolution of the machine's reasoning process. For a rule-based system, an appropriate response to the above question could be a list of rules that were "fired," much like the trace of a FORTRAN program. As the size of the knowledge base increases, and the expected number of rule firings scales proportionately, this explanation technique becomes insufficient. What is required is a system that can summarize the reasoning process employed by the system. Such an interface could take advantage of research in natural language understanding, seeking to allow those systems to cooperate with the expert system, possibly in a blackboard architecture.

The development of a "classical" expert system typically requires a team of three people - the expert, the knowledge engineer, and a symbolic programmer. As we stated before, the expert supplies the knowledge and expertise to the system, which is stored in the systems knowledge base. The knowledge engineer has the task of acquiring the knowledge from the expert, typically via interviews and by presenting the expert with simulations of the problem. This is programmed knowledge acquisition, and is independent of any knowledge obtained from external sensors or learned by the system itself. The symbolic programmer takes the input from the

knowledge engineer and literally programs the knowledge into the system. Since this is a labor intensive problem, early expert systems, such as the ones cited earlier, required many man-years of effort to develop.

This division of labor has been eased greatly by the advent of advanced expert system building environments, and now, the knowledge engineer and the symbolic programmer are very often the same person. These building environments are "shell programs," containing all of the tools that a programmer needs to develop an expert system. In such a tool, an organized knowledge base is supplied; but it is devoid of any knowledge. The inferencing capability is also supplied, allowing conclusions to be generated once the knowledge base is "filled." Drawing an analogy with spreadsheet programs for microcomputers, these environments provide the equivalent of an empty spreadsheet. The programmer has the ability to input knowledge into the knowledge base, much the same way that an accountant can input figures into the spreadsheet, and tailor the program to meet his or her needs. Finally, the ability to combine heuristics to reach conclusions is analogous to the combination of spreadsheet cells to form a new entry. Just as the spreadsheet revolutionized the use of microcomputers, these building tools have decreased the development time associated with expert systems from several man-years to several man-months, depending on the level of difficulty of the problem.

The knowledge base of an expert system is its power, since it has been found that specialized knowledge is an essential adjunct to logic. The arduous process of incorporating that knowledge into the machine is a major limiting factor, even in narrow domains. Interactive knowledge acquisition tools, such as TEIRESIAS,¹⁹ have proven their usefulness in helping the domain expert express knowledge in forms compatible with the knowledge base. More sophisticated systems can be built to infer rules themselves from presented data, as was done with MetaDendral.²⁰ Even better, we can build systems which can learn to guide their own search strategies, i.e., learn heuristics, as was done with ACT²² and EURISKO.²¹ But the principals of learning and adaptation are still poorly understood. Consequently, most expert systems are not currently learning systems. Rather, they use the

application of prestored knowledge instead of learning from problem solving, from failures, and from correcting errors. The goal of developing systems that learn, in the form discussed in Section II, could greatly decrease the time and effort spent in the development of knowledge based systems.

An advanced feature of intelligent systems, and expert systems in particular, is their ability to pursue multiple lines of reasoning in generating a conclusion or decision. This has the advantage of being able to prune or reduce the potential solution space by applications of either goal directed (backward chained), model directed, or forward chained reasoning. This is a very powerful technique, which, although still in its infancy, will allow a program to be approached from multiple angles until a solution is generated. But any opportunity for conducting simultaneous, multiple viewpoint reasoning is heavily dependent upon developing the appropriate parallel computational structures. This is due to the increased amount of knowledge based interaction in such a system, which would only worsen an existing bottleneck in uniprocessor-based AI systems. Unfortunately, as we will see in the next section, our understanding of parallelism in computations is rather limited. But the potential benefit of parallelism in symbolic computation is one factor which makes optical computing particularly attractive.

Having clarified what an expert system is, we can now investigate what they can do. Hayes-Roth, Waterman, and Lenat, in their guide to expert systems,¹⁰ identified ten areas of application for these systems. These ten, shown in Figure 25, suggest that there are a large number of potential applications of expert systems, for everything from decision aids, to the construction of a laser (within given constraints), to isolating the source of a failure in an optical system.

We would like to focus on the last of the above instances in a simplistic example of application (3). We have taken some liberties in developing the structure of this example, and we realize that it deviates somewhat from standard experimental practice. Nevertheless, we feel it

- (1) Interpretation of sensor data;
- (2) Prediction of consequences in given situations;
- (3) Diagnosis of malfunctions from observables;
- (4) Design of objects within given constraints;
- (5) Planning actions;
- (6) Monitoring or comparing observations to plan vulnerabilities;
- (7) Debugging and prescribing remedies for malfunctions;
- (8) Repair or execution of a plan to administer a prescribed remedy;
- (9) Control of systems; and
- (10) Instruction, which includes diagnosing and remedying student behavior.

Figure 25
Applications of Expert Systems

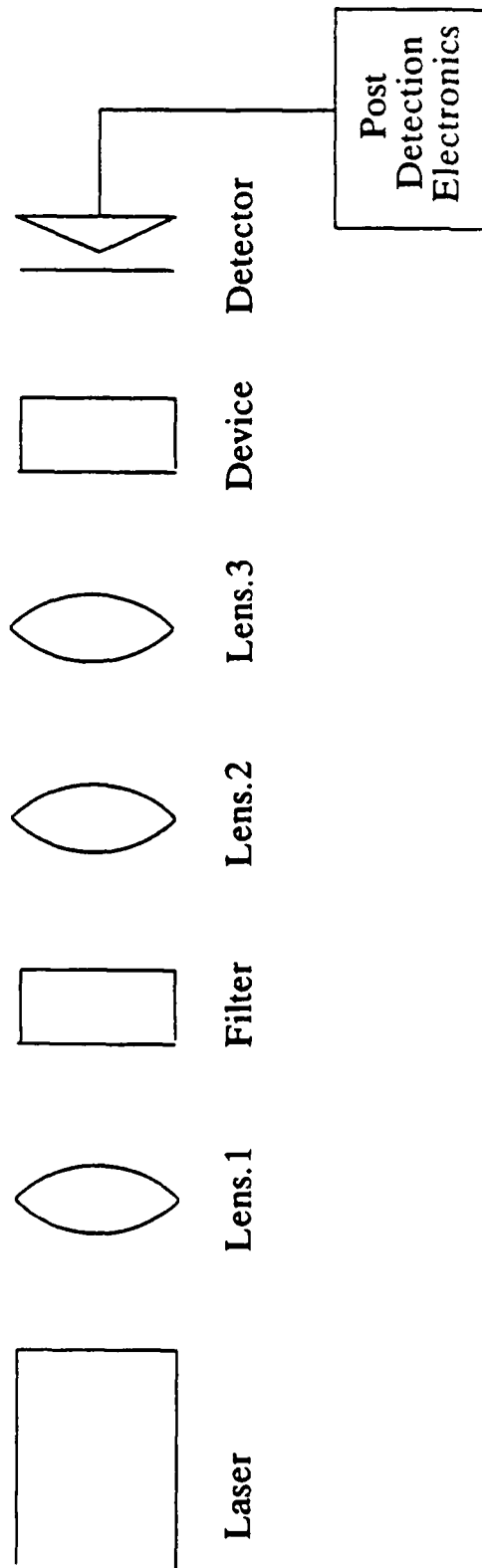


Figure 26
Prototype Optical System Used in Expert System Example

Frame: Laser	Frame: Lens.1	Frame: Filter	Frame: Lens.2
From: Opt.Sys	From: Laser	From: Lens.1	From: Filter
Input: NIL	Input: Laser.lt	Input: Laser.lt	Input: Laser.lt
Output: Light.1	Output: Light.1	Output: Light.2	Output: Light.2
To: Lens.1	To: Filter	To: Lens.2	To: Lens.3

Frame: Lens.3	Frame: Device	Frame: Detector
From: Lens.2	From: Lens.3	From: Device
Input: Laser.lt	Input: Laser.lt	Input: Laser.lt
Output: Light.2	Output: Light.3	Output: Current
To: Device	To: Detector	To: Opt.Sys

Figure 27
Frame-based Representation of Optical System in Figure 26

will be useful in conveying the principles of expert systems, as will also serve as a summary of the ideas discussed in previous sections.

For the purposes of this example, let us consider a system which consists of a laser, a narrow spectrum bandpass filter, a nonlinear optical device, a detector and a couple of lenses in the configuration.

In this system, the laser emits light into the first lens and subsequently into the filter, where the beam is changed or modified in some way. In this example, we will assume that the filter only passes light corresponding to the exact wavelength of the laser, so that any variations in the laser's output spectrum will lead to a decrease in light beyond the filter. This "filtered" beam is then transmitted through some nonlinear optical device, and the output of the device falls on the detector. Using a series of frames to represent this system, at the most simplistic level we have the following elements in the knowledge base of the system:

We have represented the optical path of the system by the slots **From:** and **To:**, reminding us of a semantic net relationship between the various frames. The changes to the light as it moves through the system are represented by the variations in the slot **Output:**. Finally, the attribute **Input:** stores the knowledge that laser light moves through the system. Let us now suppose that the detector's output drops to zero (**(= Current NIL)**, in LISP code), indicating a problem, and the system needs to assist the user in determining the source of the problem. In this context, our simple expert system can function as a diagnosis aid.

In our example, the system could proceed in one of two ways, either working backward from the detector or forward from the laser. In the former case, the system could hypothesize that there is no output because there is no input; in other words, there is no laser light reaching the detector. For this to be true, one of the following must also be true: the detector is faulty, the nonlinear device is faulty, or a component

THE BDM CORPORATION

earlier in the optical path is the problem. Here is where some expertise can enter the problem, since the system may know that:

Rule.1: If the detector output drops to null, then the detector is not at fault.

Rule.2: If there is a nonlinear optical device, then the outputs of the device are sensitive to alignment.

Applying this expertise to the problem, the system can eliminate the detector and other components as sources of the problem, and hypothesizes that the nonlinear device is at fault. By interacting with the user to obtain the additional required information, the system can resolve the truth of the initial hypotheses. This interaction with the user may take the form of:

System: "Is Device input still equal to Laser.lt?"

User: "Yes."

With this additional information, the system then concludes that the problem is in fact with the device.

In the other paradigm, the system could assume that the laser is faulty, and work its way out to the detector by way of:

If no laser.lt from the Laser, then no laser.lt into Lens.1; Lens.1:
Input = Null;

If Input = Null, then Output = Null;

If no laser.lt from Lens.1, then no laser.lt into Filter; etc...

finally concluding:

If no laser.lt from the Laser, then no laser.lt into Detector;

If Detector Input = Null, then Detector Output = Null.

Having generated the hypothesis that the laser is defective, it could then verify the hypothesis by asking the user:

System: Is the output of the Laser still equal to Laser.lt?
(Does Laser: Output = Laser.lt?)

If the answer is positive, some other component in the optical path is defective. The system can then hypothesize that another component is faulty, and iterate on the above process.

The above example is typical of expert system operation. The system reaches conclusions based on the truth of a series of knowledge base elements at any particular point in time. Hypotheses are generated and validated through additional interactions with the user, who supplies the necessary information about the state of the system. In this manner, expert systems can function as a diagnostic aid to a variety of users.

The example may also allow the reader to appreciate the difficulties associated with incorporating knowledge and expertise in AI systems. This is another instance of our recurring theme of the tradeoff between generality and performance. One of the limitations of expert systems is the narrowness of the domain of expertise incorporated into a system. Each system is a relatively isolated project, and, as a result, the techniques developed to solve the particular problem are not applicable to all expert systems. And increasing the size of the knowledge base, equivalent to expanding the domain of expertise, just leads to a combinatorial explosion of possible inferencing and machine states. There are also hardware limitations, such as the size of the working memory in the computer, which can only store so much knowledge at any point in time.

The reader may ask why increasing the size of working memory, thus enlarging the active knowledge base at any one time, will not at least partially alleviate the narrowness problem. It does, but we are still faced with the difficulty of organizing and managing the knowledge, and then channeling it through in serial fashion to the processor. This is complicated by the difficulties in representing certain types of knowledge, such as time-varying data, data with certainty which varies over time, and knowledge about processes and causality. The representations discussed in Section II.B are inadequate for many tasks because they are unable to appropriately store the time variations or statistical uncertainties associated with that knowledge. Going back to our example on isolating a failure in an optical system, use of the words usually and typically in the rules imply an uncertainty in the knowledge:

Rule.1: If the detector output drops to null, then the detector is usually not at fault.

Rule.2: If there is a nonlinear optical device, then the outputs of the device are typically sensitive to alignment.

At present, our abilities to represent these types of knowledge limit the breadth and robustness of most knowledge-based systems.

As an example, consider the problems associated with the representation of objects in space and the spatial and temporal relationships among them. Such problems, which are commonplace in vision research, also arise in representing relationships in expert systems. How will a computer "understand" that both the light which is incident on a lens and the light that emerges from the other side are really part of the same beam? Other types of knowledge may best be stored non-verbally, such as in visual images. Afterall, a picture or graphic may be worth many line of computer code. But the appropriate way to store graphical knowledge so that it can be updated, retrieved and used in making inferences is an unsolved problem.

Even if we could expand the domain of expertise, computational bottlenecks exist in the knowledge processing which limit the performance of most expert systems to between 10- and 1000-rule inferences per second (RIPS). For purposes of comparison, this corresponds roughly to throughputs of 1 to 100 MegaFLOPS for numerical computations. This is not purely a hardware bottleneck, since much of the computational load rests in how the software structures the knowledge retrieval and reasoning processes. As an example, one goal of expert systems research is to effectively modularize the way knowledge is stored and manipulated. At the present time, there is virtually no separation or modularity between the various components in an expert system - the explanation facilities and user interfaces share the same memory with the knowledge base and the inference engine. Organizing, tracking, and processing all of this knowledge in uniprocessors effectively limit the rate at which symbolic computations can be performed. This is an example of the famous "Von-Neumann bottleneck," which will be discussed in the next section.

These difficulties currently limit the size of the knowledge bases and hence the overall robustness of the system. What is desired is shown in Figure 28, where the components of the expert system have been modularized, but are still interacting at multiple levels within the system. Each component could possibly function on an independent processor, with each parallel process communicating by sending messages to other processing elements. This separation could allow for expert systems with larger domains, more robust inference capabilities, and more diverse applications.

F. CONCLUSION

Having generally introduced the main functional capabilities within symbolic computing, we have seen that each focuses in on improving the machine understanding of the domain in question. Understanding, in a limited sense, is achieved through extensive interactions with the knowledge base of the system. The main characteristics of intelligent systems, the heuristic retrieval of knowledge and the various reasoning

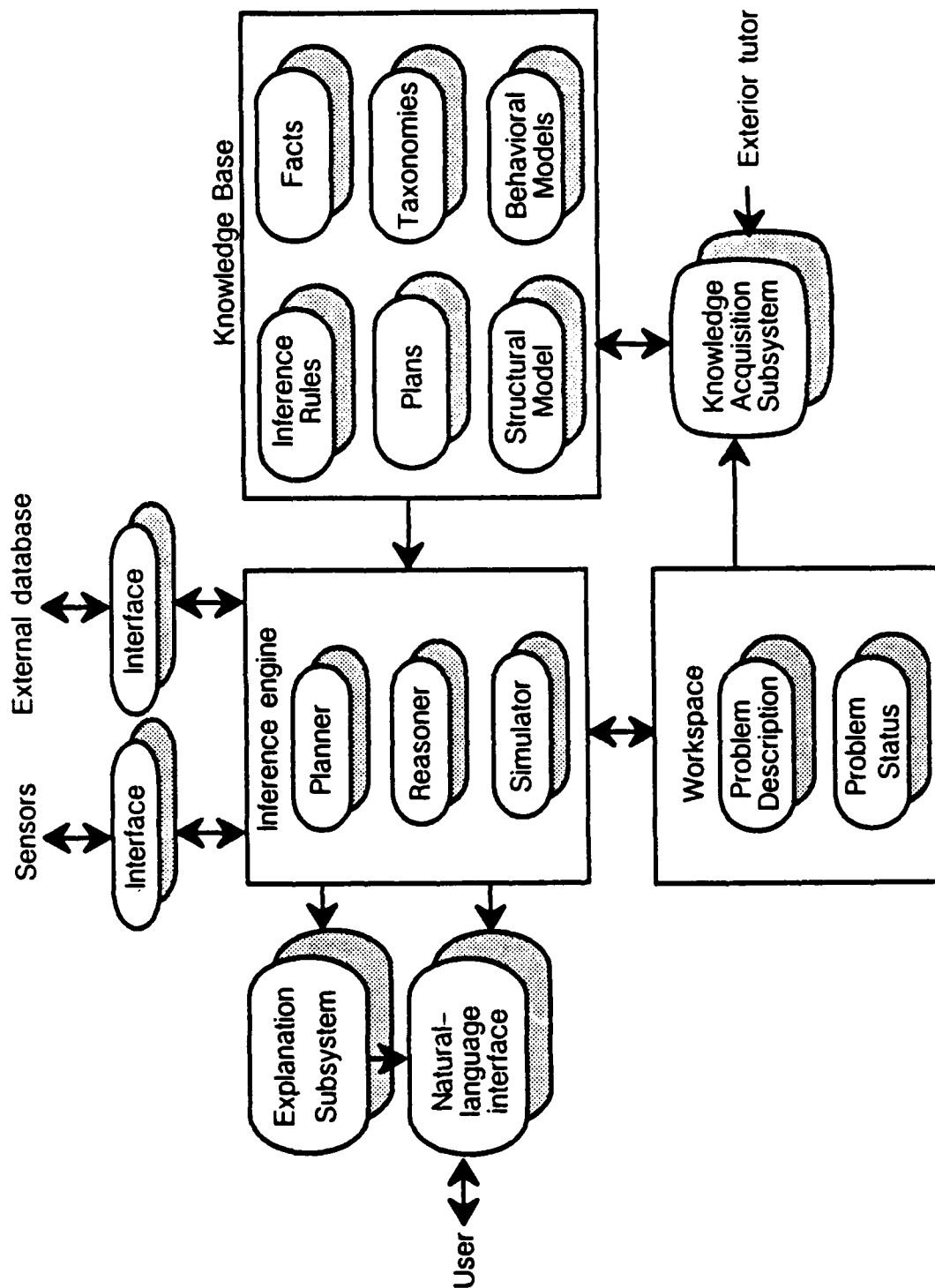


Figure 28
A Modular, Parallel Expert System

processes, were central to each of the capabilities discussed in the section. The acquisition of knowledge, and subsequent programming into the system, played a leading role only in the expert systems area. However, relative to current system performance, it is this organization, manipulation and processing of knowledge that present the main computational bottlenecks in AI systems.

Another point worth noting at this juncture is that the types of operations performed in AI systems are, for the most part, very domain specific. In many cases, the actual instructions, memory utilization techniques, evaluation and matching metrics, and search processes are embedded within the control sequence for the intelligent system. It is therefore very difficult to cite common operations, other than to show the overlap of techniques in high level processing. An example of this case be seen in the area of expert system development tools, where each tool possesses its own control structure, its own group of representations (production rules, semantic nets, frames, scripts, etc.), its own reasoning paradigm (forward chaining, backward chaining, expectation driven, or a combination of these), and, as a result, its own matching or evaluator structure. While any of these tools may have some operations in common with others, typically the variations are quite large from environment to environment.

An underlying theme of this section was the difficulty associated with representing knowledge in AI systems. To be useful, and reasonably general, multiple representation schemes will likely have to be used within any given system to express various kinds of knowledge, especially the difficult areas mentioned above. A problem which has quantities of both declarative and procedural knowledge apparently needs multiple representative schemes. Thus, systems will evolve with ever larger knowledge bases and with multiple types of representations, in order to address more sophisticated and more general problems. The challenge will lie in the the organization and management of this knowledge so that it does not lead to additional bottlenecks or decreases in the processing rate.

It has been estimated that just to overcome existing AI system bottlenecks, the computational throughput rates of symbolic computers will have to be increased by several orders of magnitude over current capabilities.²³ While highly parallel systems are being studied to address this problem, limitations in our current understanding will make it evident that parallelism alone cannot achieve the required speedup in computation rates. Use of alternative systems, such as optical computing systems, show great promise if the appropriate coding schemas and operations can be made optically compatible.

The previous sections have highlighted topics of current interest in symbolic computation, with an eye towards identifying potential applications of optical techniques. Applications such as vision and speech recognition have direct mappings into image and signal processing, tasks which optical processing and computing techniques are particularly well suited to. The major challenge for optics is the application of knowledge processing techniques on top of this lower level processing.

The main point in the earlier discussions was that the operations where optics excels are, for the most part, the same types of operations utilized in symbolic computations. So, while the promise of applying optical techniques to AI problems exists, some strong challenges remain, such as optically compatible representations, understanding parallelism (in optics and in computations in general), memory interactions, proper optical/electronic interfaces, and the ability to implement the desired architectures and required component technologies. These ideas will be explored in greater detail in the next section, which looks at potential architectures for symbolic computation.

SECTION IV
SYMBOLIC COMPUTING ARCHITECTURES

A. INTRODUCTION TO SYMBOLIC ARCHITECTURES

The previous Sections have laid the groundwork for the remaining discussion of symbolic computer architectures. Reflection on our earlier description of the fundamental characteristics of symbolic computing should raise an awareness to the importance of relationships between data elements - the relationships between objects and attributes in LISP, the relationships between nodes of semantic networks, the relationships within and between frames, etc. This has led computer scientists to investigate the performance improvements that could be forthcoming by the use of computer architectures for which the connectivity between processor nodes could reflect the relationships fundamental to symbolic computing. Such thinking derives partially from experience with numeric computers in enhancing performance by matching architecture to algorithmic structures and visa versa. Of course, one must keep in mind the importance of maintaining flexible architectures that can adapt to changing relationship patterns; otherwise, the result will be special purpose machines with limited utility.

The similarity of these highly connected architectures to neurological systems lends credence to their importance in symbolic processing. The brain, with its relatively slow components (neuron speeds on the order of milliseconds), is able to process symbolic information at rates several orders of magnitude faster than conventional (von Neumann) computer architectures. Two differences between the electronic and biological systems that stand out are the connectivity between components and the intermix between processor and memory operations. Neurons in the brain can have upwards of 10,000 synapses (biological connectors) whereas their electronic counterparts, gates, typically have only a few connections to other gates. In the area of memory distribution, the von Neumann architectures are characterized by a separation between the processor and

memory functions; and the transfer of information between the two often results in a speed bottleneck.

A criticism of von Neumann architectures is in no way intended. Afterall, when von Neumann proposed this processor/memory separation, it was based on the limitations imposed by the technology of that time period (late 1940s). Also, such architectures have proven vastly superior to the brain in performing numeric operations. Although it is beyond the scope of this Chapter to discuss optimal symbolic/numeric systems, future systems may someday consist of cooperating symbolic and conventional numeric processors.

Computers with high levels of connectivity and distributed memory have been labeled parallel processors due to their capability of supporting concurrent operations. Before discussing the potential for optical parallel processing, we would like to familiarize the reader with the principles and terminology of parallel processing in general, and to discuss broad categories of parallel architectures.

B. ARCHITECTURES FOR PARALLEL PROCESSING

There exist numerous taxonomies for classifying parallel architectures, but this discussion will touch on only those that are deemed useful in the context of this Chapter. For a more complete treatment, the reader is referred to the book by Hwang & Briggs.²⁴ As a start, one can deal with temporal parallel versus spatial parallel processors. The former most often takes the form of pipelining, which is the sequential execution of instructions or operations such that initial phases of follow-on instructions or operations are initiated before the latter phases of previous instructions or operations are completed. Spatial parallel systems are designed to execute multiple parts of a problem simultaneously. They consist of two or more processing elements, most often of approximately comparable capabilities, such that each element contains at least an arithmetic logic unit and a set of registers. Although the arrangement and connectivity of spatial parallel processors can change, the diagram in

Figure 29 illustrates the general concept behind the architectures. Note that interconnection networks, preferably programmable ones, play a major role in parallel processing.

A classification presented by Seitz,²⁵ shown in Figure 30, provides an interesting categorization of parallel systems based on the number of processors and the relative degree of processor complexity. Conventional uniprocessor architectures are plotted as a point of reference representing high complexity in a single processor. As one moves up to more than one processor, the trend is toward reduced complexity within each processor, a trend that is driven by total system cost on the one hand and by an escalating overall system complexity on the other hand. Microcomputer arrays are basically a set of computers that send messages to one another via a communication network as illustrated in Figure 31. Such systems are usually loosely coupled (versus tightly coupled); that is, the individual computers do not share main memory and I/O devices, although one computer can always draw upon another's resources through the communication network. The application of such systems in symbolic computing will likely be in solving problems that involve the use of more than one knowledge base. Each processor can work on a given part of the problem in such a way as to minimize the need for interprocessor communications.

The next category, computational arrays, represents systems whose processing elements have been designed for operations on the order of complexity of multiplication and addition. Systolic arrays, for which the processors are connected in regular patterns that match the flow of data in the computation, comprise most of the architectures in this category; however, more general purpose computational arrays will likely emerge as interconnection networks become more flexible. We will return to this point in our discussion of hybrid optical-electronic systems in Section IV.E.

The final categories of parallel machines, logic-enhanced memories and artificial neural systems, can be thought of as "smart memories." Such memories can greatly alleviate the "von Neumann bottleneck," posed by the need in a von Neumann architecture to constantly move information to and

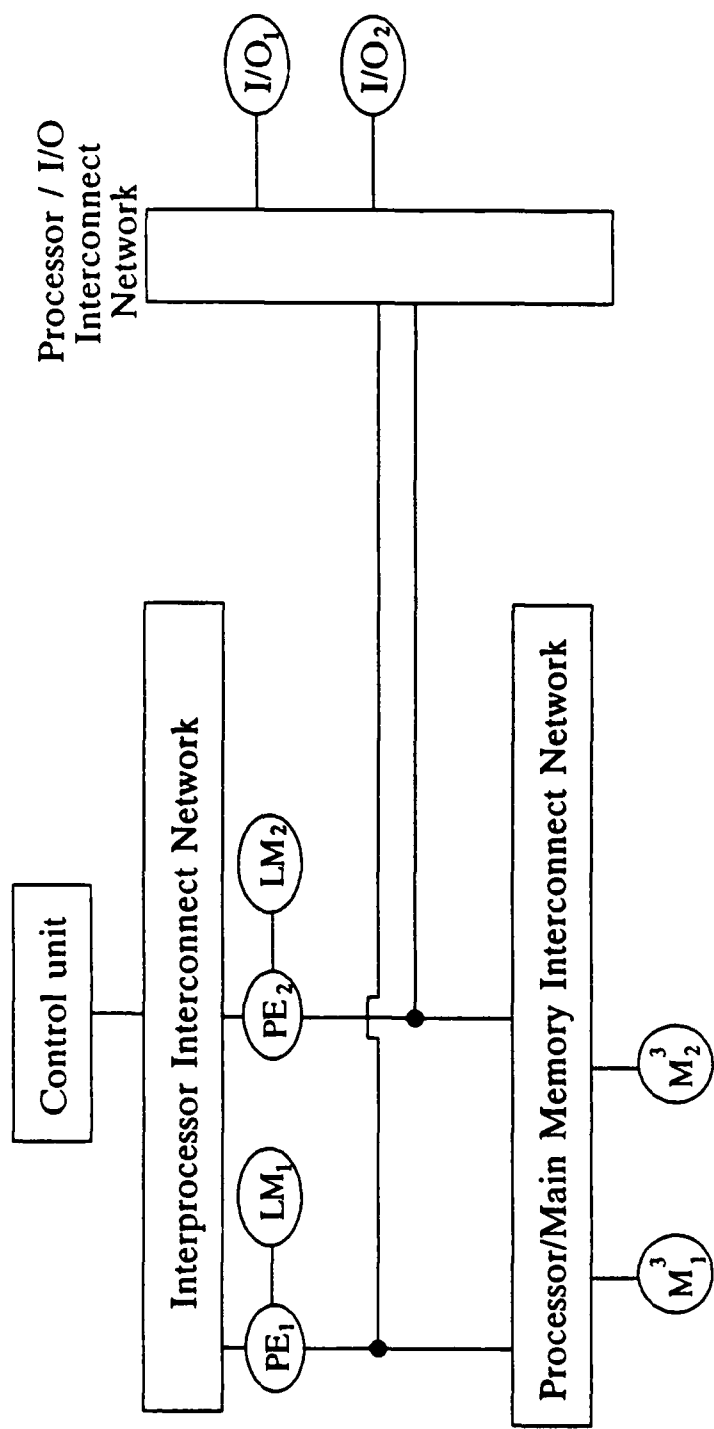


Figure 29
Block Diagram of a General Parallel Computer

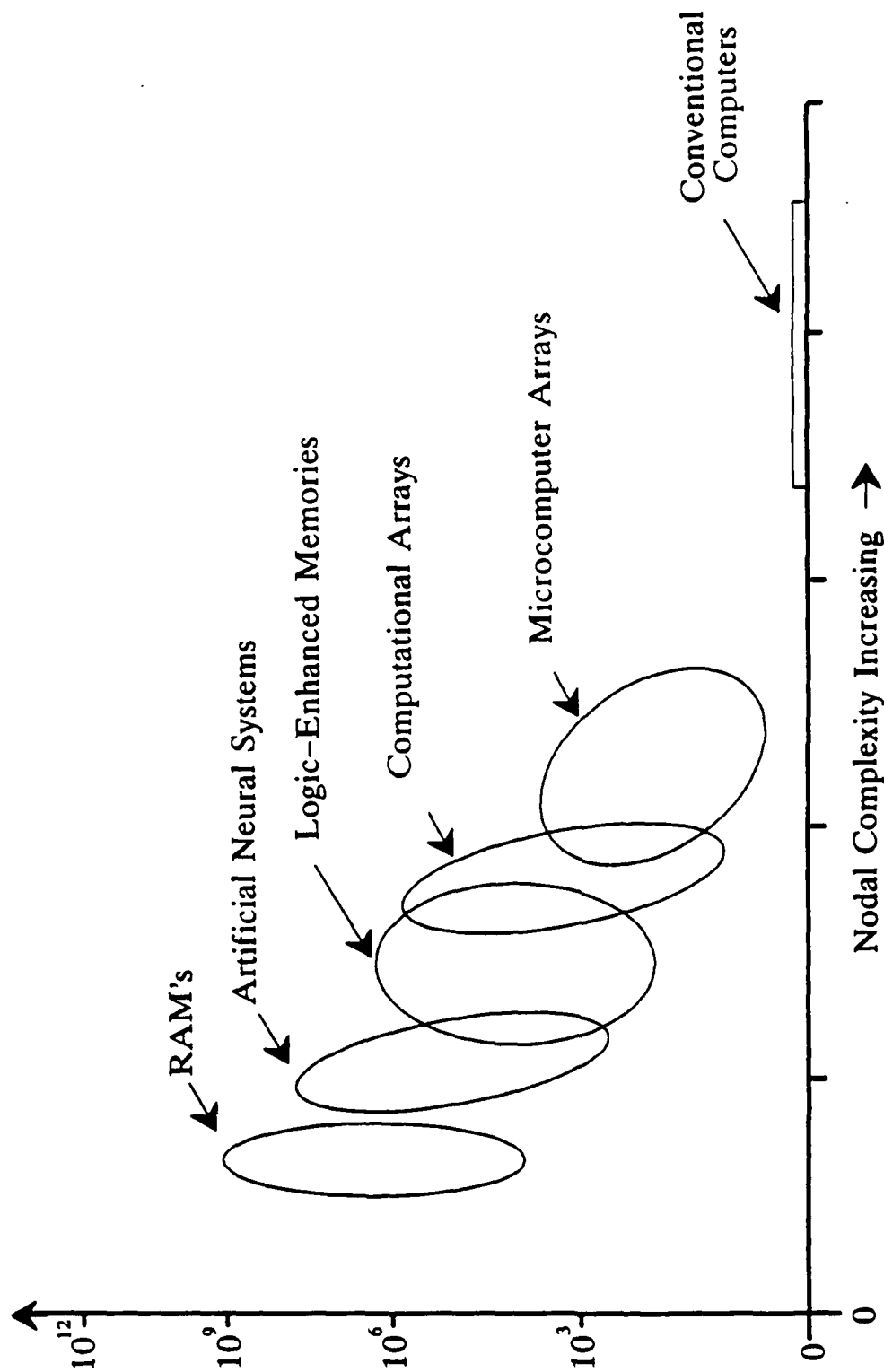
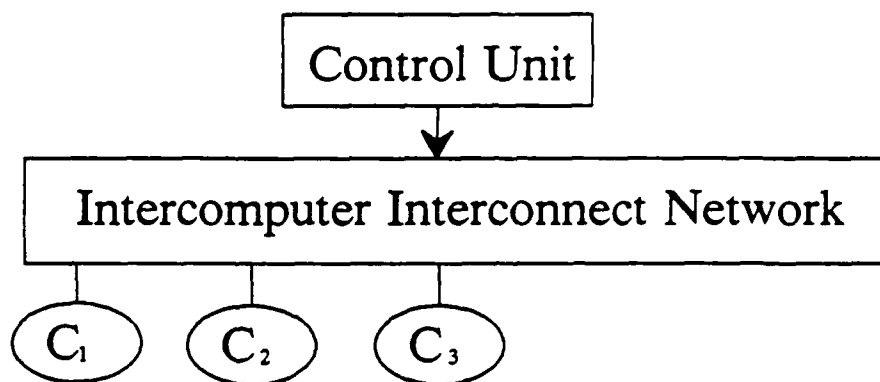


Figure 30
Classification of Parallel Processors by Nodal Complexity



C = Computer (complete with memory and I/O capability)

Figure 31
Microcomputer Array

from memory, by serving as the memory for a host computer; that is, some of the processing is transferred to the memory to avoid the speed delays incurred by transfers between separated memory and processor units. Each element (or node) of the logic-enhanced memories contains upwards of several thousand bits of storage, a set of registers, and some associated logic capable of operating on the storage contents and of directing communication with the other elements. For the case of artificial neural systems, the complexity of the nodes begins to approach that of a switching element. These systems are referred to as fine-grained parallel processors due to the relatively low complexity of the individual processing elements, and they are always tightly coupled. They must consist of at least several thousand elements to achieve a practical computing power. In fact, some fine-grained architectures with as many as one million elements are currently on the drawing boards.

The final entry shown on Figure 30, that of random access memories (RAMs), is given as a reference on the fine-grained end just as the uni-processors were shown as a reference for nodal complexity. RAM's, of course, do not function as multiprocessors due to the absence of connectivity.

It is the tightly-coupled fine-grained architectures that are attracting the most interest for symbolic computing because of the emphasis on connectivity over processing power. For example, each node of a semantic network could be mapped to a separate node of such an architecture, and the processing power can be directed toward establishing and identifying the types of the links. Some of the million processor machines mentioned above fall into the category of logic-enhanced memories, and are being considered for handling semantic networks consisting of a few hundred thousand links, and for supporting LISP with a few hundred thousand cons (connection) cells.

One other popular classification of parallel systems deals with Single-Instruction-Multiple-Data (SIMD) versus Multiple-Instruction-Multiple-Data (MIMD). There are two other categories - SISD and MISD - whose definitions should be obvious. SISD represents most of the serial

architectures in existence, and MISD represents a concept which has received very little attention due to its questionable practicality; therefore, only SIMD and MIMD are mentioned in the context of parallel processing. MIMD represents full parallelism and consequently is the most complex of the four categories, requiring individual control units for each of the processors and the ability to efficiently identify and allocate subsets of the problem at hand among the individual processors. The interconnection networks permitting interaction between the units differentiate MIMD systems from systems in which a problem is divided into operations performable on a multitude of SISD machines. Investigations into MIMD architectures have mostly been limited to loosely coupled systems, primarily due to limited knowledge of parallel processing.

The most significant gains in the near future in understanding parallel operations will likely come from implementing SIMD architectures, and therefore they have accounted for most of the current activity in parallel architectures. SIMD does not necessarily mean that all processors are executing the same instruction set but only that they are being presented the same instruction sequence, and each processor can be rendered operable or non-operable by the control unit. For the symbolic operations best suited to large fine-grained machines, SIMD may make more sense than MIMD due to the large overhead incurred by either storing entire instruction sets at each processor or routing instructions through the inter-processor network.

No matter what category given architectures fall into, they all share one overwhelming problem - the need for interconnection networks that can efficiently handle message transfers around the system. The performance of the message-transfer network is a prime factor in determining overall system performance. Figure 29 identified the three basic functional areas in which the networks are important (processor/memory, processor/processor, and processor/I/O), and although the performance parameters vary from one to the other depending on the limitations of the components being interconnected, the existing architectures are applicable to any of the areas, and the choices in the past have been made more on the grounds of

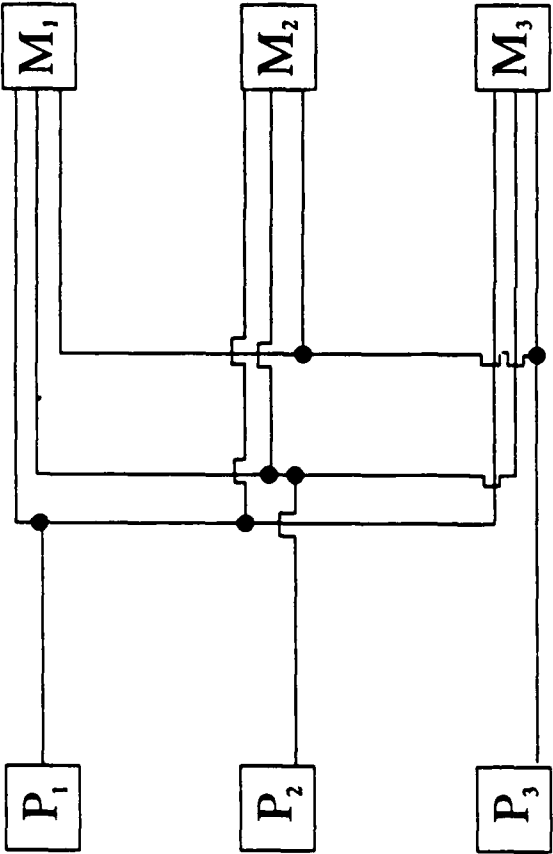
technology limitations and cost rather than functional differences. The following discussion of networks, therefore, will be independent of the functional area.

The thrust in interconnect architectures is toward programmable networks. This is not surprising since the power of the symbolic architectures, especially the fine-grained ones, comes from the interconnects; hence, increasing the flexibility of the nets by making them programmable goes a long way toward enhancing the overall processing power. Furthermore, the system designer is usually willing to trade some speed for the robustness of the slower programmable nets by exchanging hard-wired interconnects for switchable ones which can adapt the network pattern to the data. In symbolic processors, adaptability is even more important than for numeric processors because the topology of the data structures is usually irregular; therefore, the optimum network topologies cannot be determined prior to system design. In passing, it should be mentioned that programmability is also important from the fault tolerant standpoint since it permits system reconfiguration to bypass faulty processors.

The interconnect networks can be categorized into one of three general classes: multiplexed buses, multiport components, and switching networks. The bus is the simplest, and therefore the most popular, interconnect method because it involves the fewest number of switch elements. However, contention for these switches when many users (processors) are involved limits their utility mostly to loosely-coupled multiprocessor arrays for which the contention for bus resources is not as great as for the more tightly coupled systems.

The contention problem can be lessened by providing components (e.g., memories) with more than one port and by increasing the number of system buses such as shown in Figure 32. This, of course, increases the complexity of the components.

The third class of interconnects uses a network of switches that establish the communication paths around the system. The most general network, a generalized crossbar switch, is capable of establishing independent communication links between all components connected to the



P = Processor
M = Memory

Figure 32
Enhancement of Interconnection via Multi-port Memories

network; that is, there exists no sharing of switching elements between channels and therefore no contention for the switching resources. Although such a network represents the ultimate in interconnect power, its implementation cost in electronic hardware has proven too high for large systems in terms of cost, power requirements, and crosstalk avoidance. A $k \times m$ generalized crossbar has k input ports for connection to k data or message sending components and m output ports for connection to m receiving nodes. If each of n nodes of a system were to have one transmitting port and one receiving port (or instead, one bidirectional port), then an $n \times n$ crossbar would permit the n nodes to be fully interconnected and free of any contention for network resources.

For the sake of simplicity, a crossbar only of dimension 3×3 is illustrated in Figure 33. Note that the cost in terms of hardware for an $n \times n$ crossbar would be n^2 switches and $2n^2$ bidirectional communication links. For large n , implementation in electronic integrated circuit technology becomes a formidable problem, especially the design of the $2n^2$ bidirectional links with adequate bandwidth and an acceptable limitation on crosstalk. Also, electronic components have a very limited fan-out capability; for example, the fan-out limitation for an electronic gate is approximately ten other gates. The electronic solution has been to fall back to multi-stage switching networks. An example of one of many possible implementations (a baseline network) is shown in Figure 34a, where each switching element is limited to a fan-in of 2 and a fan-out of 2. The interconnect structure is a three-stage network - all switches in a vertical column would function as one stage of the network. This 8×8 interconnect network is composed of twelve 2×2 crossbar switches, the functions of which are illustrated in Figure 34b. Only a total of 48 switches ($12 \times 2 \times 2$) are needed instead of the 64 needed for an 8×8 crossbar. But the multi-stage design leaves the door open for contention. Numerous topologies (e.g., tree, banyan, delta, clos, mesh, to name a few) exist for interconnecting small-dimensional crossbars, and the choice is usually one between cost and an acceptable degree of contention. For example, if the degree of contention associated with a five-stage clos

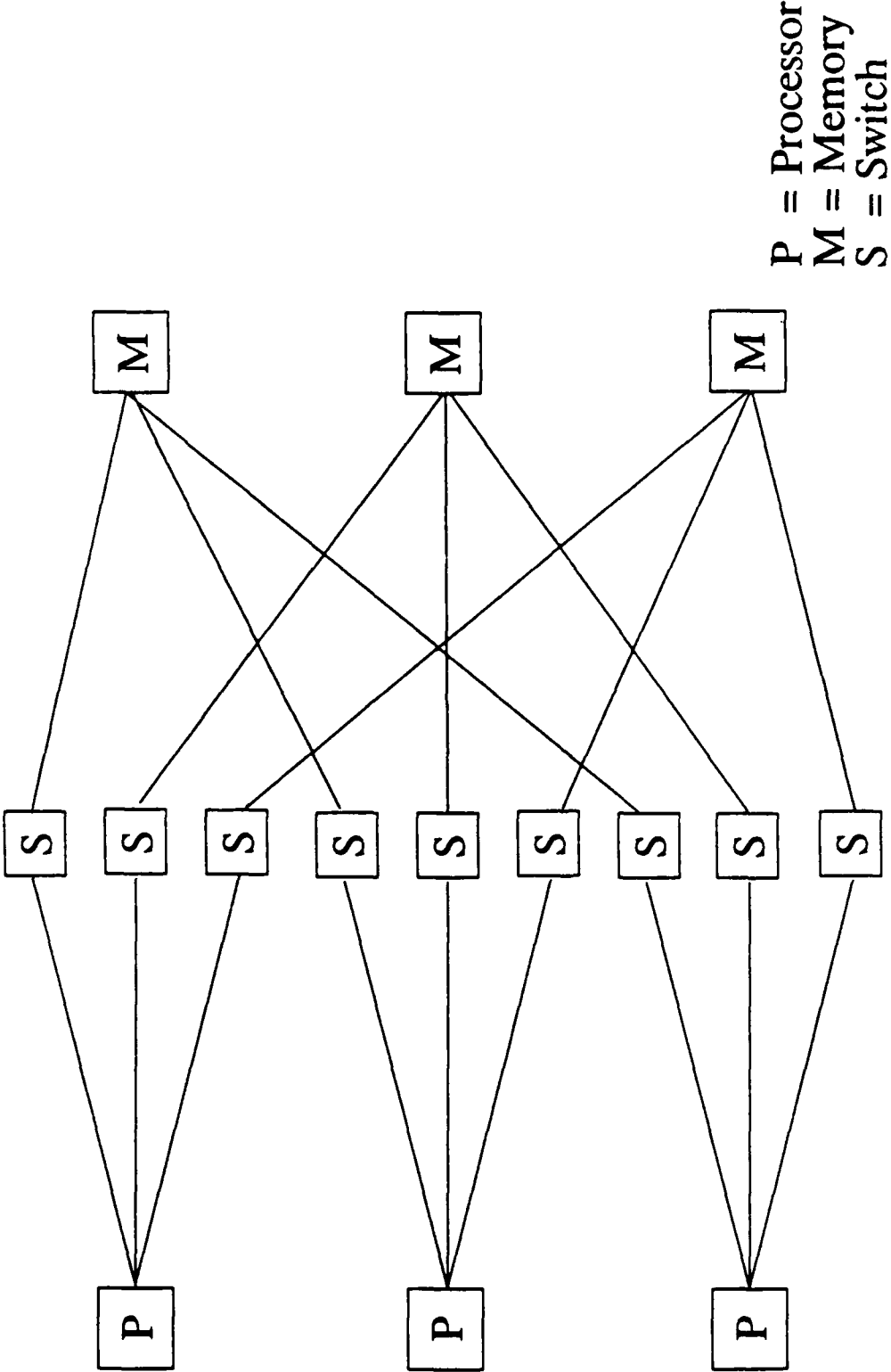
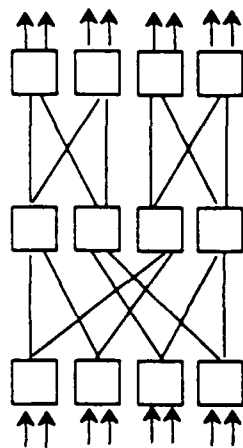
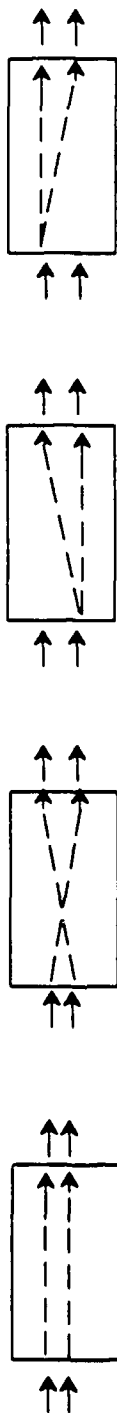


Figure 33
Processor/Memory Interconnect Employing a 3x3 Generalized Crossbar



(a) Baseline Network Topology



(b) Combinations of Switch Settings

Figure 34
A Multi-Stage Switching Network

network is acceptable, a 1000 x 1000 interconnect network would require only 146,300 switches rather than the one million required for the crossbar. A more thorough discussion of network topologies is given by Hwang & Briggs.²⁴

Given that a single generalized crossbar is not practical, the network designer must decide what topology best fits the data structure most likely to be encountered. As an example, a prime topology for interconnecting a parallel processor designed for low level image processing would be a grid structure enabling nearest neighbor interconnects since the vast majority of the low level operations involve adjacent image pixels. But as one moves toward the higher operations, regional relationships between the data become more important, requiring more global interconnect topologies.

C. OPTICAL IMPLEMENTATION OF MULTIPROCESSOR ARCHITECTURES

Any approach to optical architectures must give serious consideration to what can be accomplished with existing technologies, namely VLSI electronics. Optical computing will not seriously threaten electronic computing unless it can offer several orders of magnitude improvement in some critical measurement criterion, such as the power-speed-cost product, in a given problem domain. Therefore, a good starting point in addressing the application of optics to symbolic computing is to identify problem areas for electronics.

It is not surprising that the relative weaknesses and strengths of electronics and optics are traceable in one way or another to the fundamental physics of inter-electron and inter-photon interactions. Relatively speaking, the interactions between electrons are strong while that between photons are weak. Hence, electrons are good for the switching operations so fundamental to computing and photons are good for the inter-switch communications, providing links which are free from detrimental coupling effects that lead to crosstalk and capacitive loading. Subscribing to such reasoning, however, is impractical due to the quantum losses which accompany both the electron-to-photon and the photon-to-electron

conversions. There is research and development underway to replace some of the longer interconnect links within computers with optical channels because it is the longer interconnects that create severe power, speed, and space problems for electronics.²⁶ But such a capability stops far short of using optics to its full advantage in multiprocessor architectures appropriate for symbolic computing.

Consider the electronic-switching/optical-communications position as representing one of the four corners of the square shown in Figure 35 for which the sides of the square represent a continuum of combinations between the extremes of the corners. The upper left corner represents all-electronic systems while the bottom right represents all-optical. Since movement toward the bottom left corner is out of the question, the focus is along the upper and right sides. Upon considering computing systems for which switching is the predominant function, the tradeoff between optics and electronics is seen to fall somewhere along the upper edge; that is, all-electronic switching with some optical links. However, symbolic processing places a strong emphasis on communications as has been pointed out numerous times earlier in the Chapter. The de-emphasis on switching (fine-grained architectures) and the emphasis on communications (tightly-coupled systems) leads one to consider architectures for which the communications is optics and only some of the switching is done with electronics. It is this category of electronic/optical hybrid architectures that we believe will have a significant impact on symbolic computing.

Toward the upper end of this continuum would be those architectures which employ optical switching in the performance of reconfiguring the interconnects but which employ electronic switching for logic operations. As mentioned earlier, optics will prove to be especially valuable in providing the longer (more global) interconnects due to the power, speed, and space penalties associated with the longer electronic interconnects. An example of an architecture with such a designated mixture of optics and electronics is shown in Figure 36. For the sake of simplicity, the illustration shows only two of many possible boards and shows only four chips per board. If this were a fine-grained processor, each chip itself

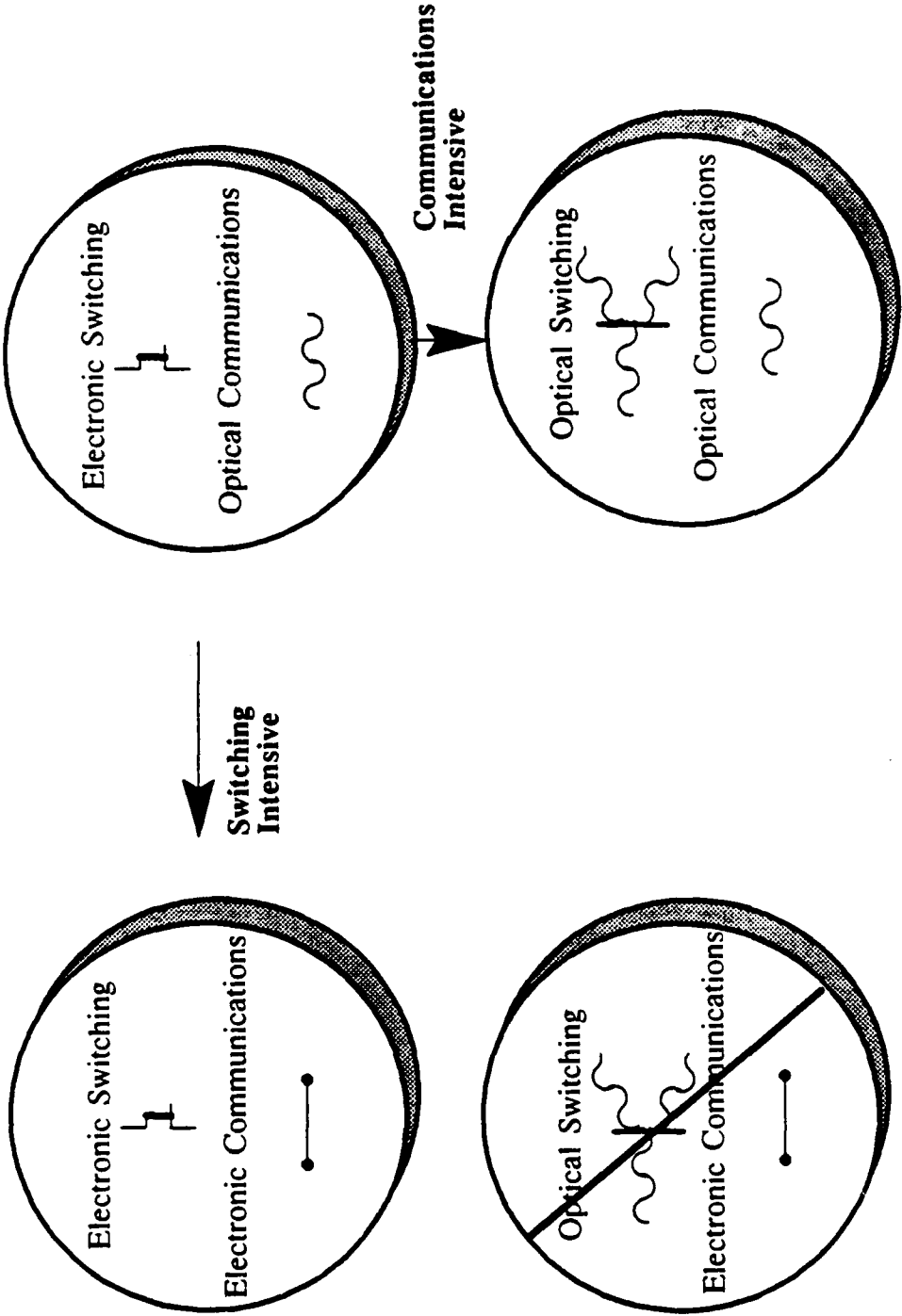


Figure 35
Electronic Versus Optical Computing

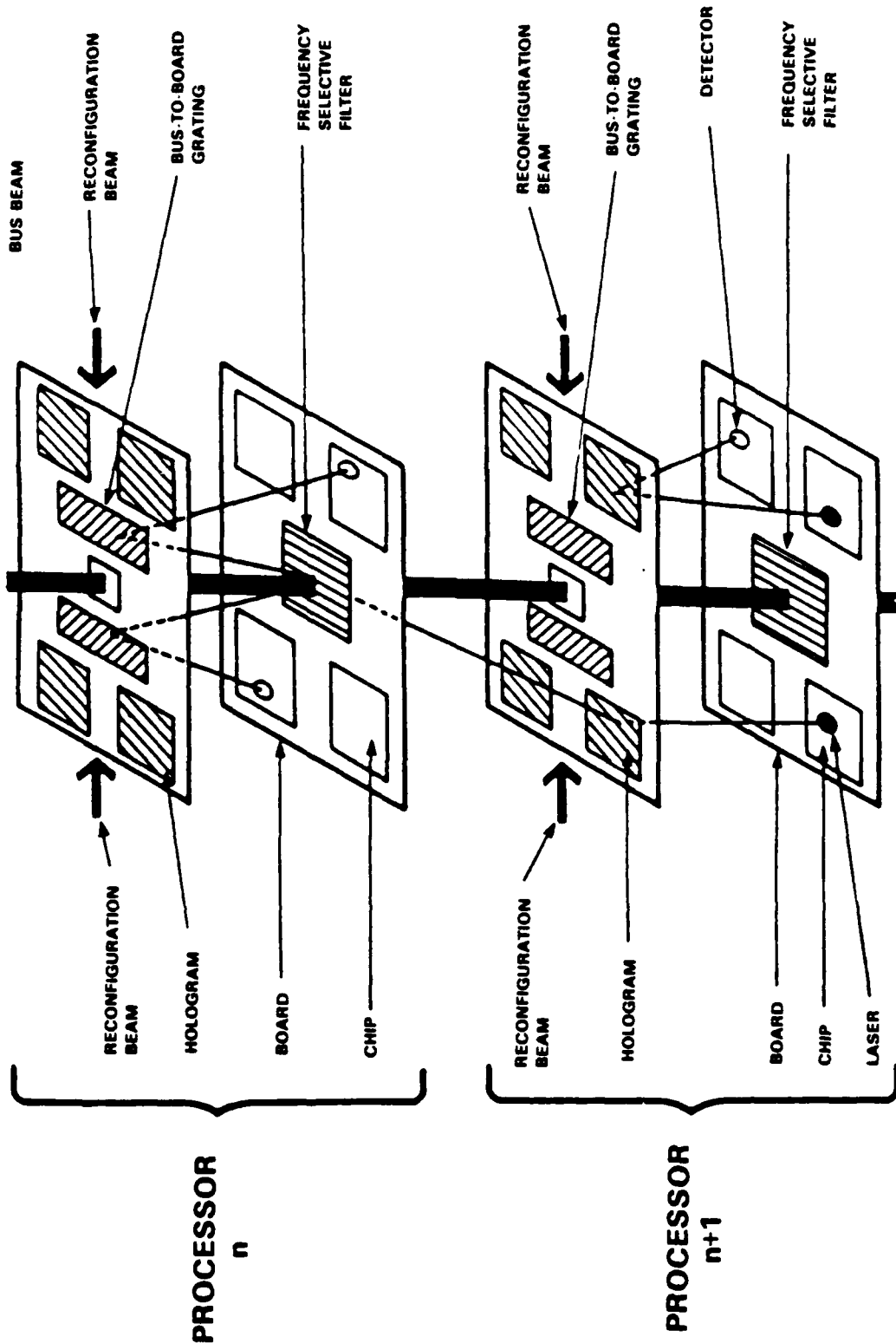


Figure 36
Hybrid/Optical Multiprocessor Architecture

would contain many processing elements (PEs). For example, one such electronic symbolic computer currently under development, called the Connection Machine,²⁷ is composed of a large array of printed circuit boards each of which contains 512 PEs equally divided between 32 chips (i.e., 16 PEs per chip).

Each board in Figure 36 contains four optoelectronic chips and one frequency selective filter (hologram). In between each board is a planar array of reconfigurable diffraction gratings that perform the majority of the switching operations involved in the interconnection process. This particular architecture employs wavelength division multiplexing (WDM) to direct optical bit streams to the appropriate board. The beam labeled /1 illustrates this operation. The hologram directly overhead of the transmitting chip directs the beam to the center of the next board where it is superimposed on the main beam which travels to all of the system boards. Upon reaching the intended board, the frequency selective filter diffracts the beam to a bus-to-board hologram which directs the beam to its final destination. The intra-board and intra-chip interconnects would be handled by the plane of holograms above the board as illustrated by beam /2. The logistics of handling a large number of multiplexed beams will not be discussed here other than to say that the optical switching most likely will be achieved through nonlinear wave mixing. For example, four-wave mixing may be used to generate holograms²⁸ which can be rapidly varied to permit interconnect reconfiguration. The reconfiguration beams shown in Figure 36 would contain the desired information for changing the holographic gratings. Note that some of the switching actions of such an architecture are performed by the multiplexing action, and the various holograms act as passive gratings that selectively direct the various wavelengths.

Such a versatile interconnect scheme based on directing light beams through free space contrasts with one of the most severe problem areas for electronic symbolic computing - that of implementing reconfigurable interconnects. At present, electronics depends on wires for all interconnects, limiting reconfigurability as well as fan-out capabilities. This places a

severe limitation on the switching network architectures, which is complicated by interconnect intensive problems, like those found in AI. The spectrum of optical networks, with their much greater versatility, will not be reviewed here; however, the interested reader is referred to publications of Sawchuk, et. al.²⁹

D. ALL OPTICAL ARCHITECTURES

As one moves toward the bottom right corner of the classification of Figure 35, the percentage of optical implementation increases until an all-optical architecture is achieved. An example of a fine-grained, tightly-coupled optical computer is shown in Figure 37. Although no one has built such a computer, it is technically possible to achieve such a system consisting of one million parallel channels. This does not mean that the system would be configured necessarily with one million nodes since such a configuration implies that the planar array of logic elements (designated as the gate array) would have just one logic element per channel. Instead, several logic elements would usually be interconnected via the interconnect media to form a processing element. For example, a square array of $n \times n$ logic elements (gates) may comprise an arithmetic logic unit, several registers, and possibly some cache memory. An example of this type of structure is shown in Figure 38, where individual elements in a 2-D SLM have been assigned the necessary functions to comprise a computational processing element. Taking an n of 5 (25 logic elements/processor) would lead to a machine with 40,000 nodes - large enough to be practical as a symbolic computer.

The input to the optical computer could be either through an array of independently addressable laser diodes or a two-dimensional spatial light modulator (2D SLM). The diode array would be capable of much higher modulation speeds, but would involve more complex circuitry, especially if operation requires uniformity over the complete array. If the input already exists as a two-dimensional light pattern such as might be output

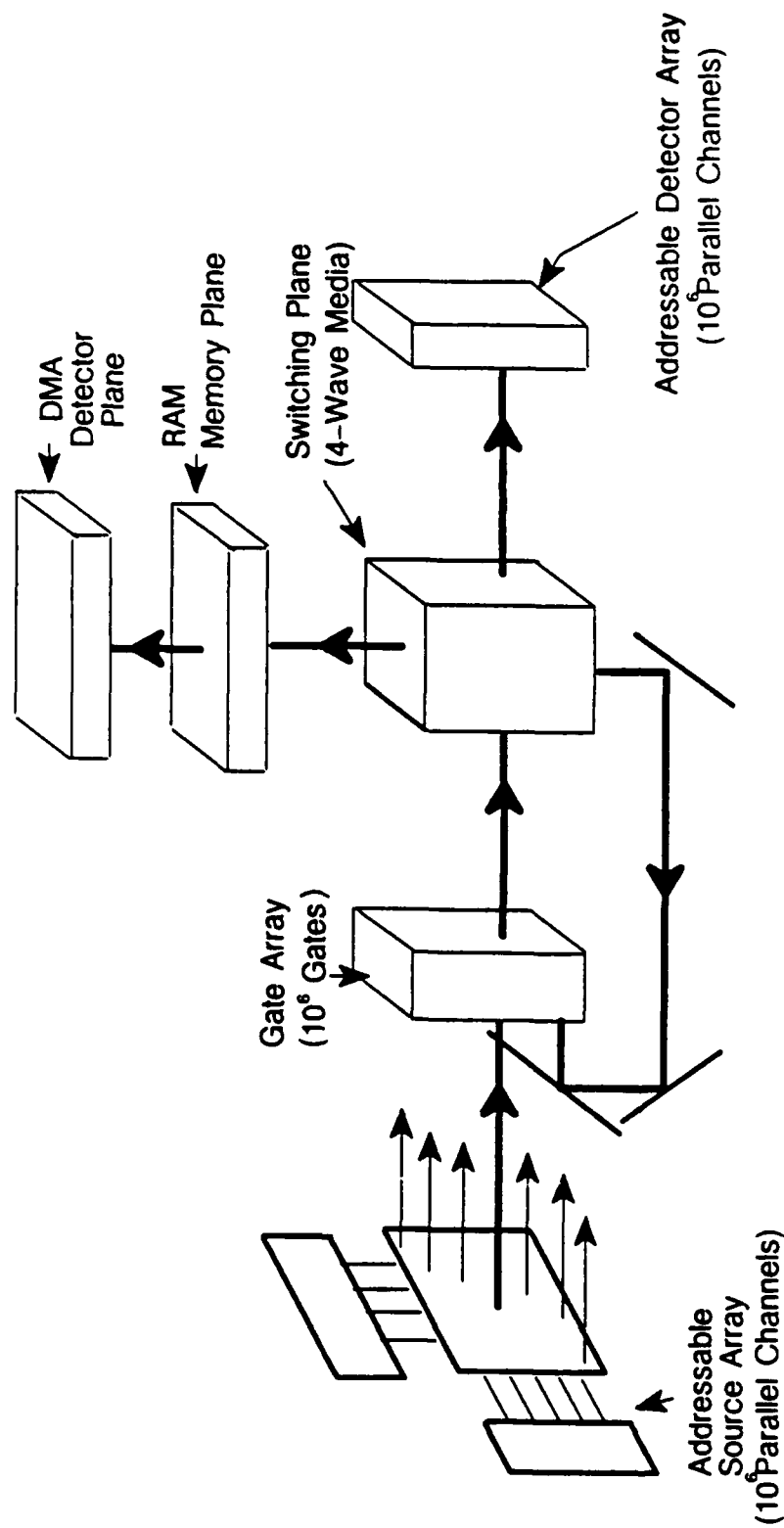


Figure 37
All-Optical Multiprocessor Architecture

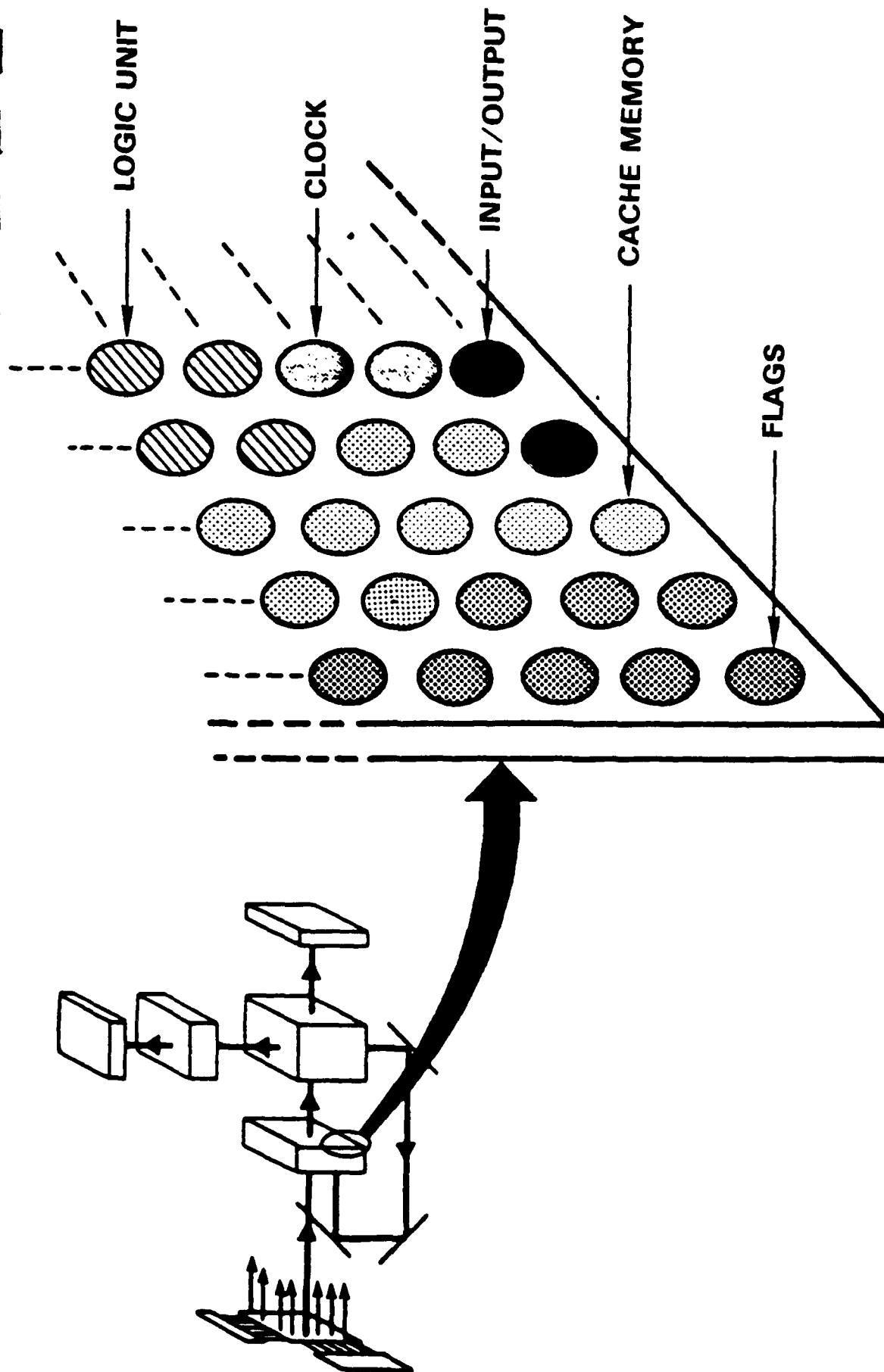


Figure 38
An All-Optical Processing Element

from a vision processor, an input device may not be needed depending on the compatibility of the two processors.

The logic element array could be either a 2D SLM exhibiting a nonlinear response or an array of optical bistable switches. The latter device will ultimately lead to much higher switching speeds, but current realizations of optical bistable switches have required impractical power levels. Improved nonlinear optical materials are currently under development for improved optical bistable devices.

The interconnect element will likely employ wave mixing in a nonlinear optical medium, similar in operation to that described previously for the hybrid architecture. However, due to the much larger number of channels that must be handled, the switching may be done in a multi-stage fashion in which multiple parallel planes of real-time hologram arrays would be exercised as illustrated in Figure 39. Note that, for the sake of simplicity, all three interconnect functions (processor/processor, processor/memory, and processor/IO) are combined into one block, but they could be implemented by three independent devices.

The detector will be a major technological challenge. In the most general case, one would like a one million channel device with each channel operating around 1 MHz (projected speed for 2D SLMs). However, the requirements will be much less for most practical processor designs. If the problem domain were to require, say, 100 iterations or more (e.g., semantic network searches to depths of at least 100), an output would be required only once every 100 microseconds. This reduces the throughput requirements for the detector to 1010 bits per second (bps), a number more in line with projections for GaAs microelectronics. Another example would be where each processor consists of a block of $n \times n$ channels as discussed above. Assuming an n equal to 5 and that each processor has just one output channel, the throughput requirement for the detector would be 4.0×1010 (bps). Some combination of these two designs should yield a detector requirement that would be well within technical feasibility.

The last major component of this opto-electronic architecture is the memory. The goal of co-locating much of the memory with the logic elements

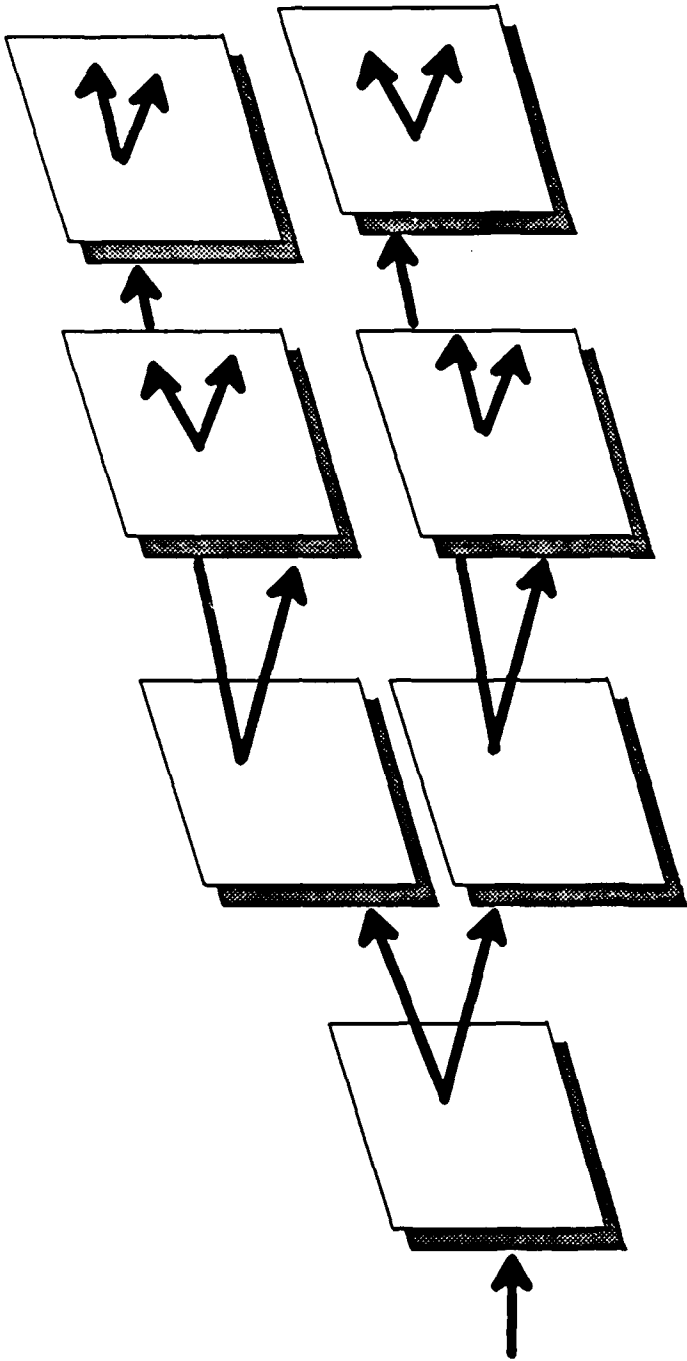


Figure 39
Multi-Stage Optical Interconnect Network

is not necessarily transferable to the optical computer domain because of the greatly reduced communications delays. Thus, Figure 37 shows the main memory as a single block, equally shared by all of the processors. Another important niche for optics is multiport memories. In fact, the use of multiple wavelengths could enable the read-out of any given memory location by a multitude of channels simultaneously, thereby avoiding the need for complex contention-resolving circuitry. For example, holographic gratings could be used to demultiplex the superimposed reflections of a multitude of wavelengths reflected from a given spot on an optical disk, or a holographic memory element could be used that would spatially separate the various read-out wavelengths. A way in which this could be implemented is shown schematically in Figure 40, where multiple beams could be used to address an optical disk simultaneously.

Another appealing attribute of using multiple wavelengths in optical computing is that the switching control is transferred to the information carrying beam itself rather than having to exist as a separate entity, adding greatly to the complexity of the computer control operations. This more closely parallels the operation of message routing systems in which initial bits of the message bit stream contain the address information which is used by each switch that the message encounters as it propagates through the network.

The processing power of the all-optical architecture could be enhanced through the use of pipelining. This could be achieved by replicating the logic element array as shown in Figure 41. Pipelining would be useful for multi-dimensional problems such as vision processing dealing with time-varying three-dimensional imagery (e.g., each plane could handle a different image depth).

By now, the reader should have an appreciation of the types of architectures required for symbolic computation, and of several ways for achieving them optically. It should be clear, however, that even these "all-optical" structures are in some sense hybrid optical-electronic architectures. In particular, electronics would be used for interfaces to the user, to digital controllers, etc, whereas the optics would be used to

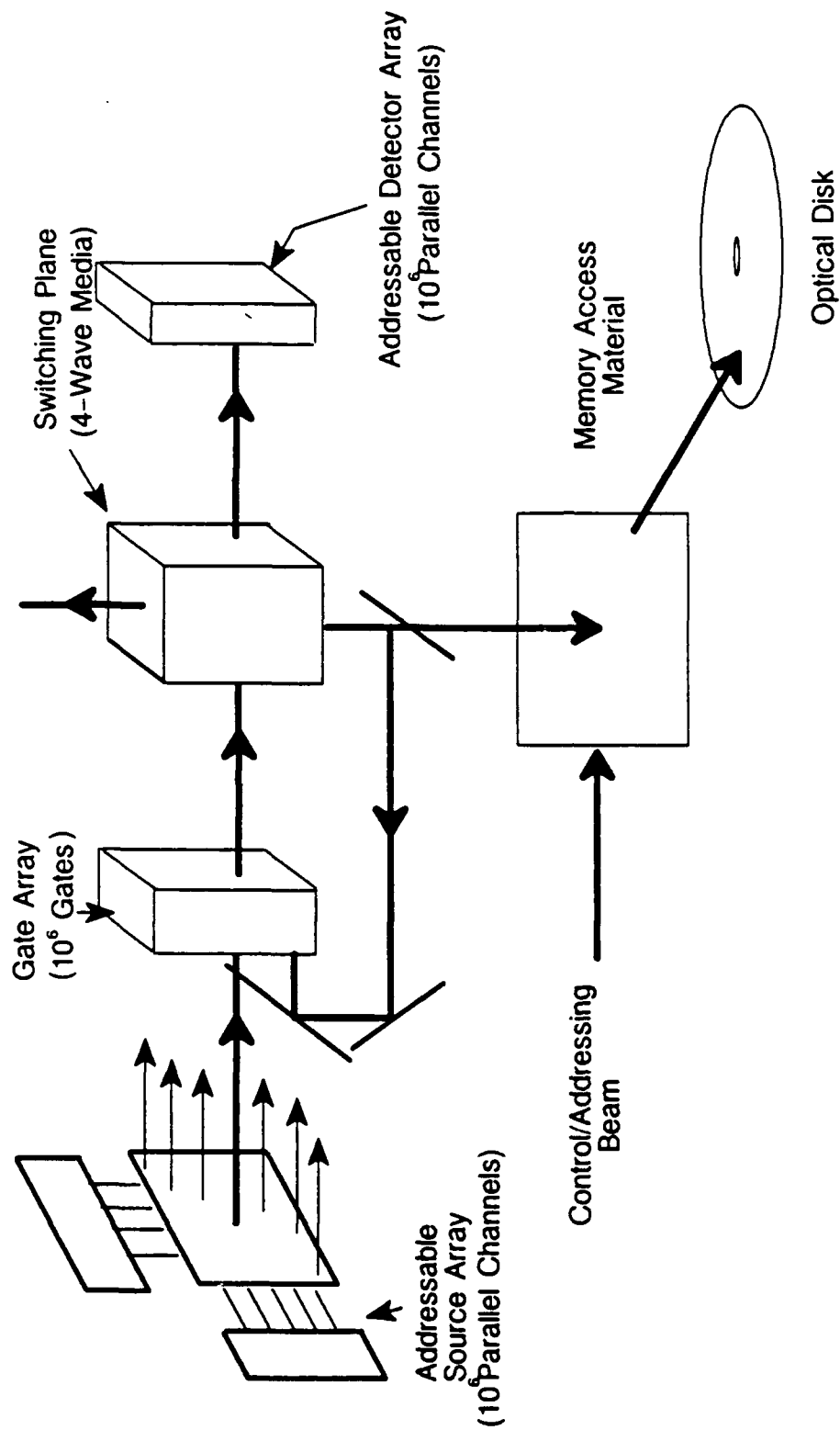


Figure 40
Optical Disk Interface to All-Optical System

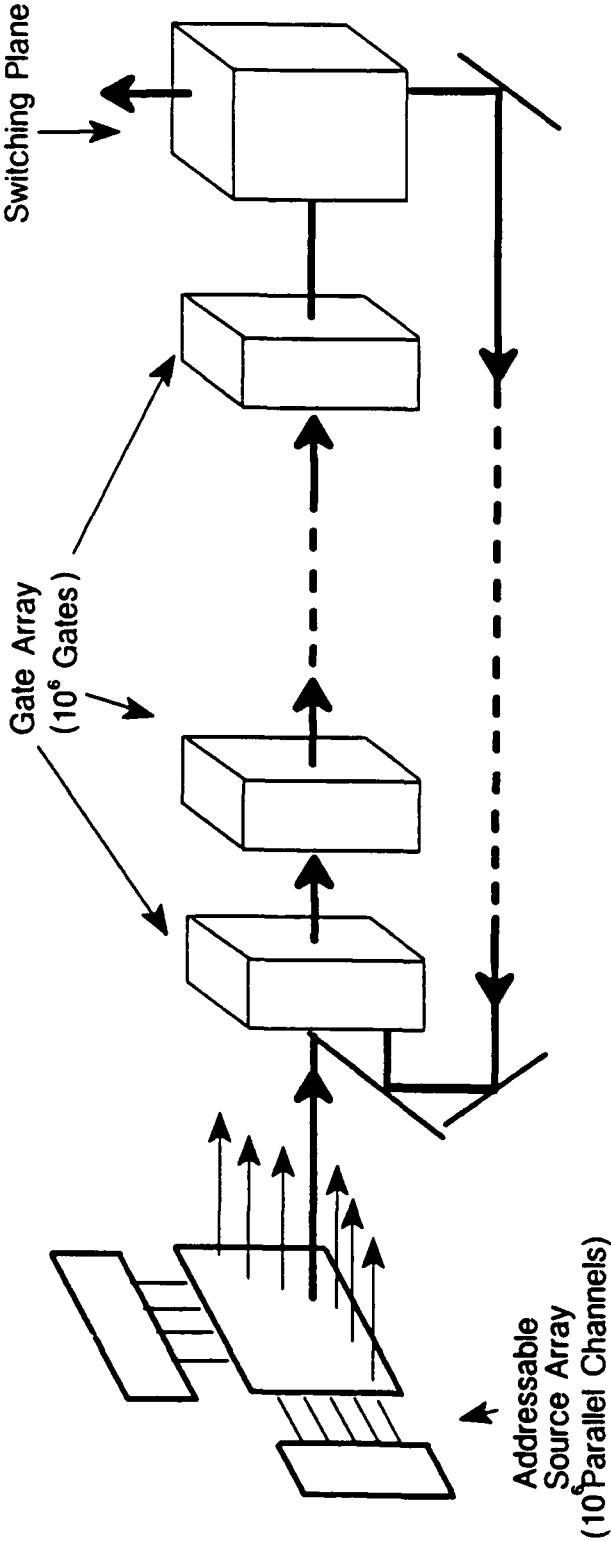


Figure 41
Pipelined Optical Gate Arrays

expedite the symbolic processing. With reference to Figure 36, we would expect that there is a spectrum of levels where the optical-electronic interface could occur. This spectrum of architectural possibilities, ranging from the all-electronic systems, to the hybrid optical-electronic systems, leads to other possible roles for the using optics in symbolic computation. Some of these possibilities will be explored in the following section.

E. HYBRID OPTICAL-ELECTRONIC SYSTEMS

There are several levels on which optics can be effectively combined with electronics. As shown in Figure 42, there is a hierarchy of functions in hybrid systems, ranging from replacing the processor for almost all operations, as in the all-optical systems of the previous section, to the use of optics only as a peripheral device, such as an optical disk for storage. The differences are in the degree of coupling between the electronic processor and the optical system, and in the amount of computation performed by the optics. In the following discussion, we are assuming that the electronic system is the host processor, and that the optical system is connected to it via one of the system busses.

At the lower end, the optical system would entirely replace the electronic system at the processor level, and would therefore perform almost of the computation and would interact strongly with the memory of the electronic system. Using the terminology of Section IV.B, such a hybrid system would be said to be tightly-coupled. At the next level, we have a structure where the optical system computes some of the primitives (multiplication, addition, subtraction, etc.), while the electronics processes others. An example of this could be an optical pattern matcher connected to the data bus of a LISP machine, where the optical system performed all the matching operations, leaving the electronics to compute other primitives. This is also a tightly-coupled system, since the processors share physical memory, and is analogous to the concept of an optical co-processor. The bandwidth of the interconnection network must be

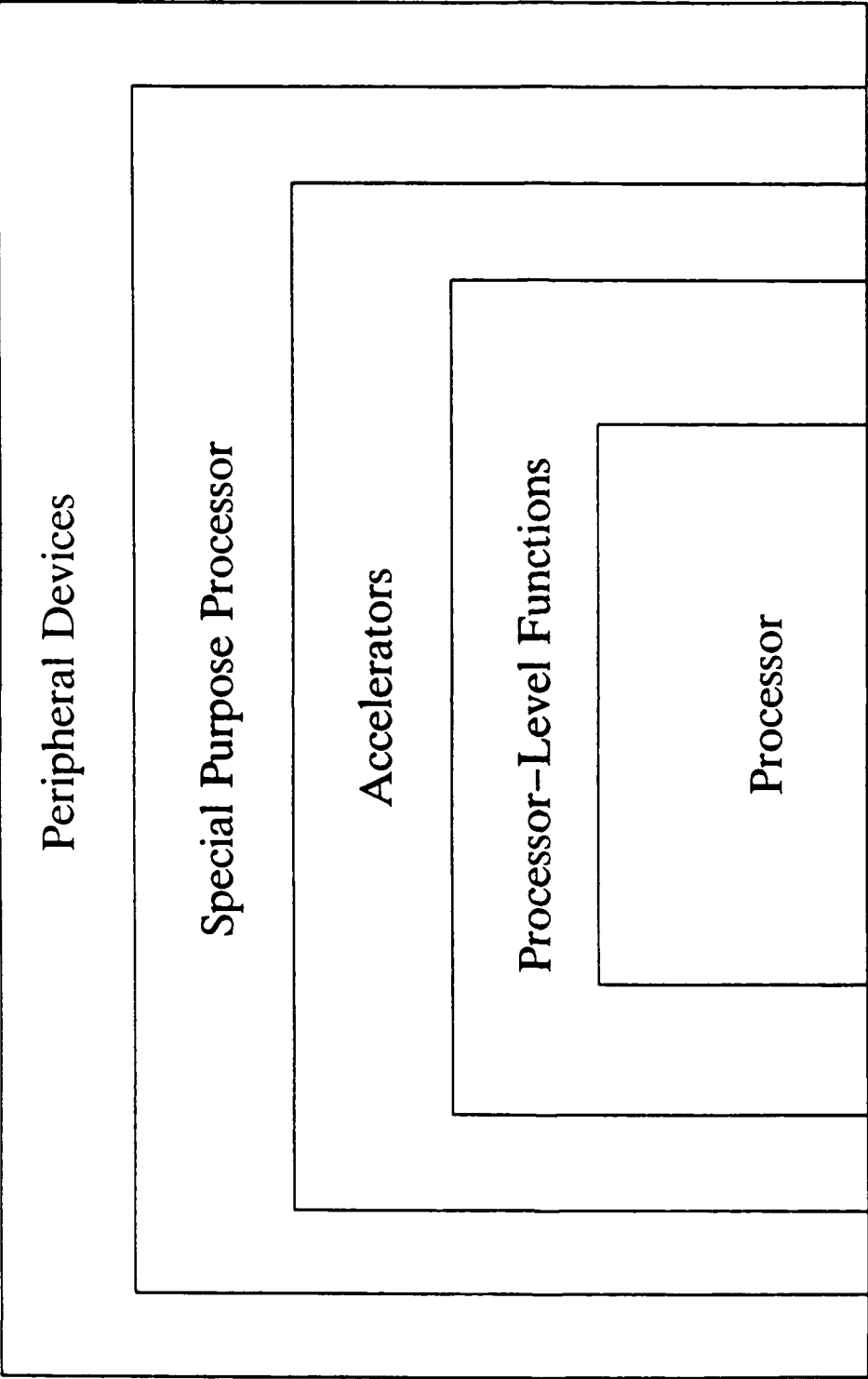


Figure 42
Hierarchy of Functions in Hybrid Systems

very high, and roughly equal to several times the memory access and transfer rate.

Accelerators are processors which dramatically increase the throughput of a particular function, such as an inner product, a correlation, or a rule firing. As we saw in Section III, inner products play a major role in AI processing, such as feature comparison, template matching, and correlation processing at the lower levels, and in knowledge base searches and inferencing at higher levels. Later in this section, we will look at how inner products can be applied to the processing of "if..., then..." types of rules.

Accelerators have had great impact in uniprocessor numeric computation, all but eliminating a number of previously troublesome computational bottlenecks. Surprisingly, they have not yet found widespread application in symbolic computing or in multiprocessor systems, and optics could help hasten that process. As in Figure 44, the optical computer could again be connected to the data bus of the system, but it does not share memory with the electronic system. This is an example of a more loosely-coupled hybrid system, where the host machine and the accelerator are proximally located but not necessarily within the same housing.

Special function processors (SFPs), as the name implies, have traded generality for performance, maximizing the throughput of a specific function. Typically they are very specialized computers with limited programmability, limited memory, and minimal interfacing requirements. As separate computing units, SFPs are connected to the host via a network, an optical fiber, or some other high-bandwidth medium. They are also referred to as computational arrays, and have been used successfully as feature extractors in low-level vision and speech, as display processors in computer graphics, and as array processors for computing FFTs. The most popular SFP is the systolic array, for which there is a rich base of experience, and a number of optical implementations as well.

For the remainder of this section, we would like to focus on two examples from the discussion above, namely the use of optical computing as an accelerator and as a special purpose processor. For the accelerator

case, we will use the example of inner product processing for "if..., then..." type rules. As a special purpose processor, we will look at the potential role of systolic array implementations of semantic net processing.

A central operation in symbolic computing is the inner product, which is equivalent to a multiplication of component elements in a vector (vector multiplication), in a matrix (matrix-matrix multiplication), or in a correlation function. In earlier sections, we identified the commonality of inner products in a large number of algorithms in the numeric computing domain. In one typical symbolic computing representation, knowledge relations are expressed in terms of logical pattern matching, such as determining the agreement of an antecedent condition (left-hand side) of an "if A, then B" relation (see Section III.E). Here A takes the form of a vector subspace of a N-dimensional vector space:

$A = \text{data/objects that belong to class A}$

which is spanned by some M vectors, where $M \leq N$:

$a_i(k) \in A, k = 1, \dots, M$

Membership in this subspace can be verified by means of a simple functional operation. For example, the null functional of a subspace is uniquely represented in terms of the vector a_A that is orthogonal to the subspace $a(k)$. Then, the inner product:

$a_A, a_i(k) = 0 \quad \text{for all } a_i(k) \in A$

Thus, a calculation, or, in this case, a rule firing, between knowledge elements in this representation and some appropriate functional may be expressible in terms of inner product functions. Furthermore, this construct could be used in either forward-chained or goal-directed reasoning, since in each case, the antecedent of an "if..., then..."

production rule must be satisfied during an inference. This is also true for frame-based representations, since, in that case, each knowledge element is a 2D array of information processed in its entirety, and the a_i s may take the form of matrix components. For each rule firing, the matching operation can be computed on the optical system very efficiently, alleviating a serious computational bottleneck in several AI systems.

Inner products are not the only operations for which optics may have a role. Systolic arrays, of which there have been several optical implementations, 7-9 have been shown to have definite mappings onto signal-flow graph networks. This implies that problems treatable or based on graph-theoretical techniques may have direct mappings onto well-defined systolic array topologies. In symbolic computing, graph theory analysis has been applied to developing relationships between definable objects and their attributes; this research has led to the semantic net representation.

For purposes of illustration, the semantic net can be viewed as a collection of nodes representing symbols, which are connected by links representing relations. The most fundamental relationship between symbols is the "IsA" link, and other types of relations could be "AtLocation," "MemberSet," and "Partof." Such relations are domain specific, and depend upon the taxonomy of the problem under investigation. In this representation, a basic question common to symbolic computation systems would be "Is A a B ?." If we assume that all connections between a general class of nodes S are constituted by "IsA" links, then this query is reducible to the problem of:

"For A a member of S ,"

"Is B also a member of S ?" and

"Is there a connectivity between A and B ?"

On a conventional architecture, this query would involve an extensive sequential search over all memory paths emanating from A . However, on a systolic system, the conclusion involves only the number of steps between

A and B , or to the edge of the array (for B not an element of S); this is because the search would proceed along all branches simultaneously.

To map this question onto the systolic system, we can define each node to reside on a processor or a pixel, and the links to adjacent nodes are the existing topological connectivity of the array. Here, the array is a 3-D construct with the third dimension being time. Node A can be tagged as the element of interest, and the resulting search towards B can occur as internodal bit stream propagation in each time step. This is shown schematically in Figure 43. While this still requires some test for the match condition, the set of semantic net problems should have some direct mapping onto systolic array systems.

In summary, optics offers several unique capabilities for the generic architectures discussed above, architectures which hold great promise for symbolic computing. The most important of these capabilities are: high speed global interconnects, interconnect reconfigurability, high fan-out elements, and multiport components for parallel processing.

F. ACKNOWLEDGEMENTS

The authors would like to express their gratitude to a number of personnel and co-workers who took time away from their own activities to work with us to improve the quality of our manuscript. While there are too many to cite individually by name, some deserve special credit. At the top of the list are Drs. Ravindra A. Athale, Teresa A. Blaxton, Roger A. Geesey and Lawrence H. Reeker, whose thoughtfulness and timely insights were extremely valuable. Many of their thoughts, explanations, and interpretations have found their way into our manuscript. John Perry, for his aid in the image understanding area and for agreeing to let us use some of his computer-generated images. Thomas S. Stark, for developing the glossary, and helping us bridge the "terminology-gap" that exists between optics and symbolic computing. And finally, Dr. Charles T. Butler, for assisting us in a number of ways, providing the authors with the time to develop our ideas more thoroughly.

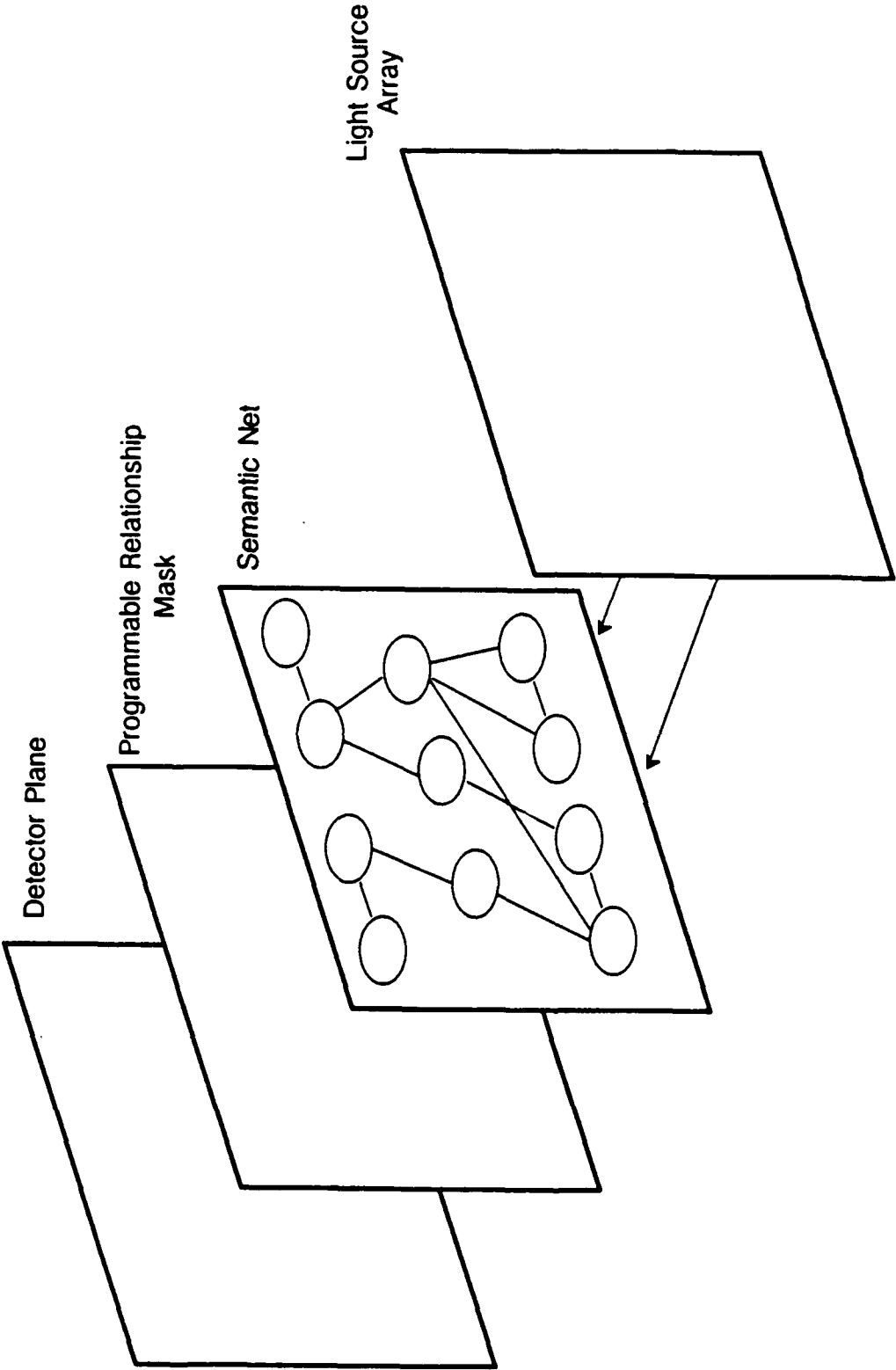


Figure 43
Parallel Processing of a Semantic Net

SECTION V
GLOSSARY

AI LANGUAGES AND TOOLS

A principle of Artificial Intelligence according to N. J. Nilssons Onion Model.

AGENDA

A part of an expert system that contains a prioritized list of knowledge based rules awaiting execution.

ANALOGICAL KNOWLEDGE

Knowledge that is represented by analogy in a computer. Examples of this are sound patterns representing words in a natural language processing system, or the representation of an image by a 2D array of numbers corresponding to steps on a gray scale.

ARCHITECTURE

The organization of the individual elements of a computer.

ASSERTION

A positive statement or declaration.

ATOM

A symbol (either constant or variable) used to identify an object in a LISP program.

ATTRIBUTE

Part of the description of an object contained in a frame. Attributes normally tell characteristics such as color, size, and value. The same as a slot.

GLOSSARY (CONTINUED)

BACKWARD CHAINING

A recursive procedure for problem solving in knowledge based systems. A progression by goal-driven inference is attempted between an assumed goal state and one or more initial states. If the progression is not possible, the goal state is altered. If the progression is possible, the initial state or states become goal states and the procedure begins again.

BACKTRACKING

A search procedure in which guesses are made about the direction to be taken through the solution space. When the guesses lead to an unacceptable result, the procedure backtracks to the point at which the incorrect guesses were made and begins the search procedure again in alternative directions.

BELIEF

A hypothesis about the outcome of some unobservable or uncertain situation.

BLACKBOARD

A part of many artificial intelligence systems in which intermediate or partial results of problem-solving are recorded.

GLOSSARY (CONTINUED)

BREADTH FIRST SEARCH

A search strategy used in knowledge based systems where the possible solutions to the problem are represented by a tree with nodes and branches. In breadth first search, all branches of the tree are examined at one node or level before moving to the next level. In this way, searching the breadth of the solution space is emphasized. See Depth First Search.

BOTTOM UP

A method of problem-solving which progresses from an initial condition to some desired condition. See Data directed inference and forward chaining.

CERTAINTY

A measure of the confidence placed by a user on the validity of a proposition, hypothesis, or inferential rule.

**COMMON SENSE REASONING
AND LOGIC**

A principle of AI from Nilsson's Onion model.

COMPUTATIONAL LOGIC

Logical reasoning done on a symbolic computer. This is the basis for non-numeric computations done in AI systems.

GLOSSARY (CONTINUED)

CONTINUOUS SPEECH	Normal spoken language. The majority of speech recognition systems accept either isolated or continuous speech.
CONTROL	The determination of the overall order or organization of problem solving procedures or activities.
DATA-DIRECTED INFERENCE	The type of inferences employed in forward chaining. By applying inference rules to supplied data or conditions a logical result is derived.
DECLARATIVE KNOWLEDGE	Knowledge consisting of facts or assertions.
DEPENDENCY	The relation between logical conclusions and the premises and inference procedures from which they were derived.
DEPTH-FIRST SEARCH	A search strategy in knowledge based systems in which possible solutions to the problem are represented by a tree with nodes and branches. In depth first search, a branch of related solutions is considered at all nodes before moving to the next branch. In this way, the depth of an assumed class of solutions can be examined before moving to the next branch. See Breadth first search.

GLOSSARY (CONTINUED)

DOCUMENT GENERATION

One of many proposed applications of natural language processing systems. After information is stored in a computer, the computer generates a document containing the information.

DOCUMENT PREPARATION

Another proposed application of natural language processing systems. NLP systems act as experienced editors, checking for errors in spelling and grammar and suggesting ways to rephrase text.

DOCUMENT UNDERSTANDING

A proposed application of NLP systems in which a document is read and its contents are assimilated by the system.

**EXPECTATION-DRIVEN
REASONING**

A control procedure that uses expectations to formulate hypotheses about unobserved situations. See backward chaining and goal-directed inference.

EXPERT FRAMEWORK SYSTEMS

The knowledge representation and reasoning mechanisms of an expert system without the domain specific knowledge base.

EXPERT SYSTEM

A computer system that achieves high levels of performance in areas that for human beings would require years of special education and training.

GLOSSARY (CONTINUED)

**EXPERT SYSTEM DEVELOPMENT
ENVIRONMENT**

The knowledge representation and reasoning mechanisms of an expert system without the domain specific knowledge base.

EXPERTISE

The capabilities that enable high performance in a particular area. In the context of AI systems, this includes tools that enable intelligent operation such as meta-knowledge, heuristic search procedures, and inference rules.

EXPLANATION SUBSYSTEM

A part of an expert system that rationalizes or explains its conclusion by providing a summary of the inference rules and data that it used to arrive at the conclusion.

FACT

A piece of declarative knowledge.

FORWARD-CHAINING

A procedure for problem solving in knowledge based systems. The procedure progresses from the data given by the user to a solution that adequately supports it. See data driven inference.

FRAME

Data structures that represent objects by a list of properties and relations to other objects.

GLOSSARY (CONTINUED)

FUZZY LOGIC

An approach to inexact reasoning consisting of a proposition and a fuzzy set. The fuzzy set contains ranges of possible values the proposition can have and numerical values corresponding to the probability of occurrence in each range.

GOAL-DIRECTED INFERENCE

The type of inferences employed in backward chaining. Emphasis is placed on examining the states and inference rules which produce a desired goal. See backward chaining and expectation-driven reasoning.

GRAY SCALE

Analog or digital numbers corresponding to shades of gray. The maximum and minimum numbers represent white and black.

HEURISTIC

Of or relating to problem solving procedures that utilize self educating techniques to improve their performance.

HEURISTIC SEARCH

A principle of AI from Nilsson's Onion model.

GLOSSARY (CONTINUED)

HIERARCHICAL PLANNING

An approach to planning in which computer time and memory are saved by considering only high level details of the plan. Vague and lower level details are then formulated into subplans.

HUMAN ENGINEERING

The process of engineering man-machine interfaces to achieve maximum utilization of that machine.

INFERENCE

The process of passing from a proposition whose truth is established to another whose truth is believed to follow from that of the former.

INFERENCE ENGINE

A part of an expert system containing the procedures it will use to solve problems.

INFERENCE RULES

Methods or strategies employed by AI systems for solving problems by logical inference.

ISOLATED SPEECH

Speech that has pauses between the words. The majority of speech recognition systems accept either isolated or continuous speech.

GLOSSARY (CONTINUED)

KNOWLEDGE

Stored information for use in the solution of AI problems. In the case of expert systems, knowledge is thought of as facts, beliefs, or heuristic rules. In the case of speech and image recognition systems, knowledge is thought of as; stored pattern, image, or word data, beliefs, and heuristic rules.

KNOWLEDGE ACQUISITION

The extraction and formulation of knowledge for use in AI systems. The AI computer equivalent of learning.

KNOWLEDGE BASE

The repository of knowledge in an AI computer system.

KNOWLEDGE ENGINEERING

A discipline associated with the building of expert systems.

**KNOWLEDGE PROGRAMMING
TOOLS**

Software that allows an expert who knows little about knowledge engineering to program knowledge into an AI system.

KNOWLEDGE REPRESENTATION

A principle of AI from Nilsson's Onion model.

GLOSSARY (CONTINUED)

LEARNING	The process of improving the performance of an AI system by using past experience to alter its stored knowledge or problem solving strategies.
LEXICON	A list of morphemes contained in a NLP system's knowledge base.
LOGIC ORIENTED PROGRAMMING LANGUAGES	Programming languages whose purposes are to program and solve logic problems.
LISP	The most widely used AI programming language. LISP was developed by John McCarthy at MIT in 1958. LISP is an object oriented programming language.
LISP MACHINES	Computing architectures dedicated to executing LISP programs.
LIST	A series of elements (either atoms or other lists) enclosed in parentheses in a LISP program.
MACHINE TRANSLATION	A proposed application of natural language processing systems for the translation of a document from one language to another.

GLOSSARY (CONTINUED)

META	A prefix used with AI subjects to denote the existence of knowledge about the base word or subject, as in meta-knowledge, which is knowledge about the system's knowledge base.
MORPHEMES	A basic linguistic unit having meaning.
MORPHOLOGICAL ANALYSIS	A technique used in natural language processing. The meaning and use of a word is determined by separating the word into parts and determining the meaning of each part.
NOISY DATA	Data having characteristics that introduce uncertainty into the reasoning processes of AI systems.
OBJECT-ORIENTED PROGRAMMING LANGUAGES	Programming languages whose purposes are to portray the characteristics and relationships between objects.
PARSING	The act of breaking a sentence or phrase into its component parts in order to identify the form, function, and syntactical relationship between each part.

GLOSSARY (CONTINUED)

PRAGMATICS

Knowledge about human discourse and conversations, concerning the overall context in which sentences or phrases are written or spoken and how the various phrases are related to each other.

PREDICATE CALCULUS

A formal language of symbol structures for representing facts.

PREDICATE LOGIC

Logical operations that, through the manipulation of propositions, allow one to make assertions.

PROCEDURAL KNOWLEDGE

Knowledge about procedures or actions, typically specified as If..., then.... types of production rules.

PRODUCTION RULES

Two-part statements that specify a certain action to be taken when an antecedent condition is satisfied, usually in the form of If..., then... types of rules. Procedural knowledge is contained in production rules.

PROLOG

PROgramming in LOGic, was developed by A. Colmerauer and P Rousset at the University of Marseille in 1973. Prolog is a logic oriented programming language.

GLOSSARY (CONTINUED)

PROPERTY LIST

A construct in a LISP program that associates a property and a corresponding value with each atom. Property lists describe the "state of the world" in an AI program and therefore are updated frequently.

PROPOSITIONAL LOGIC

Logic which, through the use of propositions, determines the truth or falsehood of other propositions.

PRUNING

The act of eliminating a solution or group of solutions from a problems solution tree.

RULE

"If...Then" statements that support deductive reasoning.

RULE SET

A collection of rules that constitutes a module of heuristic knowledge.

SCHEDULING

Determining the order of execution of processes in AI programs.

SCRIPTS

Data structures that represent sequences of events. Scripts represent procedural knowledge and are similar in principle to frames.

GLOSSARY (CONTINUED)

SEMANTICS

The non-literal interpretation of word meanings. Knowledge in this area is used extensively in speech and natural language interpretation.

SEMANTIC NETWORK

A scheme for representing relationships between objects in an AI system's knowledge base in terms of class equivalence and inheritances.

SLOT

A single description of an object in a frame. Slots can contain information such as name, color, definition, or value.

SOLUTION SPACE

A conceptual way of thinking of the possible solutions to a problem, which has a direct bearing on the number of branches that an AI system can search in the course of solving a problem.

SOLUTION TREE

The representation of the possible solutions to an AI problem by a tree with nodes representing solution types, and branches representing their relationships.

**SPEAKER DEPENDENT SPEECH
RECOGNITION SYSTEMS**

Speech recognition systems that can only accept input from a particular speaker.

GLOSSARY (CONTINUED)

**SPEAKER INDEPENDENT SPEECH
RECOGNITION SYSTEMS**

Speech recognition systems that can understand input from any speaker without being trained to each individual voice.

SPEECH RECOGNITION

The recognition of spoken language by a computer system.

SYNTAX

The use of words to form phrases, clauses, and sentences.

TEMPLATE MATCHING

One way in which images and spoken words are identified. Arrays representing frequency (sound for words, light for images) or intensity versus time are matched against those of known images and words for similarity.

TOP DOWN

An approach to problem solving which moves from some current condition to an initial condition. See **BACKWARD CHAINING** and **GOAL DIRECTED INFERENCE**.

TRUTH MAINTENANCE

The act of maintaining truth or consistency in the elements of a knowledge base.

VISION

The understanding and interpretation of scenes or images by a computer system.

WELL FORMED FORMULA (WFF)

A syntactically valid statement in predicate calculus.

SECTION VI

REFERENCES

1. C. L. Forgy, et al, "Opportunities for Parallelism in AI Tasks," Carnegie-Mellon Reports, 1985.
2. H. Winston, Artificial Intelligence, Addison-Wesley, Reading, MA 1984.
3. E. Charniak, D. McDermott, Introduction to Artificial Intelligence, Addison-Wesley, Reading, MA, 1985.
4. Fennell and Lesser, "Parallelism in Artificial Intelligence: A case study of Hearsay II," IEEE Trans On Computers, Volume C-26, No. 2 (February 1977) 98-111.
5. D. Marr, Vision, Freeman, San Francisco, 1982.
6. A. Barr, E. Feigenbaum, et al, eds. The Handbook of Artificial Intelligence, 3 Volumes, Kaufman, Los Altos, CA 1981.
7. R. Lindsay, B. Buchanan, E. Feignebaum, J. Lederberg, Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project, McGraw-Hill, New York, 1980.
8. J. McDermott, "R1: The Formative Years," AI Magazine, 2, No. 1 (Spring 1981) 21.
9. E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, American Elsevier, New York, 1976.
10. F. Hayes-Roth, D. Lenat, D. Waterman, Building Expert Systems, Addison-Wesley, Boston, MA, 1983.
11. H. L. Andrews, "Speech Processing," IEEE Computer, Volume 17, No. 10 (October 1984) 315.
12. R. Balzer, L. Erman, et al, "HEARSAY-III: A Domain Independent Framework for Expert Systems," Proc. AAAI-1 1980
13. D. H. Ballard and C. M. Brown, Computer Vision, Prentice Hall, Englewood Cliffs, NJ, 1982.
14. R. D. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
15. F. Itakura, Minimum Prediction Residual Principle Applied to Speech Recognition, IEEE Trans. Acoustics, Speech, and Signal Processing, Volume ASSP-23, (February 1975) 67.

REFERENCES (CONTINUED)

16. See, for example, references in Artificial Intelligence, 17, No. 1-3, (Special Issue on Vision) 1981.
17. K. Fu and A. Rosenfeld, "Pattern Recognition and Computer Vision," IEEE Computer, Volume 17, No. 10 (October 1984) 274.
18. E. Rich, Artificial Intelligence, McGraw-Hill, New York, 1983.
19. R. Davis, D. Lenat, Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill, New York, 1982.
20. See, for example, T. A. Blaxton and B. G. Kushner, "An Organizational Framework for Building Adaptive Artificial Intelligence Systems," in preparation.
21. B. G. Buchanan, D. H. Smith, et al, "Applications of Artificial Intelligence for Chemical Inference XXII: Automatic Rule Formation in Mass Spectrometry by Means of the Meta-DENDRAL Program," Journal of the American Chemical Society, Volume 98, (1976) 6168.
22. D. Lenat, "EURISKO: A Program that learns New Heuristics and Domain Concepts," Artificial Intelligence, 21, No. 1, (January 1983) 31.
23. J. R. Anderson, The Architecture of Cognition, Harvard University Press, Cambridge, MA, 1983.
24. K. Hwang and F. A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill (1984).
25. C. L. Seitz, "Concurrent VLSI Architectures," IEEE Transactions on Computers, Volume C-33, No. 12 (1984).
26. J. A. Neff, "Optical Interconnections Between Integrated Circuit Chips," Proceedings of the International Electronics Packaging Conference, Orlando, FL (1985).
27. W. D. Hillis, The Connection Machine, The MIT Press (1985).
28. D. M. Pepper, "Nonlinear Optical Phase Conjugation," Optical Engineering, Volume 21, No 156 (1982).
29. A. A. Sawchuk, B. K. Jenkins, C. S. Raghavendra, & A. Varma, "Optical Interconnection Networks," Proceedings of the IEEE International Conference on Parallel Processing, St. Charles, IL (1985).

REFERENCES (CONTINUED)

30. H. J. Caulfield, W. T. Rhodes, M. J. Foster, and S. Horvitz, "Optical Implementation of Systolic Array Processing," Opt. Commun., Volume 40, No. 12 (December 1981) 1986.
31. P. S. Gulfoyle, "Systolic Acousto-optic Binary Convolver," Opt. Eng., Volume 23, No. 1 (January/February 1984) 20.
32. S. Cartwright, S. C. Gustafson, "Convolver-based Optical Systolic Processing Architectures," Opt. Eng. (January/February 1985) 59.
33. John Perry, "Image Understanding by Computers: Algorithms and Implementations" BDM Technical Report BDM/W-86-0131-TR, February.
34. J. L. Baer, "Computer Architecture," IEEE Computer, Volume 17, No. 10 (October 1984) 77.

PART II

TABLE OF CONTENTS

Chapter		Page
I	Program Philosophy	1
II	Optical Relational Algebra Machines	9
III	Pattern Matching in Symbolic Computing	19
IV	Summary	32
Appendix A	Optical Implementation of the Compare-and-Exchange Operation for Applications in Symbolic Computing	A-1

CHAPTER I

PROGRAM PHILOSOPHY

The field of optical techniques in computing and information processing can be said to have started in the late 1950's with the invention of synthetic aperture radar and the optical processor to reconstruct images from scrambled radar returns. Since that time the field has undergone a significant evolution in terms of the mathematical framework employed, the architectural concepts developed and the active and passive device technology that emerged. The computational problems and the application areas that optics attempts to tackle have undergone radical changes as well. This shift in focus is governed as much by the emergence of new computational problems to be solved as by the development in the optical hardware and the invention of new architectural concepts. A comprehensive historical discussion of optical information processing can be found in an article by J. W. Goodman (1). An admittedly subjective analysis of some important milestones in optical information processing and computing is presented in the first section of this chapter. The application of optics to problems in symbolic computing will then be viewed according to the perspective provided by our historical analysis. Finally a brief outline of the technical report will be provided.

A. HISTORICAL OVERVIEW OF IMPORTANT MILESTONES IN OPTICAL COMPUTING

1. Optical Pattern Recognition

Optical pattern recognition research began at almost the same time as the synthetic aperture radar processor and is identified with Vander Lugt's classic paper in 1963 on holographic matched filtering. His system is based on the characteristics of electromagnetic wave diffraction and the mathematical formulation of the correlation operation in the Fourier domain. The matched filter correlator is a very powerful system that is capable of performing the 2-D correlation between high resolution images (500X500 pixels) using a very simple and passive optical system with only input and output interfaces (an incoherent-to-coherent converter/spatial light modulator and a 2-D detector array, respectively) limiting the frame rate, resolution, and power consumption of the overall system. The 2-D correlation is essentially a template matching operation which has the obvious limitations of sensitivity to changes in rotation, scale, and other statistically insignificant changes in the objects to be recognized. In the last 20 years several workers have attempted, with some success, to improve the robustness of the optical pattern recognition systems to various distortions (rotations, scale) by utilizing reference images

that are not simple templates of the objects to be recognized (2). In spite of its limitations, however, the classical matched filter correlator is useful when a very well defined pattern is to be identified and located within a complex scene.

2. Acoustooptic Processing

Acoustooptic processing is an example of a technology-driven development. The operation of an acoustooptic Bragg cell is based on the interaction between high frequency (tens of MHz to GHz) sound waves induced in a transparent crystal by an applied RF signal and an optical beam traversing the crystal. This device provides a direct and simple transduction mechanism between RF electrical signals and coherent optical signals. The propagating nature of acoustic waves in the crystal leads to a built-in mechanism for scrolling the temporal signals across a finite size window. Both of these features have led to applications of optical techniques to significant signal processing problems in radar, sonar, and communications systems. Acoustooptic processors for performing spectrum analysis and convolution/correlation have reached a high level of maturity. A recent book edited by Berg and Lee gives a thorough account of the algorithmic, architectural, technological, and applications aspects of acoustooptic processing (3). In the last five years the scope of acoustooptic processing has broadened considerably with the invention of several novel architectures by Psaltis and his associates at CalTech. These architectures are designed to exploit the highly developed acoustooptic device technology, which is inherently 1-D, for 2-D signal processing problems such as image processing and synthetic aperture radar reconstruction. This example indicates how architectural innovations interact with technological advances to open new application domains.

3. Optical Matrix Processors

Optical pattern recognition and acoustooptic processing systems are based on mathematical techniques developed for analyzing linear shift-invariant systems. The fundamental operations involved in that framework are Fourier transform and convolution or correlation. Since the propagation of electromagnetic waves are governed by equations that are also linear shift-invariant, Fourier transform and convolution or correlation operations were natural choices for the initial work in analog optical processing. To broaden the domain of applications of optical processing, it is desirable to relax the constraints on the mathematical framework. The first step in that direction is the elimination of the shift-invariance requirement

from the problems. The mathematical framework of matrix algebra is most convenient for this set of problems and has the additional advantage of having been studied extensively over the past hundred years by a number of mathematicians. The basic operations of matrix algebra involve scalars, vectors, matrices, and rules for addition and multiplication between these entities. These fundamental operations are characterized by the need for global, one-to-many, many-to-one, and many-to-many types of interconnects between the elements of 1-D or 2-D arrays which represent vectors and matrices respectively. Since optical systems provide analog multiplication, analog addition and the desired interconnects, they have been investigated intensively over the last 10 years for implementing the elementary matrix operations and more complicated algorithms (4). Although the original motivation of these studies was to apply optical processing to linear transforms and processing of signals from sensor arrays, the basic approach is also relevant to a wider variety of problems in image processing and numerical computing.

4. Digital Optical Computing

In the previous three sections we have discussed optical systems that use analog representation of data. The systems were also based on linear operations of multiplication and addition. The most prominent drawback of these two features is the limited computational accuracy achievable with analog data encoding (6 to 8 bits) and a relatively narrow problem domain that a given system can handle without a major redesign. Both of these drawbacks can be remedied through use of a digital representation of data (most commonly the binary number system) and through nonlinear optical devices performing Boolean logic operations. The digital optical computers employ architectural concepts that are variants of those which are ubiquitous in electronic digital processors. Although this approach to optical processing also dates back to early 1960's, much excitement has been generated by the recent progress in optical bistability (5). Novel nonlinear optical materials are being developed and optical logic devices are being demonstrated which operate at picosecond switching speeds and femtojoules of switching energies. Significant progress has also been made on fabricating 2-D arrays of these logic devices and on innovative approaches to digital computer design which better utilize the parallelism and arbitrary interconnection characteristics of optical systems. These systems have not yet been applied to solutions of specific problems, and the support issues of appropriate programming languages, operating systems, memory management,

and communications protocols which are critical to electronic digital multiprocessors have received little attention for digital optical processors.

5. Optoelectronic Interconnects for Electronic Multiprocessors

The development of optoelectronic interconnects for electronic multiprocessors is based on the philosophy that optics is inherently better suited for communications while electronics is more suited for switching and logic. Hence, a hybrid system that employs optics and optoelectronics for long distance, global, high bandwidth, and reconfigurable interconnects between the components of an electronic multiprocessor should be optimal (6). The interconnects can be employed at the intra-chip, inter-chip, board-to-board, or subsystem-to-subsystem level. The relevant optoelectronic technology is being driven by the optical communications industry through its need for high reliability, high speed light sources, modulators, detectors, transmission media, and networks. The tradeoff considerations and performance requirements of the optoelectronic interconnects are, however, significantly different from those of the telecommunication industry and hence research programs are currently focussing on the specific component requirements of the inter- and intra-processor communication.

6. Optical Neural Net Architectures

Optical neural net architectures is an exciting new field of optical processing that has been investigated during the past 3 years. The main direction of research activity in this field has been in optical architectures to implement the various models of neural nets in animals and humans as proposed by psychologists, neurophysiologists, and applied mathematicians. There is a belief that the neural nets in animals and humans consist of very large numbers of relatively simple and slow processing elements (neurons) that are massively interconnected and that operate in parallel to generate the extraordinary system behavior of perception and cognitive processing. These neural characteristics seem to be well-matched to the features of analog and digital optical processors. This discipline is still young and it is difficult, at this stage, to discern any useful trends and application directions.

B. APPLICATIONS OF OPTICS TO PROBLEMS IN SYMBOLIC COMPUTING

It is apparent from the discussions in the previous section that the important developments in optical processing have resulted primarily from three sources:

- (i) major technological innovations (such as with acoustooptic devices and optical bistability),
- (ii) new mathematical insights (such as in Fourier plane matched filter correlators and the framework of matrix algebra),
- (iii) novel architectural concepts (such as acoustooptic synthetic aperture radar/image processors and optical neural nets).

The proposal to use optical techniques in symbolic computing differs from all of the former examples in that the main driver in this case is the intended set of applications, such as expert systems, natural language understanding, and perception (visual and auditory). These novel applications may require a combination of innovations in technological, algorithmic and architectural aspects of optical computing that were discussed in the previous section, or they may require an entirely new set of concepts.

The current work in application of optical techniques to problems in symbolic computing is heavily influenced by existing approaches which were developed to attack the applications described above. The research in Artificial Intelligence (AI) has attempted to solve these problems over the last 20 years. Under DARPA's Strategic Computing program, the AI techniques developed in the laboratories are now being applied to real-world problems of military significance. These techniques have one common element in that they all make extensive use of a knowledge base that contains facts and rules for combining the facts about the specific problem domain. The critical factors that affect the performance of a symbolic computing system are the knowledge representation used, the size of the knowledge base, and the speed of the inference engine with which it is associated. Thus a new approach to symbolic computing that aims at offering improved performance must consider all of these factors.

In studying the use of optical techniques for symbolic computing, it is useful to look at the relation between symbolic and numerical computing. The manipulation of symbols typically involves some numerical computations. Thus the techniques developed for optical numerical computations can provide

a useful first step for symbolic computing. An important observation is that the precision requirements on these numerical computations are usually not very demanding. This is a consequence of the fact that although the numerical computations may be performed to determine the flow of control or data, the results of computation themselves are seldom propagated. Hence the problem of error propagation, that forces the numerical computations to use a large word length and floating point representations, is either nonexistent or severely reduced in symbolic computation. The relaxed accuracy requirements imply that optical systems using analog data representations may prove to be useful. A comprehensive discussion of the various considerations can be found in a book chapter by Neff and Kushner (7). At the level of mathematical formulation and architectures, however, symbolic computing is sufficiently different from numerical computing and signal/image processing, that new terminology, mathematical framework, and architectural concepts need to be adopted by the optical computing researchers. In the initial stages, the emphasis is on utilizing already existing tools and applying them to specific operations in symbolic computing where such a connection can be made. The developments in optical pattern recognition, acoustooptic processing and digital optics discussed in the earlier section can be modified for this application. This may not lead to the most effective use of optics in solving symbolic computing problems. The next stage is the identification of key concepts and operations in symbolic computing for which there is no direct analog in the conventional optical computing. This direction of research could lead to optical accelerators for existing digital electronic processors. The final stage is a complete reformulation of the problems of symbolic computing in order to make it more amenable to optical realizations. This stage aims at developing an entire system for a given set of applications.

The philosophy behind the research effort described in this report is to primarily focus on identifying new primitives in symbolic computing that do not have a direct analog in previous research in optical computing, and on laying the foundation for an entirely new formulation of symbolic computing in terms more suitable for optical implementations. The research in optical symbolic computing is still young enough that a direct comparison with digital electronic systems will be premature. Therefore the main thrust of this program will be on expanding the scope of the field and stimulate further thinking in new directions rather than to investigate specific narrow approaches with the hope that they will result in an optical system that is competitive with electronic systems.

C. OUTLINE OF THE TECHNICAL REPORT

In this section we will outline the contents of this technical report. This outline will serve to integrate the material contained in the individual chapters and put it in proper perspective with respect to the program philosophy. Each chapter addresses one specific aspect of symbolic computing. The selection of the particular aspects of symbolic computing was designed to cover different levels of complexity and different formulations of the basic goal of building knowledge-based systems. The implications of these different aspects of symbolic computing to optical implementation will be discussed in a subsequent technical report.

In the second chapter, we will discuss the operation of compare-and-exchange which is used frequently in relational algebra machines. This operation has not yet been studied for optical implementation and it is our belief that a fast and efficient optical implementation of it will have a significant impact on the eventual construction of optical relational databases that play a crucial role in symbolic computing.

Pattern matching is a very common elementary operation that forms the basis of sophisticated production systems such as OPS-5 or PROLOG machines. This operation can be viewed as a simple correlation or a vector inner product and then be implemented optically. Indeed this approach forms the basis of a number of proposals for optical symbolic computing. In chapter three we will discuss the pattern matching operation from a broader perspective and study some of the strategies that have been developed for efficient implementation of pattern matching.

Appendix A of this report contains a reprint of a paper that was presented at the Los Angeles Symposium of Society of Photooptical Instrumentation Engineers (SPIE). In that paper we describe several optical systems for performing the compare-and-exchange operation, and thus lay down a foundation for building the type of database systems discussed in chapter II of the report.

REFERENCES:

1. J. W. Goodman, J. Of Electrical and Electronic Eng., Australia, Vol.2, No.3, 1982.
2. Special Issue of Optical Engineering, Vol.23, 1984.
3. "Acoustooptic Signal Processing: Theory and Applications", N. Berg and J. N. Lee eds. Marcel-Dekker Pub. (1983).
4. R. A. Athale, "Optical Matrix Processing", in "Hybrid and Optical Computing", H. Szu ed., SPIE Publication No. (1987).
5. "Optical Bistability III", H. M. Gibbs, N. Peyghambarian eds.,.
6. J. W. Goodman, S. Y. Kung, F. J. Leonberger, and R. A. Athale, Proc. of IEEE, Vol.
7. J. A. Neff and B. G. Kushner, "Applications of Optics to Problems in Symbolic Computing", in "Optical Computing", R. A. Arrathoon ed., Marcel-Dekker Pub.

CHAPTER II

OPTICAL RELATIONAL ALGEBRA MACHINES

In recent years, the field of Artificial Intelligence has advanced significantly by developing practical applications in the form of Expert Systems, and has captured popular imagination in the process. Compared to other applications of computing, which manipulate numbers, the Artificial Intelligence applications concentrate mostly on the symbolic representation of knowledge. Therefore, these systems are also characterized as "Knowledge-based Systems." The main issues in knowledge-based systems are appropriate schemes of representing knowledge, transforming the knowledge into more useful forms, and adding to the knowledge base as a result of these transformations. Even in very simple applications, the size of the knowledge base is often huge, making the transformation procedure the performance limiting step. The requirement of maintaining the consistency of the knowledge base imposes additional constraints on the procedures employed for updating the knowledge base. In this Chapter, we will introduce the basic concepts in relational data bases and lay foundations for the use of optical techniques in relational database machines.

A. Background in Database Systems

Loosely speaking, a database is a collection of related information maintained in a machine readable form and accessible through a computer. Historically, databases grew from the use of data files. As more and more information was stored in machine readable form, and as individuals and organizations found increasing need to access that information repeatedly and rapidly, the data file became a database and specialized hardware and software were developed to easily access, retrieve, maintain, and update the information in the data files. Large databases are maintained on a secondary storage media such as tape or disk. As such, they require procedures that will access the information as efficiently as possible. Increasingly, large direct memory access systems are built to enhance the performance of database systems. The functions that every database system must provide include building, updating, retrieval, and data reduction.

Building is the process of adding records to the database. Most databases are organized in an alphabetical or arithmetic sequencing of one particular element, or access key, that is common to all records. The process

of inserting or adding a record to the database is designed to minimize the amount of reorganization of the database that is induced through this process. Interestingly, one result of the use of databases has been the assignment of an employee number for each employee hired to most companies. The use of sequential numbers insures that updating the employee database is always the process of appending new records to the end of the file. The file append operation is the least complex mechanism for adding information to a file.

Updating is the process of changing the information that is resident in a particular record of the database. When performing an update, a record is read from the file, its contents are changed, and the record is then returned to the file. Files which require frequent updating are organized so that the processes of reading and writing are performed with as little disturbance of adjoining records as possible.

Retrieving is the process of searching the file for particular records which contain information requested by the user of the database. The retrieval operation requires that the database be organized to optimize the searching process. This optimization is aided by knowledge of the types of retrieval requests that will be made by the users.

Data reduction is the process of refining, organizing, and computing statistics upon the retrieved information. External to the database and operating on information from the database, data reduction functions are part of a database system and distinguish database systems from file systems. Simple file systems do not include data reduction capabilities.

B. Relational Data Base Machines

Data base systems can be roughly categorized either as conventional or relational systems (1). Conventional data base systems are composed of records which are themselves collections of individual entities having various attributes. Related entities share a common valued attribute. In general, to determine related entities, the data base records are sorted according to some attribute, thereby localizing those items whose attributes are similar in value. Relational data base systems are composed of objects and relations among those objects. Relational systems are organized as sets of relations which themselves are organized as tuples of entities. To discover related objects, the relations are searched for those objects.

Access to relational data bases is performed through a query

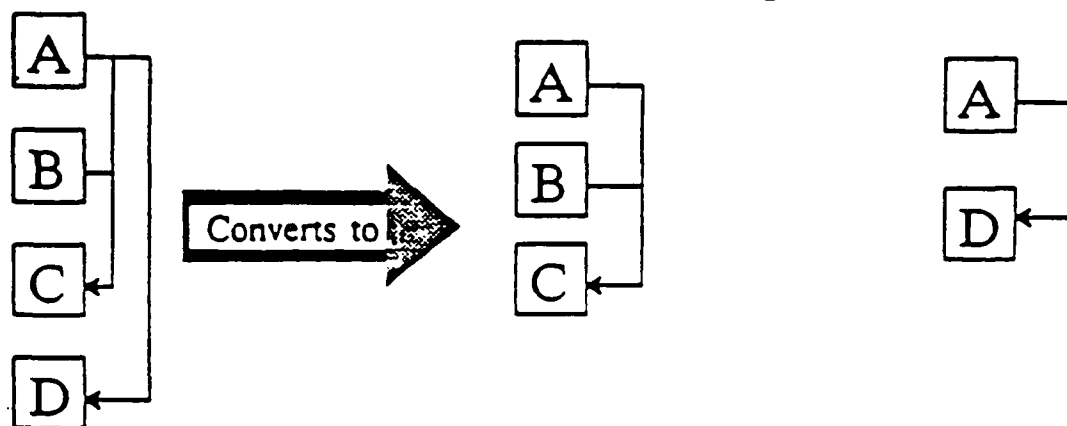
language. The two classes of relational query languages are algorithmic and predicate calculus based. The algorithmic languages allow one to express a query as the application of specialized operations to the relations of the data base. The predicate calculus languages allow the user to specify a query as a predicate that the tuples of some final relation must satisfy. In a predicate calculus language, tuple selection is generally achieved through the projection of intermediate relation tuples through the filter provided by the predicate.

In order to maintain the consistency and truth of a relational data base, it is necessary that the data base be in third normal form. Any data base may be transformed to third normal form by following a standard procedure. First normal form is achieved by insuring that all relations are in a tabular form in which each entry in a table represents one and only one data item, all items in any column are the same kind, each column is assigned a distinct name, all rows are distinct, and both rows and columns can be viewed in any sequence at any time without effecting either the information content or the semantics of any function using the table. Second normal form can be developed from first normal form data bases by insuring that each nonprime attribute of the second normal form relation is fully functionally dependent on each candidate key of the first normal form relation. Third normal form relations are formed from second normal form relations by insuring that every nonprime attribute of the third normal form relation is nontransitively dependent on each candidate key of the second normal form relation. Figure II-1 presents graphical analyses for the conversion process used to obtain third order normal form data bases. Again, the key reason for generating third normal form relational data bases is that third normal form data bases maintain their semantic integrity during the application of the selection, projection, join and division operations that are applied by predicate calculus or algorithmic systems to discover responses to user queries.

Expert system knowledge bases and rule bases have been developed as specialized data bases with their own specialized data base managers, update mechanisms and retrieval mechanisms. As expert system technology has matured, a realization has come that the capabilities available in advanced relational data base systems are applicable to the problems of maintenance and retrieval of knowledge from expert system knowledge and rule bases. The maintenance of data base consistency with third normal form data bases is akin to the requirement for truth consistency for knowledge bases. Efficient retrieval and update of relational data bases can be effected by specialized data base machines.

Conversion to Second Normal Form

Remove Nonfull Dependence



Conversion to Third Normal Form

Remove Transitive Dependence

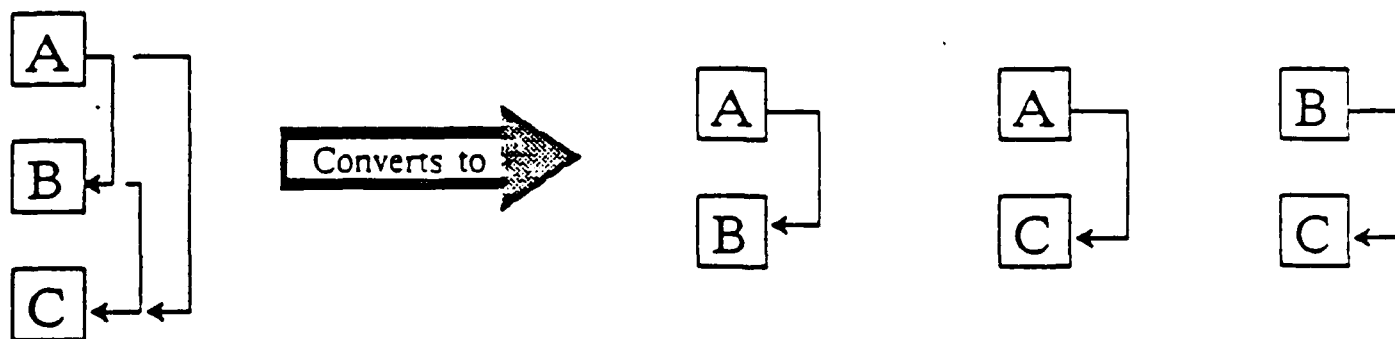


FIGURE II-1 Conversion

A variety of hardware structures have been developed for implementing any data base system. These structures can be divided into five classes. Figure II-2 diagrams these five approaches.

(1) In the Conventional Data Base Management Approach, the user is provided with a software system to which the application program is interfaced. All actions except simple file retrieval and mechanical disk seek operations are handled by software on the host system.

(2) Software Single Backend Approach is taken in a system which separates the application program on one piece of equipment from the equipment on which the data base manager runs. Examples of this type of system are those in which an intelligent terminal, such as an IBM-PC, is connected to a host running a standard data base manager. The intelligent terminal is configured with software to aid the user in developing queries which are translated to the specialized fixed-format data base queries necessary on the mainframe.

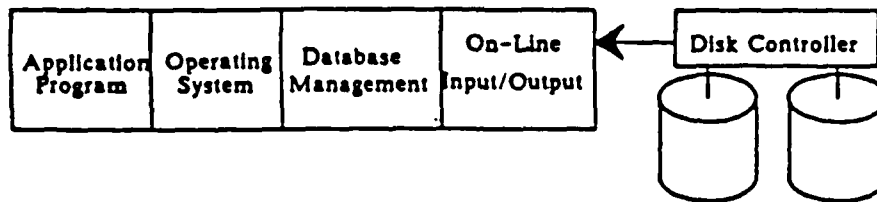
(3) The Software Multiple Backend Approach is taken in systems which are similar to the software single backend systems except that the backend must handle multiple controllers and intelligently determine the appropriate path that the user's queries should follow to optimize data retrieval from multiple sources.

(4) The Intelligent Controller Approach is an attempt to optimize the multiple backend controller problem with a smart disk system. The data base manager runs on the host machine with the application program.

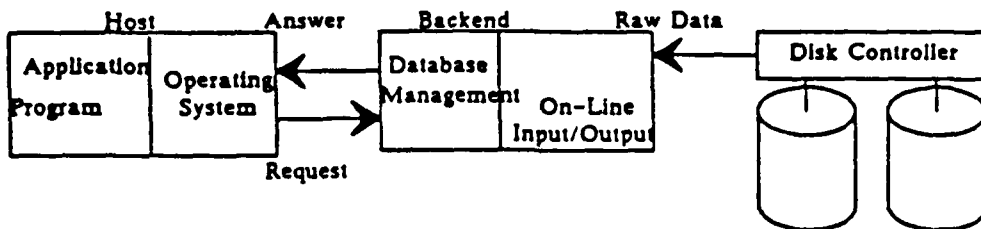
(5) The Hardware Backend Approach is an attempt to develop a specialized machine completely in control of the data base and all its associated functions. All operations on the data base are sent through the intelligent backend controller.

Recent work in electronic data base machines has shown that complex third normal form data bases can be converted to collections of binary relations that can themselves be represented as inverted horizontal structures (2,3,4). These inverted horizontal relations can then be processed in a modified Batcher's odd-even merge network configured as a hardware backend machine. The modified network provides the capability to implement an operation identified as generalized P-Selection.

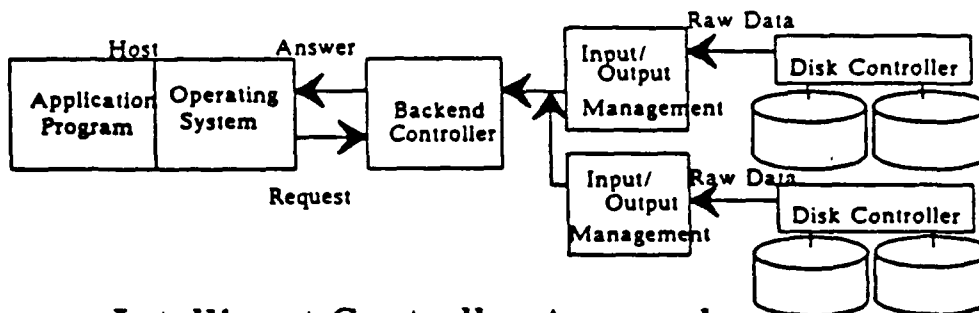
Conventional Database Approach



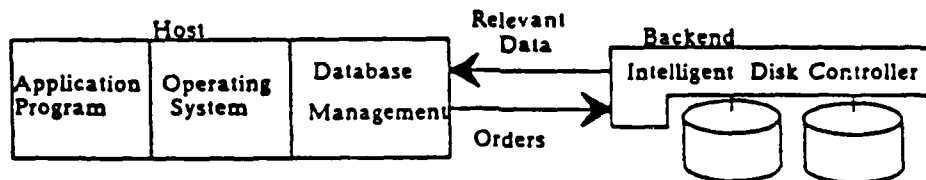
Software Single Backend Approach



Software Multiple Backend Approach



Intelligent Controller Approach



Hardware Backend Approach

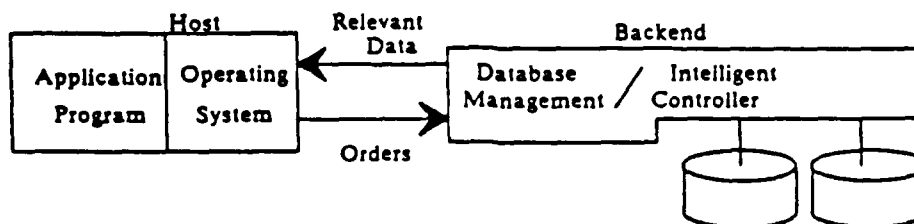


Figure II-2 Approaches to Database Machine Architecture

C. Generalized P Selection

A Batcher's odd-even network for sorting eight items is presented in Figure II-3. The figure contains an example of sorting a permutation of eight binary numbers. In every case, the algorithm executed in the switches is:

- (1) Accept two data items, one each from the top and bottom switch inputs.
- (2) Compare the two data items.
- (3) Send the high valued item out on the lower output and the low valued item out on the upper output.
- (4) Repeat indefinitely.

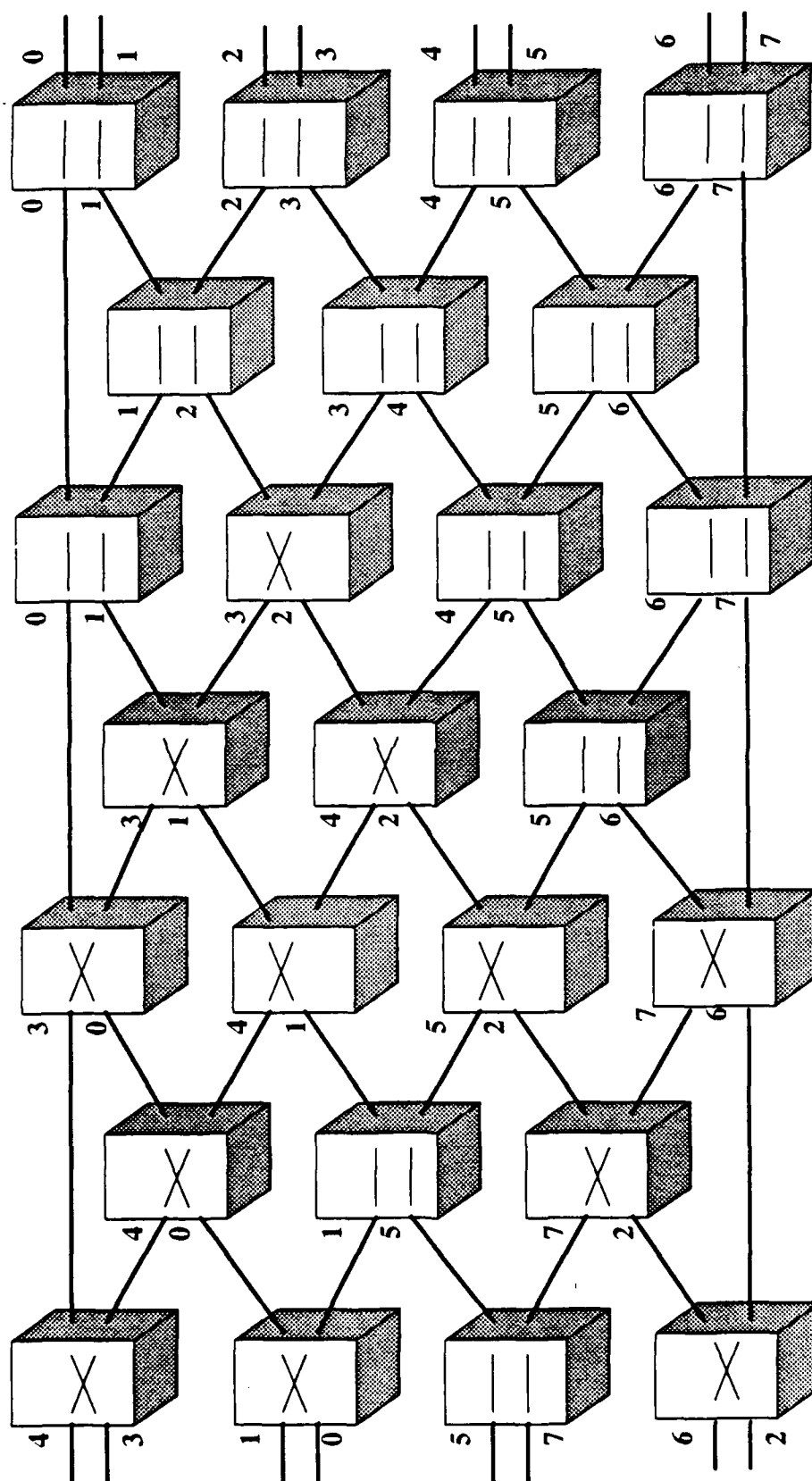
This algorithm will eventually result in all items being routed to their appropriate relative location at the output of the network. The amount of memory required at each switch to store two data elements for comparison may be arbitrarily large. Fortunately, binary data applied to the network in a bit serial, most significant bit first, fashion, can be sorted. Each switch will require a reset line which initializes the switches in a direct pass through mode and sets a latch element which records that the switch is in an "unselected" state. As soon as the inputs to the switch differ, the switch is latched to either pass data directly through or crossed depending upon whether the lower or upper input was "1," respectively. The switch continues to route the remainder of the input bits through to the output bits according to the state latched by the initial decision.

This sorting technique can be generalized to a selection technique. We define selection as the identification of like elements in two separate data sets. The bit serial sorting network requires the addition of a tag element on each data item. This tag bit is set for the selected elements. As demonstrated in Figure II-4, half of the network receives data from the database while the other half of the network receives a set of replicated items for selection. Whenever two like elements from different datasets (marked R and B) meet, the tag bits of the R elements are set. The selected elements are those with set tag bits.

The selection technique can be modified to provide a means for generating the union, intersection and difference between any two sets. These basic set operations provide the power to perform the relational database

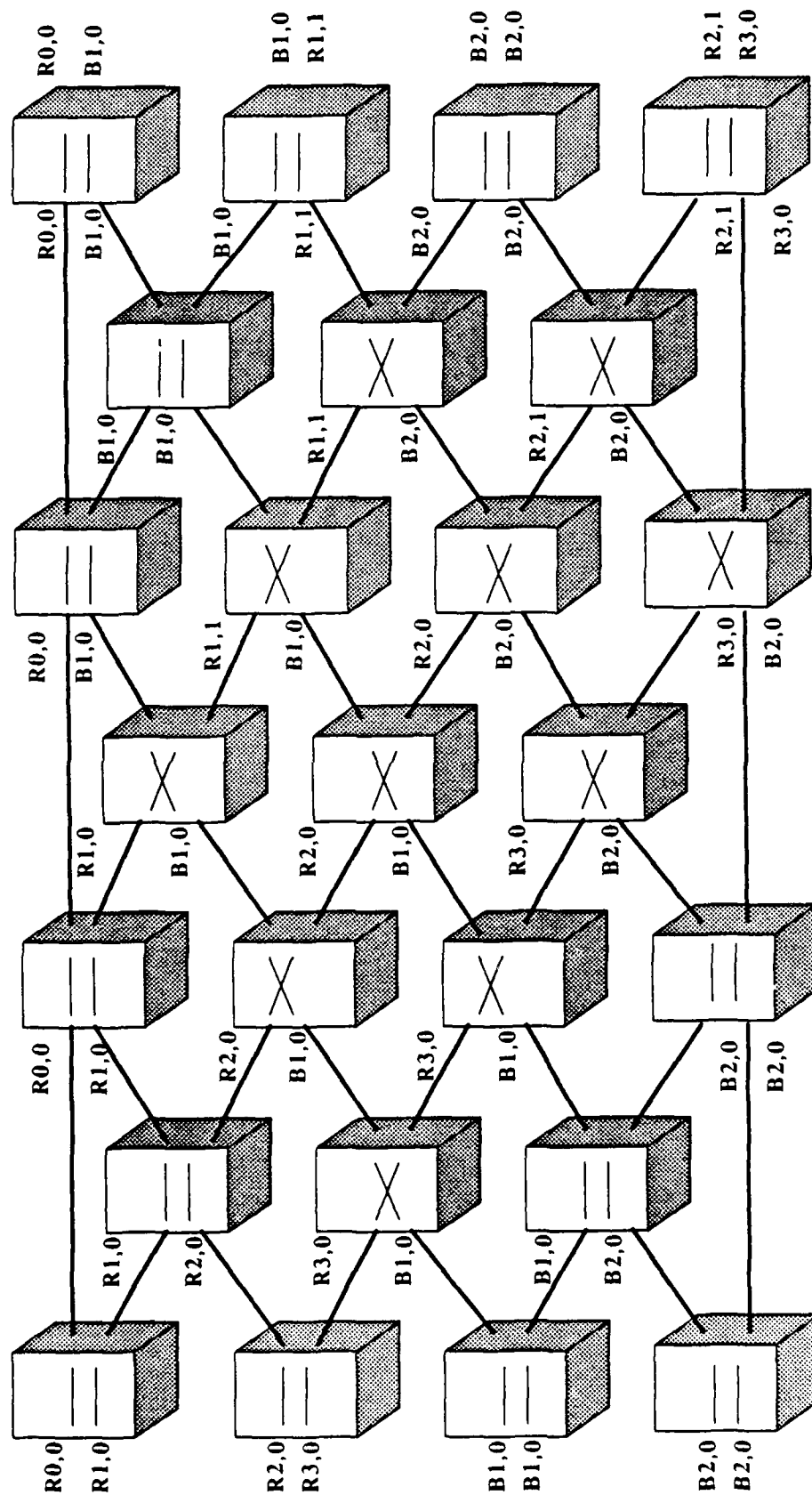
SORTING A PERMUTATION IN A BATCHER'S NETWORK

FIGURE II-3



SELECTING A 4 ELEMENT SET WITH ANOTHER 4 ELEMENT SET

FIGURE II-4



operations of selection, projection, join and division. The odd-even network is an order $\log N$ depth network with order $N \log N$ switches. As the number of processed tuples grows, a relational database machine designed around this scheme will require an extremely large interconnect network. Optical interconnects and optical computing elements are appropriate for use in this setting.

REFERENCES:

1. J. Martin, Computer Data Base Organization, Prentice Hall: England Cuffs, New Jersey 1977.
2. W. L. Shu, and A. K. Sood "Parallel Processor Implementation of Relational Algebra Operations," Vector and Parallel Processors in Computational Science II, Oxford, U.K., August 1984.
3. W. L. Shu, and A. K. Sood, "Hardware Implementation of Relational Algebra Operation," Proceedings of NATO-ASI on Relational Data Base Machine Architecture, les Arcs, France, July 1985.
4. W. L. Shu, and A. K. Sood, "Data Base System using Horizontally Separated and Inverted Data Structure," Preprint, 1981.

CHAPTER III

PATTERN MATCHING IN SYMBOLIC COMPUTING

In seeking innovative approaches to pattern matching, several starting points of departure might be pursued. The sophisticated and now classic pattern matching facilities for character strings in the formal language SNOBOL could be a basis. Lower level techniques involved in implementations of computer "hashing" functions on data arrays might be appropriate. Alternatively, some higher level pattern matching functionality might be wholly viewed – such as that involved in the matching operations in symbolic computing with respect to net or graph structured representations, or the explicit handling of sets of "production" rules.

The overall purpose of this effort is to provide a focus to subsequent considerations of the opportunities to apply optical computing capabilities to pattern matching – that is, give some guides to the necessary as well as generally useful properties for match operations. For this purpose, specific models for pattern matching were considered prejudicial to the intent. Specific implementations were seen as constraining the possible innovations that might be pursued. What is introduced here is a general functional decomposition of pattern matching that will actually emphasize the considerable variety of capabilities that might usefully be accommodated in implementations.

This report initially proposes a general decomposition of pattern matching that is motivated by the basic OPS – the most widely known and proliferated system for implementing rule-based "production." Some examples of other pattern matchers are then also discussed in relation to the decomposition – essentially confirming the general nature of the components. On this basis, the desired area of focus is established that will facilitate tradeoff considerations on aspects of implementation possibilities.

A. THE CASE OF INFERENCE ENGINE OPS

OPS is actually a family of systems that was developed at Carnegie-Mellon University (CMU) in the late 70's, transitioned into industrial versions in the early 80's, and was developed commercially into such current products as OPS/83, available from Production Systems Technologies, Pittsburgh. OPS has been widely covered by the popular trade literature, even to the level of describing the major components:

-Production memory (holding the "if-then" rules called the productions)

-The working memory (holding representations of the individual conditioning components that make up the left-hand sides of the productions)

-The interpreter (the underlying mechanism determining what productions are satisfied by the state of working memory, and controlling the overall execution of the system's "program").

Also widely reported is the cycle of activity conducted by the OPS controller: match, resolve conflict, and act. It's in the first phase of the process, the match step, that the left-hand sides of the productions are compared to the contents of the working memory. Matches are placed into the conflict set (so-called because multiple matches usually occur and require decisions of choice about the sequencing in the execution process). The matching operation, then, is the testing for relevance of productions against the existing problem state as represented by the working memory.

The basic data structure in OPS is a list – because the developers worked in the LISP programming environment that was prevalent in AI research then as now. An example of the pattern match, is given by OPS developer Charles Forgy (ref Forgy, AI Journal 19, pp 17-37, 1982). The basic object that exists in the working memory is a list of attribute-value pairs, headed by an identifier:

(Expression !Name Expr17 !Arg1 2 !Op * !Arg2 X)

Symbols beginning with "!" designate the attributes in the structure of the object. The attributes have an associated value that follows in the list. The above object "Expression" has the name Expr17, with value 2 for the first argument (!Arg1), an unspecified operator (!Op) * (ie, a wild card), and X as a second argument (!Arg2).

Variables are handled in OPS so that portions of the pattern can be left open in the specifications. The designation of a variable uses a symbol enclosed in the characters ">", as <VAL>. The utility of variables comes in essentially querying the elements of working memory for the contents of the specified attributes. The scheme allows multiple occurrences of a variable with the syntax that all the occurrences of that variable must agree with repetitions of some same value in the left-hand side of the candidate production.

Thus:

(Expression !Arg1 <VAL> !Arg2 <VAL>) would match either
(Expression !Name Expr4 !Arg1 aThing !Op * !Arg2 aThing) or
(Expression !Name Expr9 !Arg2 0 !Arg1 0 !Op *).

But is would not match:

(Expression !Name Expr8 !Arg1 0 !Op * !Arg2 otherThing).

B. THE PATTERN: SPECIFICATION AND ITS COMPLEXITIES

The pattern is actually a specification for matching. Substantial differences in pattern matching capabilities in various systems result from the constraints and flexibilities possible with allowed specifications. The above OPS example shows that the explicit indexing of attributes within the structure allows variations in the position and appearance of attribute-value pairs in expressions. The specifications in the pattern can also include predicates and algorithms for the "values" of the pattern. For instance, the Arg2 specification, instead of referencing <VAL> in the above example, might make a greater-than test, "(>aNumber)", or even involve some logical predicate.

Complexities beyond those of OPS have appeared in other pattern matching implementations – complexities such as functions and algorithms for operation on the values of candidate elements or determination of inter-element relations. These types of specifications may also be used in combination (e.g. evaluation of a functional form, using the values of the candidates' !Arg1 and !Arg2 for arguments, and comparison to the candidates' !Arg5 value).

The pattern's specifications determine how the pattern matching operates on the inputs – the values of the candidate. The pattern matcher is also characterized by its outputs – the information returned by the match operation. Since the pattern matching is intended to resolve details of the agreement between the pattern and the candidate object, the output needs to convey how the detail was resolved, e.g., what value of the candidate was "bound" for the specification's variable and what predicate was tested?

In addition, the output process needs to include the designation of the candidates successfully matched, or else provide some reference "pointer" to the matched structure. Whenever the specification excludes consideration of some of the attributes of the objects, the pattern matching "doesn't care" what values appear for those attributes in the candidates; however, the consumer for the match operation likely has vital interests in such attributes. The

implication of specification with partial explicit references is that the consumer is interested in all varieties of objects with respect to the "don't-cares."

These examples convey the gamut of pattern matching features that are relevant to symbolic representation and computing, and distinguish the input and output behavior of the pattern matching operation. In the following, several aspects of commonality with general data base access, query, and retrieval are pointed out, and that recognition leads up to introducing a generic model for the pattern matching process.

C. SIMILARITY WITH DATA BASE FUNCTION

Accessing and processing selective elements of data structures for pattern match determinations is functionally the same as the elementary data base operation. For instance, the directory handling functions of a disk operating system (say, MS/DOS) perform many of the "match" aspects described for the examples above (the directory is actually operated as a small data base operation).

A more highly structured example is the "record" data facility within a programming language such as Ada or Pascal. In such programming systems, the elementary data structures underlying the complex record data structure are manipulated by function modules making up the record facility (resident within the "shell" of the system). The practical advantage of the record structure for the programming environment is in treating a diverse collection of component structures (various component objects) as a whole. These capabilities typically include such whole-record operations as tests for equivalence or other forms of comparison – that is, pattern matching. In addition, the record user is provided means to command the elemental access operations of "get", "put", etc. that deal with the individual component attributes. Certainly a record data facility could be considered a basis for a pattern matching facility.

D. GENERIC MODEL FOR PATTERN MATCHING

The purpose in drawing the functional similarities of pattern matching with records and more general data base functionality is to support the identification of the generic decomposition of pattern matching as depicted in Figure III.1. The figure shows, on the left, pattern matching interacting with candidate data in terms of the underlying data structures. The interaction is depicted as data base operations that "serve" the necessary transactions be-

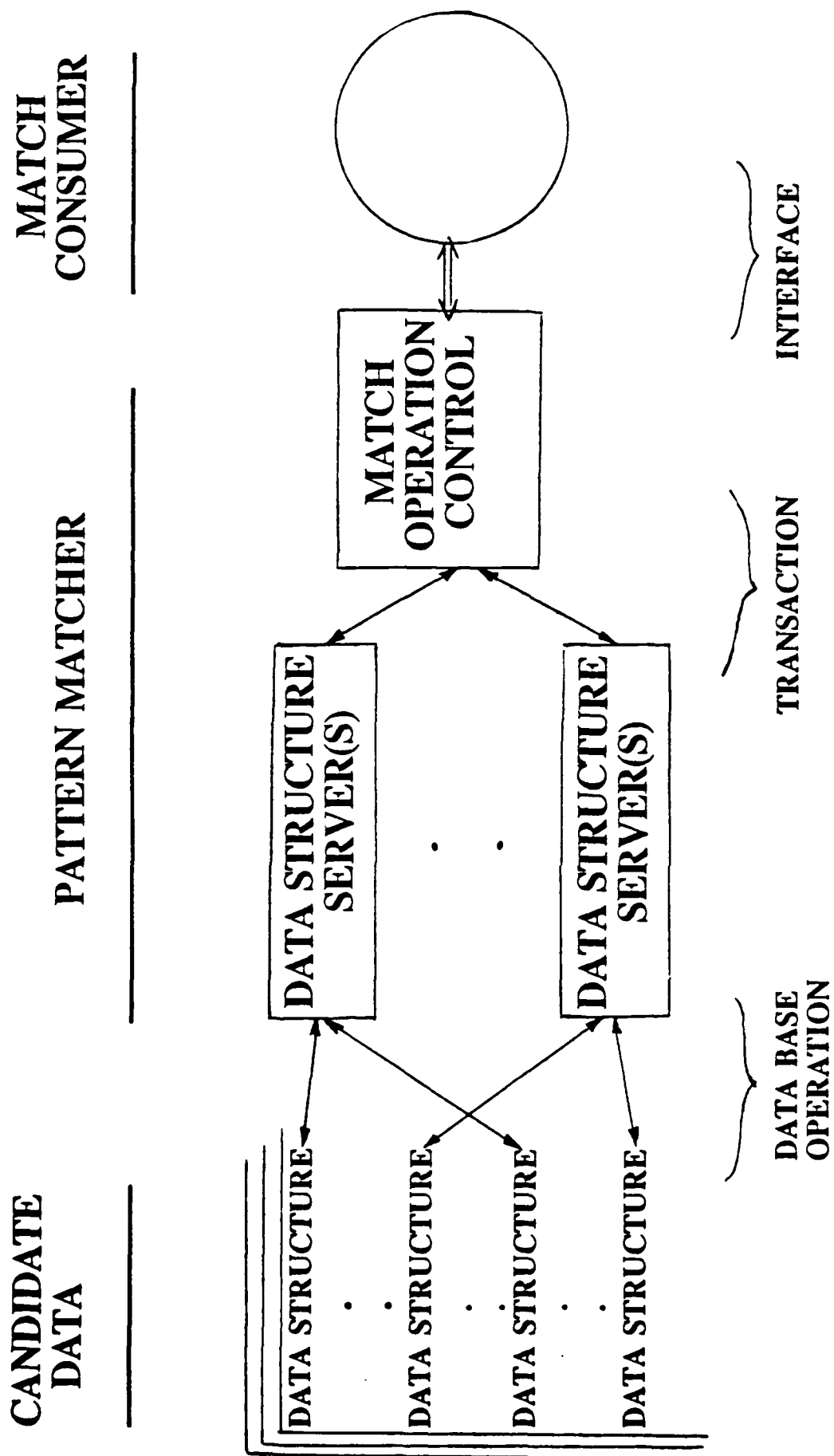


FIGURE III.1 PATTERN MATCH DECOMPOSITION

tween the matcher and the memory system for the patterns. Here, the "server" terminology is used in the sense of interactions between modules of a computer system, such as a processor being supported by disk or printer servers. The diagram indicates that data-serving to the pattern matching operation may be supported by multiple servers. This structure would be necessary in situations where the patterns of the system included components of greatly contrasting data structures, for instance, patterns having numerical, textual, and graphical elements among the attributes.

On the right of the diagram, the decomposition indicates that the match operation primarily involves conducting transactions with the data server or servers. Under this decomposition, the server transactions directly determine whether individual attributes of the candidate satisfy the specifications of the pattern. The transactions are essentially queries, providing access to the candidate's values on an attribute-by-attribute basis. Aspects of the match involving interactions with multiple attributes are conducted within the controller as a higher level operation; that is, the controller compiles the complete match assessment from the results of transactions with the data structure servers. In addition, the controller is responsible for an initial breakdown of the overall pattern specification into server transactions.

E EXAMPLES OF DATA TRANSACTIONS IN PATTERN MATCHING

Generally, the decomposition described above captures the characteristics of most disparate pattern matching facilities. Two particular examples provide a range of contrasts with the OPS case that was previously illustrated. One is from the domain of natural language for which R. Schank formulated a concept of "scripts" for representation of a line of discourse (ref, R. Schank and C. Riesbeck, "Inside Computer Understanding," Erlbaum, 1981). The other is a production system implementation system developed at CMU by D. Touretzky and G. Hinton for symbols and rules using distributed representations in a neural net (ref, D. Touretzky and G. Hinton, "Symbols Among the Neurons," Proceedings of IJCAA-85, Los Angeles, 1985).

Schank's scripts represent situational expectations in the form of a mini-story line, much as a formatted police report would stylize a particular incident. The patterns for matching in the deductions and inferences of the scripts approach are called "conceptual dependencies," and include a relatively extensive amount of information that details actors, actions, goals, and other subject references about objects or "directions." The pattern matching of scripts in language understanding provides an important example because

of the complexity of the objects matched and the need to extensively "bind" variables in the matching process so that details of the matched objects are retrieved. Early implementations of the pattern matcher for Scripts systems required relatively complex LISP programming by comparison to the match implementation in OPS. Special macro-codings in scripts programs were involved in the functions of handling the binding associations between variables and the respective components of the candidate - functionally that is the equivalent to data retrieval.

In simulations with objects and patterns of minimal complexity, Touretzky and Hinton devised a production system interpreter encompassing pattern matching and variable binding in a neural network architecture. Although distributed, the concept includes explicit components, using "relaxation" processes, for query and binding.

F. PARALLEL OPERATIONS IN PATTERN MATCHING

The generic decomposition between the serving of data structures and the control of the match operation also leads to consideration of yet other important aspects of pattern matching - the amount of exploitable parallelism. In addressing the exploitation of parallelism for rule-based systems, A. Gupta, C. Forgy and associates at CMU (ref. A. Gupta, C. Forgy, A. Newell, and R. Wedig, "Parallel Algorithms and Architectures for Rule-Based Systems", 13th Int. Symposium on Computer Architecture, Japan, 1986) looked specifically at parallelizing the match algorithm in OPS.

From estimates of match performance based on simulations, the CMU investigators not only obtained comparisons among alternative production systems, but also brought out generic factors affecting the amount of exploitable parallelism in the systems. One of their key results about parallelism is also of interest for the pattern match decomposition:

"Generally most of the affected productions require only a small amount of processing, while a few require much more processing. The few productions account for the bulk of the processing performed during the match, and hence the amount of exploitable parallelism is largely determined by these production. ...the obvious way to handle this problem is to divide the match process into many very small tasks."

In the decomposition, the division of the match process can be considered in terms of paralleled data structure servers. But the controller conducts activities that are serial to the overall match process. Primarily, the compilation role of the controller implies serialization because all the separate transactions with the candidate's data must be waited-out before resolving the match.

G. FOCUS ON DATA STRUCTURE SERVERS

The several aspects of the generic controller that have been discussed above – specification breakdown, interface with the consumer of the pattern matching, and compilation of elemental results from the likely paralleled array of servers – are NOT suggestive of major prospects for gaining enhancements in overall performance of pattern matching by innovation in control. These aspects actually imply that the control functionality is inherently serial to the overall matching operation in ways that would not be impacted by innovative implementation. On the other hand, the potential leverage gain by enhancing the individual components of a paralleled structure provides strong inducement for focusing attention on improving implementation of data structure servers.

The first level of the objective of this effort is now resolved by focusing on generic data structure servers for priority development. The remaining step to completing coverage of the objective is to provide a framework for consideration of enhancement alternatives. For this, the generic decomposition facilitates the understanding of the issue because of the emphasis on the commonality between functional capability of servers and data base systems. All prospects for innovative advances in data bases technology can be immediately considered opportunities for application to pattern matching.

H. ALTERNATIVES: DATA BASE ABSTRACTIONS

Data base technology at large currently provides mature design methodologies and implementations (ref G. Wiederhold, Database Design, McGraw Hill, 1983). Even with maturity, there remain varieties of systems for varieties of problems, and good reasons for matching capabilities of the systems to criteria determined by the objectives of the application. Wiederhold emphasizes that "the design of databases requires analysis for the prediction of performance, and this requires that the file organization can be easily "abstracted" (emphasis added). He relates suitability of a system to "fast access for retrieval, convenient update, and economy of storage." These principles

are all equally important for defining the desired "focal area" of capabilities for pattern matching developments.

The remainder of this report on pattern matching introduces an approach to "abstracting" alternatives for generic data structure servers. Here, the focal area is made more specific, but the outcome strongly adheres to maintaining an independence with respect to implementation. The concern about independence is to assure that the focal area makes sense without explicitly referencing such strongly typed data structures as lists (N.B. OPS evolved solely along the list orientation of the LISP environment in which it has existed).

Alternatives could reasonably range over the commonly known file and data base structures and organizations, such as the most simplistic linear and random access files, the more complex records that were discussed in examples above, or up to the sophisticated hierarchical and relational data base systems. However, that spectrum is considered too broad practically for these purposes. Intuitively, the focal area should be kept on the low side of such a spectrum - essentially on a par with arrays and lists, but giving consideration to greater structural variety.

It is proposed that Smalltalk objects be used in modeling the focal area of alternative concepts for data structure servers. The depiction in Figure III.2, from the description of the Smalltalk language (ref, A. Goldberg and D. Robson, "Smalltalk-80, the Language and its Implementation", Addison-Wesley, 1983), represents a simple taxonomy of varieties of objects. The representation of the taxonomy is that of a tree of relationships for data structures and associated behavior having highly descriptive names. The relations can be easily read from the figure; for instance, the "String of Characters" is an "Arrayed Collection" that is accessible by a "key" (the integer count of the character position in the string) as a consequence of an "external" (by a convention, not contained in the data instance) "ordering of the character elements".

The reason that Smalltalk structures provide a useful referenced taxonomy follows from the rigorous protocol approach that was pursued in definition of the Smalltalk system. Today, Smalltalk is an implemented programming system. However, in its development, Goldberg and associates at Xerox PARC achieved the implementation-independent definition of the language that enabled them to formally report the results in workshops to a significant cross-section of industry (with eventual publication of the book referenced

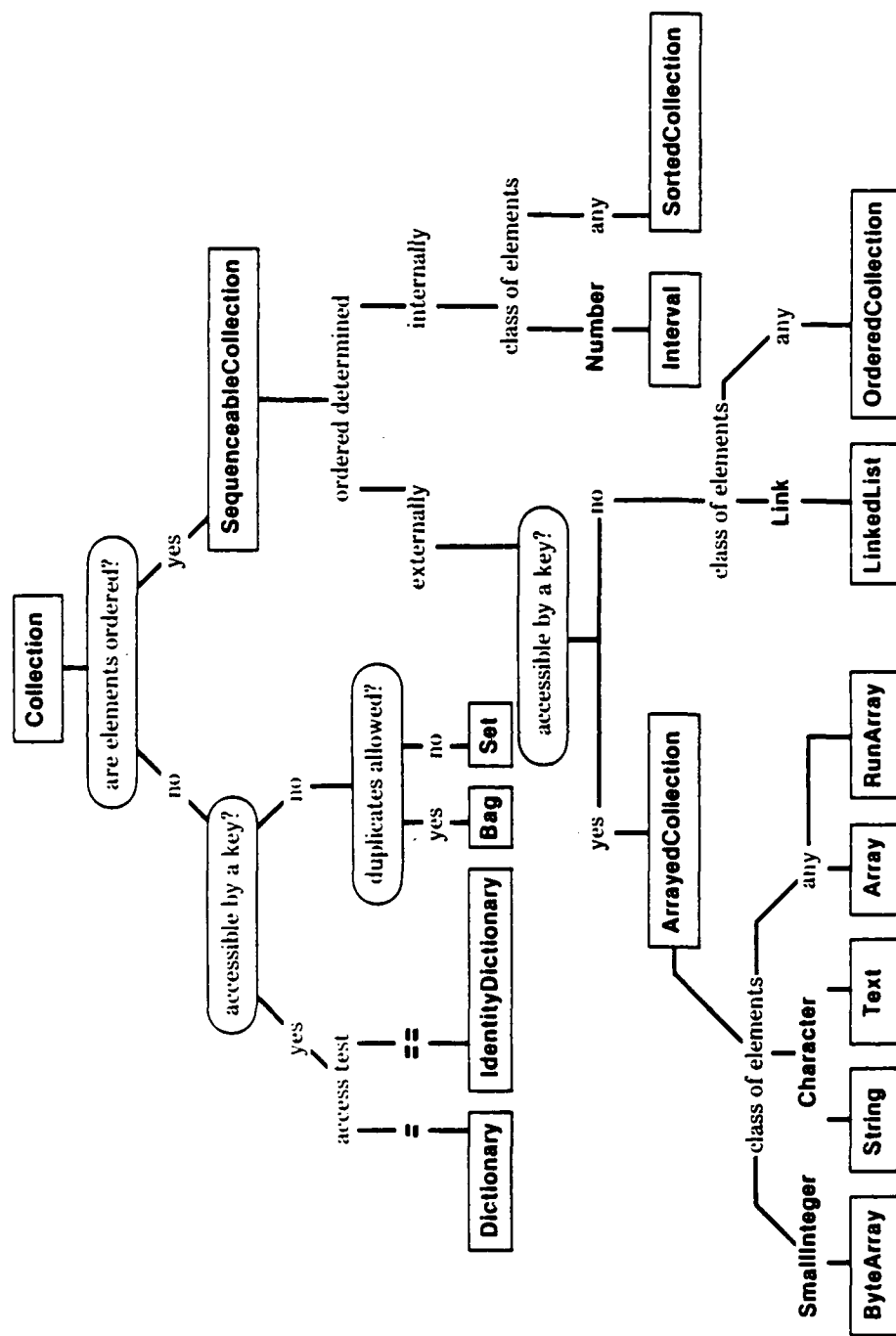


FIGURE III.2 TAXONOMY OF OBJECTS OF DATA COLLECTIONS

above). The resultant protocol-based definition of object-oriented structures ideally provides an appropriately abstracted focal area of server alternatives.

A sample of the protocols in Smalltalk is shown in Figure III.3, taken from the reference. The "Sequenceable Collection", appearing on the right-hand branch of the relations in Figure III.2, is a higher level construct that actually specializes to the list structure, among others. The details of the protocol in Figure III.3 (a partial showing of the object's total protocol) cite the basic requirements for serving this type of data structure. This protocol provides "for accessing, copying, and enumerating elements of a collection when it is known that there is an ordering associated with the elements. Since the elements of the collection are ordered, there is a well-defined first and last element. It is possible to obtain the index of a particular element and the index of the beginning of a sequence of elements within the collection (various "indexOf" terms)." These behaviors would be useful in pattern matching to determine data premises having anything to do with a subject term (e.g., a keyword). Alternative consequences can be obtained when locating a match to a component of the contents of the collection; and the protocol supports access to adjacent data.

Objects that are further specialized from the Sequenceable Collection inherit its protocols and add more, thereby elaborating on the parent structure. The String (left-side, bottom of the taxonomy) is specialized to collections of characters for elements and adds protocol for comparisons (in terms of ASCII alpha-character ordering). Hence, the appearance of ". aString" in a pattern would resolve candidates that collate after the argument, aString.

In these discussions of Smalltalk objects, it has been emphasized that the protocol of the object not only defines data structure but also the behavior of interactions with the instances of that structure. Thus, through the behavioral nature of the protocol, Smalltalk objects model the functionality for data structure servers in pattern matching. The control of pattern matching can be considered operating over the maximal range of pattern specifications achievable for the data structure. Consideration of the maximal range for patterns is desirable for placing maximal expectations on implementation alternatives, and essentially stipulates that a pattern specifications can be any program pertaining to the data structure.

I. CONCLUSIONS

Finally, the objective for this effort can be completed with the identi-

Class SequenceableCollection represents collections whose elements are ordered and are externally named by integer indices. SequenceableCollection is a subclass of Collection and provides the protocol for accessing, copying, and enumerating elements of a collection when it is known that there is an ordering associated with the elements. Since the elements are ordered, there is a well-defined first and last element of the collection. It is possible to ask the index of a particular element (*indexOf*) and the index of the beginning of a sequence of elements within the collection (*indexOfSubCollection:startingAt:*). All collections inherit messages from class Object for accessing indexed variables. As described in Chapter 6, these are *at*, *atPut*, and *size*. In addition, SequenceableCollections support putting an object at all positions named by the elements of a Collection (*atAllPut:*), and putting an object at all positions in the sequence (*atAllPut:*). Sequences of elements within the collection can be replaced by the elements of another collection (*replaceFrom:to:with:* and *replaceFrom:to:with:startingAt:*).

SequenceableCollection instance protocol

accessing	
<i>atAll:</i> aCollection put: anObject	Associate each element of the argument, aCollection (an Integer or other external key), with the second argument, anObject.
<i>atAllPut:</i> anObject	Put the argument, anObject, as every one of the receiver's elements.
<i>first</i>	Answer the first element of the receiver. Report an error if the receiver contains no elements.
<i>last</i>	Answer the last element of the receiver. Report an error if the receiver contains no elements.
<i>indexOf:</i> anElement	Answer the first index of the argument, anElement, within the receiver. If the receiver does not contain anElement, answer 0.
<i>indexOf:</i> anElement ifAbsent: exceptionBlock	Answer the first index of the argument, anElement, within the receiver. If the receiver does not contain anElement, answer the result of evaluating the argument, exceptionBlock.
<i>indexOfSubCollection:</i> aSubCollection startingAt: anIndex	If the elements of the argument, aSubCollection, appear, in order, in the receiver, then answer the index of the first element of the first such occurrence. If no such match is found, answer 0.
<i>indexOfSubCollection:</i> aSubCollection startingAt: anIndex ifAbsent: exceptionBlock	Answer the index of the receiver's first element, such that that element equals the first

element of the argument, aSubCollection, and the next elements equal the rest of the elements of aSubCollection. Begin the search of the receiver at the element whose index is the argument, anIndex. If no such match is found, answer the result of evaluating the argument, exceptionBlock.

replaceFrom: start to: stop with: replacementCollection

Associate each index between start and stop with the elements of the argument, replacementCollection. Answer the receiver. The number of elements in replacementCollection must equal stop-start+1.

replaceFrom: start to: stop with: replacementCollection startingAt: repStart

Associate each index between start and stop with the elements of the argument, replacementCollection, starting at the element of replacementCollection whose index is repStart. Answer the receiver. No range checks are performed, except if the receiver is the same as replacementCollection but repStart is not 1, then an error reporting that indices are out of range will occur.

Examples of using these accessing messages, using instances of String, are

expression	result
'aaaaaaaa' size	10
'aaaaaaaa' atAll: (2 to: 10 by: 2) put: \$b	'ababababab'
'aaaaaaaa' atAllPut: \$b	'bbbbbbbbbb'
'This string' first	\$T
'This string' last	\$g
'ABCDEFGHJKLMNOP' indexOf: \$F	6
'ABCDEFGHJKLMNOP' indexOf: \$M ifAbsent: [0]	13
'ABCDEFGHJKLMNOP' indexOf: \$Z ifAbsent: [0]	0
'The cow jumped' indexOfSubCollection: 'cow' startingAt: 1	5
'The cow jumped' replaceFrom: 5 to: 7 with: 'dog'	'The dog jumped'
'The cow jumped' replaceFrom: 5 to: 7 with: 'the spoon ran' startingAt: 5	'The spo jumped'

FIGURE III.3 PROTOCOL FOR SEQUENCEABLE COLLECTIONS

fication of alternatives in terms of the simple taxonomy of data structure servers in Figure III.2. Each alternative data structure server is a direct basis for an alternative pattern matching capability. The linkage between the generic decomposition portrayed in Figure III.1 and Smalltalk objects is a rigorous characterization with considerable stimulation for spawning innovative implementation alternatives. Smalltalk objects provided the desired independence with respect to implementation, yet identify substantive data structure alternatives and a robust characterization of the requisite server functionality.

The course of this effort has been an investigation of various pattern matching circumstances and aspects of complexity as a basis for considering implementation alternatives. There may be some tendency to look toward decomposition of complex pattern matching into more elementary match operations. This investigation has avoided a singular focus on identification of a common denominator in elemental pattern matching in order to provide a basis for dealing with efficiency of an overall application. The reason has been that a priority emphasis on effectiveness of an elementary operation may adversely complicate the results of imbedding the element in the application. Pattern matching alternatives should enable trading off design factors of the elements in a process decomposition. The taxonomy that is presented here is intended to keep open the perspective of practical, higher level pattern matching and support critical functional analysis of tradeoffs.

CHAPTER IV

SUMMARY

Optical computing has evolved significantly over the last 25 years in terms of the technology and applications. It had its beginning as a purely analog and dedicated signal and image processing system based on increasingly sophisticated technology; from film and lenses to acoustooptic Bragg cells, spatial light modulators, and computer generated holograms. Recently the discovery of high speed optical devices performing the binary logic operations that are fundamental to the construction of a digital machine have led to novel research directions in digital optical architectures. The applications of optical techniques to problems in symbolic computing represents a new application direction. The research described in this technical report was aimed at studying the nature of symbolic computing operations and then transitioning this understanding to optical implementations. It was discovered during the course of this research that there were some primitive operations that played a central role in two important knowledge based systems: relational databases and production systems.

High performance relational database machines often used special purpose processors to handle user queries in an effective manner by performing some common relational algebra operations, such as projection, join, selection. One common approach to parallelizing these operations is to employ multistage networks of modules performing "compare-and-exchange" (C&E) operation. This architecture was identified as a key element in building optical accelerators for relational algebra machines. Since optical components can be configured to build the type of regular interstage communication networks which involves global connections, the research effort was focussed on designing optical realizations of the active units of this architectures- the C&E unit. Several designs based on digital optical, hybrid optoelectronic, and polarizaton devices were developed. The details can be found in the reprint in Appendix A. Future research in this area will concentrate on optimizing digital optical circuits based on bistable optical logic devices for performing the C&E operations and studying different network architectures from optical implementation consideration.

Pattern matching is a common operation to a variety of production systems that form the heart of the inference engines. The exact nature of the pattern matching operation is governed largely by the data representation used. Although optical techniques have been widely used to perform pattern

recognition on images and correlations on time signals, that experience cannot be directly applied to the pattern matching operations of symbolic computing. We have initiated a task on analyzing the nature of various pattern matching operations and identifying different components of the operation. This in turn will lead to opportunities for optics to make an impact in giving superior performance. Chapter III in the report described the framework for analyzing the pattern matching operation. In the coming nine months this framework will be utilized in defining the role optics can play in this critical operation.

In summary, the report identified compare-and-exchange and pattern matching operation as the two critical building blocks for any future optical systems for processing symbolic data. Several optical designs for implementing compare-and-exchange operation were formulated and a strategy was developed for investigating the role of optics in pattern matching operation.

Optical implementation of the compare-and-exchange operation for applications in symbolic computing

G. W. Stirk, R. A. Athale, and G. B. Friedlander
The BDM Corporation
7915 Jones Branch Drive, McLean, Virginia 22102

Abstract

The throughput of data-structure manipulation operations presently limits the applicability of relational database machines. Since most relational algebra operations can be treated as modifications of sorting algorithms, special-purpose hardware based on fast sorting algorithms should increase the performance of these machines. Parallel sorting algorithms representable as self-routing, multistage networks are ideal for optical implementation because they require global interconnects and simple parallel-processing units. The processing units perform a local operation called compare-and-exchange (C&E). Our goal is to realize fast optical sorting networks. Therefore, we describe C&E implementations in analog optics, and digital optics with all-optical, hybrid optoelectronic and polarization logic. Furthermore, we delineate application domains of the networks based on system and technology characteristics.

Introduction

The application of many artificial intelligence (AI) techniques to interactive environments like process control depends on the throughput of real-time relational database machines. Currently, two throughput bottlenecks limit AI applicability to problems with small knowledge bases: the time to extract specific data increases with the size of the knowledge base, and the temporal length of relational algebra operations increases with the amount of data extracted. Techniques to improve the former by parallel architectural organization and operation have been successful.¹ These techniques in conjunction with high-speed processing of the extracted information provide the key to extending the size of real-time relational databases. With past experience in mind, we focus our attention on optical accelerators for large-kernal, relational-algebra operations to increase system throughput.

The most common relational database operations include special-purpose operations like selection, projection, division and join along with the logical set operations of intersection, union, difference and Cartesian product.² In general any fast sorting algorithm serves as a basis to form the logical set³ and most of the special-purpose operations.⁴ All procedures that sort in sublinear time--less than $O(N^2)$ --require global communication between SIMD parallel processing nodes--properties inherent to many optical computing architectures. Parallel sorting algorithms--at least those representable to first order as fixed, multistage networks of processing elements--are ideal for implementation on optical architectures because they require sparse, space-variant interconnects with low fan-in/out. Moreover, the simple processing elements can be built using high-speed optical devices to further enhance the performance of network architectures. Hence, in this paper we explore the optical implementation of multistage network sorting algorithms.

Sorting with self-routing, multistage networks

Relational-algebra operations manipulate structures rather than individual data. In general, the ultimate disposition of each datum in the structure depends on its relative value and position with respect to the rest of the data. Self-routing multistage networks implement a class of parallel divide-and-conquer algorithms, where the final position of each datum is calculated using parallel local position computations within each stage along with sparse, global interconnects between stages. The local computation, compare-and-exchange (C&E), is a generic local-position calculation and routing algorithm for multistage networks that can be modified, through the addition of a simple algorithm that reads and manipulates message tags, to perform most relational-algebra operations.

The self-routing algorithm for sorting consists of two separate operations: comparison, where calculation of an exchange control signal occurs; and exchange, where the data switching operation takes place. The calculation of the exchange control signal in the comparison module is a nonlinear, numerical operation. In particular, the comparison operation decides which input to the switching module is larger and generates a control signal for the exchange unit which maps the inputs to appropriately coded output channels (fig 1). The C&E module represents a 2×2 crossbar under distributed control, where the locally routed information completely determines the configuration of the switch, in contrast to centralized control, where information external to the switch determines its configuration. Self-routing is preferred even though the best of both classes of algorithms take $O((\log_2 N)^2)$ time to sort the data,^{5,6} because central control takes an additional $O(\log_2 N)$ time to direct the data to the calculated destinations. Besides the additional temporal complexity, central control increases the spatial complexity of the network by adding processors and connections. Thus the constants and lower order terms in the expressions for temporal and spatial complexity imply that self-routing is the least costly solution.

$$\begin{aligned}
 \text{COMPARISON: } E(A, B) &= 1 \text{ FOR } B > A \\
 &= 0 \text{ FOR } A > B \\
 \text{EXCHANGE: } S(A(H), B(L), E) &= (A(L), B(H)) \quad \text{FOR } E = 1 \\
 &= (A(H), B(L)) \quad \text{FOR } E = 0
 \end{aligned}$$

Figure 1. Compare-and-exchange equations

In order to passively rearrange switched data within each stage, the previous stage of the network must tag the data as "High" or "Low" reflecting the outcome of the exchange unit. Since most active optical devices operate in an intensity mode, data is encoded permanently as intensity. The remaining degrees of freedom available to encode the transitory high/low tag are wavelength, polarization, phase or spatial position. As we will discover in the next section, the encoding strategy chosen for the tag will restrict the architectures and technologies available for C&E.

There are many networks that sort using self-routing algorithms. The odd-even, bitonic and perfect-shuffle were originally proposed^{5,9} to sort $N=2^n$ items distributively in $O((\log_2 N)^2)$ time; it was later shown that a large class of networks can simulate the shuffle-exchange in $O((\log_2 N)^2)$ time, and hence, sort by self-routing in the same order of time.^{6,7} Still other networks provide lower sorting times for specific values of N .¹⁰ An optical architecture for the perfect shuffle has been demonstrated.¹¹ The optimal network architecture for a sorting application depends on many parameters including but not limited to the number of inputs, the characteristics of available switching and routing technology, the network characteristics, systems engineering and the application requirements. Since a detailed trade-off analysis of all these issues cannot be made without more theoretical and experimental evidence, we limit ourselves to a discussion based on available information.

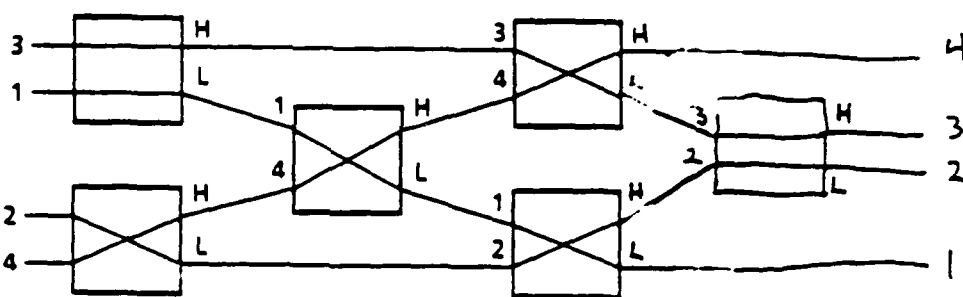


Figure 2. Sorting with compare-and-exchange on a multistage network

The operation of a 4x4 multistage network of C&E modules is shown in Figure 2. The data values are noted on the connection lines between modules. Since optical implementations of multistage networks have been proposed,^{11,12,13,14} in the next section we focus our attention on the active switching component of optical sorting networks, the C&E modules.

Analog implementations

One approach to analog comparison requires two stages of computation: the first stage subtracts one data element from the other, the second stage performs a nonlinear sign detection on the result to produce the exchange control signal. A possible implementation of the comparison operation using this strategy calls for coherent subtraction of amplitude encoded data and interferometric phase demodulation for sign detection. An incoherent implementation consists of the addition of two biased beams, one contrast reversed, followed by thresholding at the bias for sign detection.

Some open issues remain concerning any analog implementation of the C&E operation. In error-free sorting the finite accuracy of the comparison calculation directly limits the maximum dynamic range of the inputs. In addition, signal degradation by sampling for comparison, crosstalk noise, and attenuation in exchange are serious problems. Since signal regeneration is not inherent to analog exchange, the SNR decreases with successive stages. The signal-to-noise ratio at the input to the last comparison operation further limits the dynamic range of the inputs, and ultimately limits the stage cascadability. Since the number of stages is the logarithm squared of the number of inputs, and symbolic computing operations are of indeterminate size and precision, the domain of applicability of analog multistage network computations is

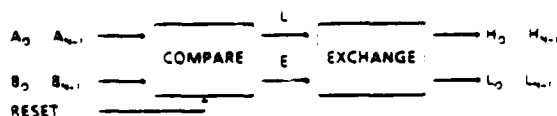
symbolic computing will be severely limited. Because of the problems with analog approaches, we concentrate on digital implementations of C&E.

Digital implementations

The advantages of digital implementations of C&E are many. Often, digital logic regenerates signal levels. Hence, C&E units are infinitely cascable. If crosstalk noise is small and constant with N , infinite cascability implies that the number of inputs is limited only by physical constraints. Moreover, the complexity of digital C&E modules remains constant regardless of network size and dynamic range of the inputs, contrasting with the network size dependent dynamic range requirements of analog C&E modules. The advantages of the digital approach to multistage networks for symbolic computing are compelling enough to warrant a detailed investigation of implementations of C&E with optical logic.

In a digital C&E module the input word mismatch occurring closest to the most significant bit determines the switch configuration. Hence, digital comparison for sorting is inherently bit serial and the data is represented as binary temporal strings. A description of the algorithm for digital C&E is as follows:

- 1) insert the bit streams according to the interstage connections into pairs of channels consistently labeled high/low;
- 2) at the first occurrence of a mismatch between the strings, a latching signal is set to one;
- 3) if the high channel contains the larger datum, place the switch in the barred configuration, otherwise the second channel contains the larger datum and place the switch in the crossed configuration;
- 4) at the end of each string reset the latch to zero (Fig 3). Reset control forces the network to operate synchronously. The time evolution of the switch position is illustrated in Figure 4 for typical input streams.



- AT THE FIRST OCCURRENCE OF A $\neq$ B, L IS SET TO 1
 - IF A > B THEN A > B, DON'T EXCHANGE E=0
 - IF A < B THEN A < B, EXCHANGE E=1
- RESET L=0 AT THE END OF EACH WORD

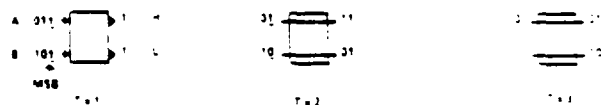


Figure 3. Digital compare-and-exchange rules

Figure 4. Time evolution of digital compare-and-exchange

Several strategies exist for designing digital optical C&E modules: the most straightforward borrow well-developed digital electronic techniques. Direct implementation techniques begin by forming a truth table for the module that explicitly represents the mapping between all possible input and output states. Logical minimization then reduces the truth table into a set of equations whose primitive operations form a logically complete set. The designer then directly implements the equations in a technological substrate that supports the appropriate logic primitives. The primary advantage of a direct implementation is speed because minimized equations possess a constant temporal depth of two to four. However, direct implementation is only useful when the number and size of the terms in the equations are small; otherwise the space-bandwidth product of the implementation becomes impracticably large and one must resort to algorithmic approaches, where the problem is subdivided into more tractable units. Algorithmic approaches are undesirable for two reasons: there is no generic recipe for algorithm construction, and algorithmic temporal and spatial complexity usually depend on the size of the problem.

The truth table of the C&E module is small enough to imply that the direct approach may be feasible. However, as illustrated in the set of equations and circuit derived for the NAND synthesis of C&E in Figure 5, the interconnect and processor spatial complexity of the compare operation mandate a costly, direct optical implementation because optical logic devices and space-variant connections are expensive. But special-purpose primitive operations reduce the gate number and interconnect spatial complexity of the direct implementation. For example, latching logic significantly reduces the spatial complexity of the compare operation as shown in Figure 5. Latching implies that some form of memory is present in the logic gates eliminating the need for feedback or flip-flops. When the most significant mismatch is found in the bit streams, one of the first-layer logic gates latches into state true while the other remains unlatched at

false. The second layer gate must also latch to prevent later bit comparisons from altering the exchange setting. Once set, the exchange unit remains latched into the appropriate crossed or barred position until the latching gates are reset. Other latching logic families, like those based on latching inverters or NOR gates, may form alternative comparison circuits. However, the transformation rules of Boolean algebra, like DeMorgan's theorem and logical minimization, in general do not apply to latching logic equations. Therefore, to form circuits based on other latching logic families, rules must be found to transform the equations or functional circuits must be constructed by inspection or trial-and-error.

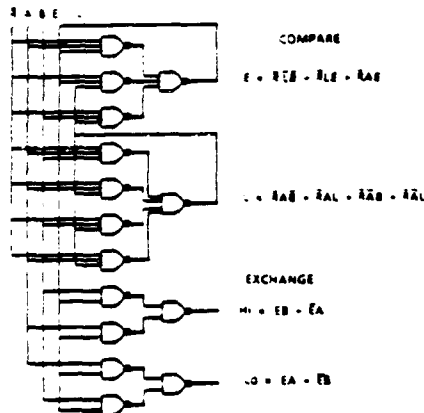


Figure 5. NAND synthesis of compare-and-exchange

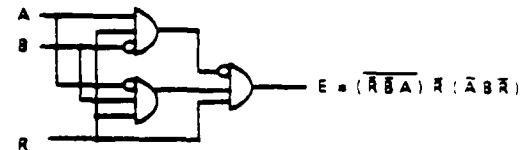


Figure 6. Latching logic implementation of the compare operation

Latching gates are possible using a variety of optical logic technologies. In particular, bistable Fabry-Perot etalons are ideal candidates to implement latching AND gates for the compare operation because of their high speed.⁴ The latching circuit tolerates their low gain because the fan-out of the latching gates is one. Any device having a decreasing-sigmoidal transfer function serves as an inverter, i.e., a properly biased spatial light modulator (SLM), an etalon in reflection mode, or a SEED¹ (Fig 7). Because of their speed, comparison units based on bistable Fabry-Perot etalons are well suited for high time-bandwidth product signals.

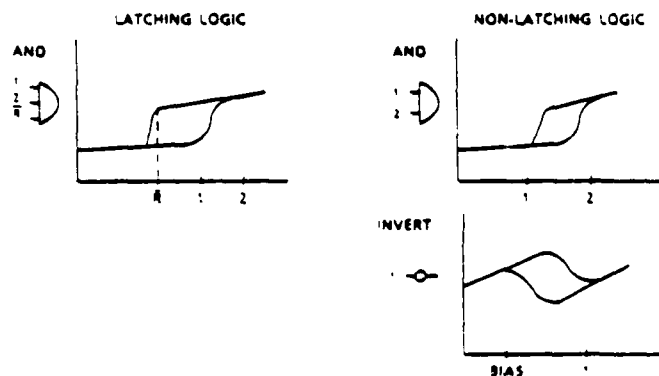


Figure 7. Logic transfer functions

Spatial-position-encoded exchange units built with all-optical logic regenerate signal levels. Thus, only signal to noise, crosstalk, uniformity and other systems engineering considerations limit the number of channels per stage and the total number of stages. Hence, all-optical exchange applies to deep networks with many high-speed inputs. As shown in Figure 7, any device with a properly biased sigmoidal-increasing transfer function serves as a standard AND gate. Since the OR'ed signals are mutually exclusive, any passive combiner functions as an OR gate. If the comparison unit described above is used in conjunction with this exchange module, the last latching AND gate in the comparison unit along with the high and low exchange outputs must have a gain of at least two to drive the next unit's input logic. In addition, the high speed of the devices indicates that the module latency will be governed by intra-module interconnect times. Module architectures with lenses for space-invariant connections as shown instead of space-variant imaging with holograms--will reduce the interconnect times and increase the allowable space-bandwidth product. Such architectures allow 4-f imaging systems that collect scattered and diffracted light to prevent degradation of the signal-to-noise ratio with increasing N . Furthermore, the inter-stage, space-

variant connection time determines the overall sorting system latency, but the computation can be pipelined for high throughput at the expense of increased spatial and control complexity. Somewhat slower active devices, like the SEED, can be used for exchange, but they may increase the system latency and reduce the throughput. In any case, the information on each channel following a latching operation may have a higher bandwidth than the comparison logic to increase throughput, but the bit rate must be slower than the exchange logic.

The potential throughput is greatly increased if the exchange module uses passive switches that allow trailing information to propagate at optical media time-bandwidths. Polarization encoded switching using Wollaston prisms and controllable half-wave plates¹¹ is one technology that performs passive routing. A photoactivated, polarization encoded exchange unit is shown in Figure 8. A photodiode, photoconductor or phototransistor receives the exchange signal and produces an electric field dependent change in the polarization of the dynamic half-wave plate through the electrooptic effect. When activated, the dynamic half-wave plate rotates the polarization of the orthogonally polarized signal beams through 90°; thereby acting as a passive switch. Subsequently, the Wollaston prism or polarizing beamsplitter separates the high and low channels. The advantage of polarization switching, in addition to its passive nature, is that exchange occurs in one stage and the data may occupy the same spatial channel. Moreover, the fan-out of the previous comparison module need only be one. However, the frame rate of optically controlled, dynamic half-wave device arrays is presently constrained to the millisecond regime by combined optical and electrical switching power dissipation limitations. With such new materials as ferroelectric liquid crystals, the response time can be pushed to microseconds using a photoconductor or photodiode, or possibly nanoseconds using phototransistors and low density arrays. Since exchange based on polarization-tagging is non-regenerative, the cascability and the number of inputs is primarily limited by absorptive, diffractive, sampling and scattering losses. A polarization-preserving SLM or etalon will regenerate each channel at the expense of reduced time-bandwidth product. For all these reasons, polarization coded exchange applies to small networks with long, ultra-high time-bandwidth product packets or large networks with slower signals.

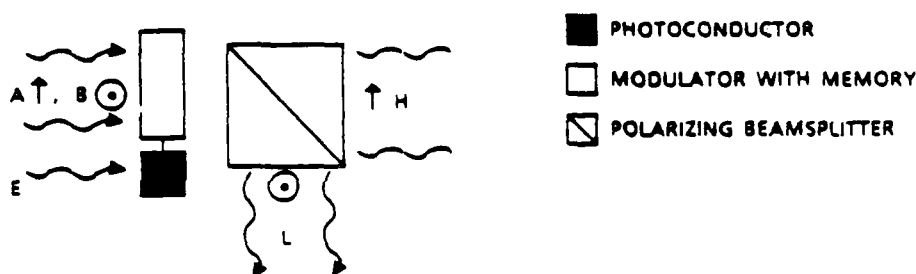


Figure 8. Polarization encoded exchange

There are other comparison technologies compatible with polarization exchange besides all-optical logic. In particular, electro-optic latching logic is well suited for the comparison operation. A two-stage subsystem of detectors and modulators is shown in Figure 9. The modulators are nonlinear and normally transmitting; a signal from a detector converts the corresponding modulator to opaque. For correct operation, the second stage of comparison logic must latch to opaque, thus blocking the exchange signal when A is greater than B, and the subsequent exchange unit must latch to the crossed position when the exchange signal indicates that B is greater than A. Hence, a reset signal is needed at both these points to process subsequent messages. Latching electrooptic devices could use PLZT or the lower switching energy ferroelectric liquid crystals as the bistable material. Because of the frame rate limitations of electrooptic device arrays, the bandwidth of the message headers is limited.

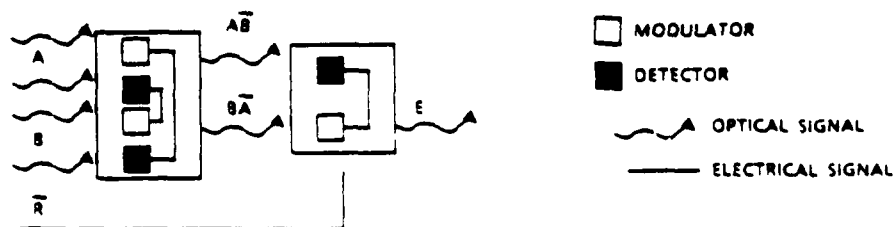


Figure 9. Comparison with electro-optic latching logic

Hybrid optoelectronic systems are possible where optics performs communication functions and electronic circuits compute the C&E operation. For regenerative logic, the active optical devices can be three-port modulators,¹⁷ light emitting diodes or laser diodes. High performance circuits for comparison are expected to have response times limited by the GaAs electronics to a few GHz.¹⁸ The hybrid technology is potentially powerful, but still in its infancy: therefore, it is difficult to predict relevant domains of applicability.

An alternative approach employs an electronic circuit for comparison and integrated optical circuitry for exchange. Delays must be introduced in the fibers so that the result of the comparison operation reaches the IOC before the data. However, the sampling losses are considerable, so without optical amplifiers to regenerate signal levels the size of the network is limited. Optical amplifiers have noise problems but can be expected to improve dramatically considering the amount of research conducted in this area. An integrated optics approach related to the three-port hybrid architecture is shown in Figure 10. One electronically controlled switch is used to regenerate the signal and a subsequent switch routes the signal to the proper output. Thus, the size of the network is limited by physical constraints and the speed of the electronic circuit governs the delay. However, electronic and integrated optic circuit coupling is an immature technology.

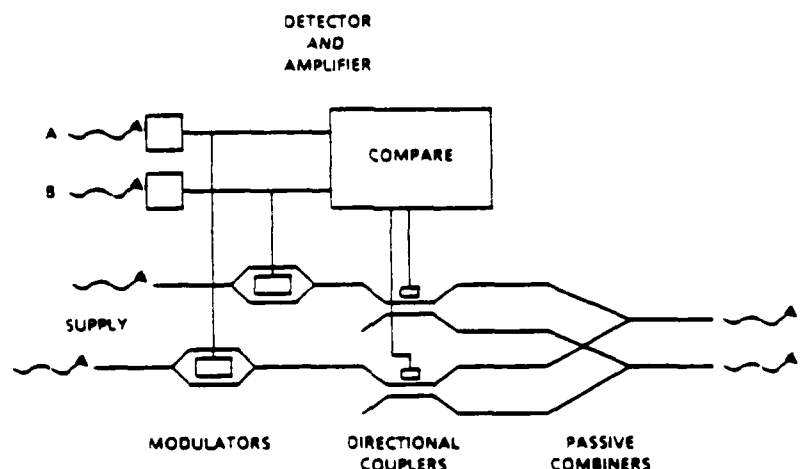


Figure 10. Electronics for comparison, integrated optics for exchange

Conclusions

The compare & exchange operation can be implemented with a variety of optical technologies. The optimal technology depends on the requirements of the application of interest. Unfortunately, we do not yet have reliable figures for relational-algebra operation requirements in terms of input and output formats, relation size, speed, power, and throughput. This information is needed to determine if any of the technologies will be competitive with electronic systems. However, the all-optical, polarization, electrooptic and hybrid approaches to C&E appear viable. In addition, multistage networks of C&E modules may be useful for related problems with different performance requirements including telecommunication and inter-processor message routing.

Acknowledgements

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-86-C-0030.

References

1. Qadah, G. Z., Database machines: A survey, AFIPS Conf. Proc., Vol. 54: 1985 National Computer Conference, 211-223.
2. Ullman, J. D., Principles of Database Systems, Computer Science Press, Rockville, MD (1982).
3. Schwartz, J. T., Ultracomputers, ACM Trans. Prog. Lang. and Sys. 2(4), 484-521 (1980).
4. Sood, A. K., and Abdelguenti, M., Parallel and pipelined processing of some relational algebra operations, Int. J. of Electron., Vol. 59 (1985).
5. K. E. Batcher, Sorting Networks and their Applications, Proceedings of the 1968 Spring Joint Computer Conference, Vol. 32. AFIPS Press, Reston Va., 307-314.

6. Siegel, H. J., The universality of various types of SIMD machine interconnection networks, Proc. 4th annual Symp. Comput. Arch., Silver Spring, MD, Mar 23-25, 70-78 (1977).
7. Preparata, F. P., and Vuillemin, J., The Cube Connected Cycles: A Versatile Network for Parallel Computation, Comm. ACM 24(5), 300-309 (1981).
8. Lev, G. F., Pippenger, N., and Valiant, L. G., A Fast Parallel Algorithm for Routing in Permutation Networks, IEEE Trans. Comput. C-30(2), 93-100 (1981).
9. H. S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE Trans. Comput. C-20(2), 153-161 (1971).
10. D. E. Knuth, The Art of Computer Programming: Sorting and Searching, Vol. 3, Addison-Wesley, Reading Mass. 1973.
11. A. Lohman, What Classical Optics can do for the Digital Optical Computer, Applied Optics 25, 1543-1549 (1986).
12. A. A. Sawchuk and B. K. Jenkins, "Dynamic Optical Interconnections for Parallel Processors", Proc. SPIE 625, 143-153 (1986).
13. R. A. Spanke, "Architectures for Large Nonblocking Optical Space Switches", IEEE J. Quant. Elect. QE-22(6), 964-967 (1986).
14. Hill, A. M., One-Sided Rearrangeable Optical Switching Networks, J. Lightwave Tech. LT-4(7), 735-739 (1986).
15. Lee, Y. H., Gibbs, H. M., Jewell, J. L., Duffy, J. F., Venkatesan, T., Gossard, A. C., Wiegmann, W., and English, J. H., Speed and Effectiveness of Windowless GaAs Etalons as Optical Logic Gates, Appl. Phys. Lett. 49(1), 486-488 (1986).
16. Miller, D. A. B., Chemla, D. S., Damen, T. C., Wood, T. H., Burrus, C. A. Jr., Gossard, A. C., and Wiegmann, W., The Quantum Well Self-Electrooptic Effect Device: Optoelectronic Bistability and Oscillation, and Self-Linearized Modulation, IEEE J. Quant. Elect. QE-21(9), 1462-1476 (1985).
17. Wheatley, P., Bradley, P. J., Whitehead, M., Parry, G., Midwinter, J. E., Mistry, P., Pate, M. A., and Roberts, J. S., Novel Nonresonant Optoelectronic Device, Elect. Lett 23(2), 92-93 (1987).
18. Midwinter, J. E., private communication.

PART III

TABLE OF CONTENTS

<u>Section</u>	<u>Title</u>	<u>Page</u>
	INTRODUCTION	1
	SORTING WITH OPTICAL COMPARE-AND-EXCHANGE MODULES	5
	ALL-OPTICAL COMPARE-AND-EXCHANGE SWITCHES	35

LIST OF FIGURES

<u>Figure</u>	<u>Title</u>	<u>Page</u>
SORTING WITH OPTICAL COMPARE-AND-EXCHANGE MODULES		
1	Compare-and-Exchange Module	13
2	Pipelined Bitonic Sorter on a Perfect Shuffle Connected Network	14
3	Time Evolution of Digital Compare and Exchange	18
4(a)	Transfer Function of a Latching AND Gate	20
4(b)	Latching AND Implementation of Compare Operation	20
5	Regenerative Exchange Circuit	26
6	Polarization Encoded Exchange	28
ALL-OPTICAL COMPARE-AND-EXCHANGE SWITCHES		
1	The compare-and-exchange module. E represents the exchange signal, A, B represent the two input numbers, and H, L represent the higher number and the lower number, respectively.	38
2	Compare circuit in which L indicates the latching logic gate.	38
3	The use of $A_i \bar{B}_i$ and $\bar{A}_i B_i$ to compare A and B. When $A < B$, $\bar{A}_i B_i = 1$ will appear first. When $A > B$, $A_i \bar{B}_i = 1$ will appear first.	40
4	Generation of $A_i \bar{B}_i$ and $\bar{A}_i B_i$ using a single bidirectional, reflection-mode Fabry-Perot etalon.	40
5	Experimental layout for all-optical compare and exchange with IF- ZnS interference filter, $\lambda/2$ -half-wave plate, $\lambda/4$ - quarter-wave plate. E represents the exchange signal, A, B represent	

LIST OF FIGURES (CONTINUED)

<u>Figure</u>	<u>Title</u>	<u>Page</u>
	the two binary encoded numbers, and H , L represent the outputs of the larger number and the smaller number, respectively.	42
6	Expected operation of compare-and-exchange circuits test set 1 and 2 on horizontal axis. All curves are drawn upside down consistent with the experimental photographs. The vertical axes are in arbitrary units.	43
7	Computer simulated transfer functions for the ideal interference filters. The horizontal axes are input power in arbitrary units and the vertical axes are reflectivity or transmissivity of the filters. (a). Reflectivity R_1 of the comparator IF_1 with $I_0=0$, $I_1=A_i$ (or B_i), and $I_2=A_i+B_i$; (b). Reflectivity R_2 of the latching NAND gate IF_2 with $I_1=\bar{R}_i$, and $I_2=\bar{R}_1+A_i\bar{B}_i$; (c). Transmissivity T_3 of the latching AND gate with $I_1=\bar{R}_2$, $I_2=\bar{R}_2 + \bar{R}_1A_i\bar{B}_i$ (or $\bar{R}_2+\bar{A}_iB_i$), and $I_3=\bar{R}_2+\bar{R}_1A_iB_i + \bar{A}_iB_i$; (d). Transmissivity T_4 of IF_4 with $I_0=0$, $I'_0=E$, $I_i=A_i+B_i+E$; (e). Reflectivity R_4 of IF_4 with definitions as in (d); (f). The sum of (d) and (e).	44
8	Experimental results of the inputs A , B and the logic outputs $\overline{AB}$, $\overline{AB}$. The input powers are 11 mW each, and the output power is about 5 mW.	47
9	Output of the exchange-prohibited signal $\overline{R_1AB}$. The upper two traces show the two groups of numbers coming into the system. In the first group A is larger than B and in the second one, B is larger than A . The power of the holding beam is 19 mW, and the output power is about 6 mW.	47
10	Output of the exchange signal E . The holding power is 20 mW, and the output power is 5 mW.	49
11	The high output and the low output of the system with $A>B$. The input power is 14.5 mW, and the output power is 6.5 mW.	49

LIST OF FIGURES (CONTINUED)

<u>Figure</u>	<u>Title</u>	<u>Page</u>
12	The high output and the low output of the system with $B > A$. The input power is 14.5 mW, and the output power is 6.5 mW.	50

INTRODUCTION

Symbolic computation differs from its conventional numeric cousin in several fundamental ways. Foremost among the differences is the set of applications each intends to address. Typical symbolic computing applications include logical inference, information extraction, problem solving, and text/speech/image understanding. These applications typically require the processing of large amounts of information. In addition, many symbolic computing application environments are interactive and characterized by real-time performance requirements.

Using conventional serial hardware, however, the time it takes to process large amounts of symbolic information precludes real-time applications because the minimum time interval is fixed by practical considerations. For instance, multiplying two matrices takes $O(N^3)$ steps, where $N \times N$ is the number of elements in each matrix, and serial sorting can be done in no less than $O(N \log N)$ time steps, where N is the length of the list.¹ In addition, serial operations can only be pipelined to a limited degree: usually a memory fetch can occur while the cpu is processing another piece of data, but only one processing step can be performed at a time. Fortunately many symbolic computing operations have parallel algorithms that are pipelinable, and thus may run faster on parallel machines.

Symbolic computing applications that are parallelizable include calculating the transitive closure, shortest path or connected components of a relational graph. In addition, pruning a graph by consistent labeling with parallel matrix operations may reduce subsequent graph search times. Parallel algorithms for image and signal recognition include filtering and large kernel convolution and correlation. The logical set and relational algebra operations like intersection, union, division, projection, join, and cartesian product also can be speeded-up by parallel algorithms. In the rest of this report we will focus on sorting, which is common to both symbolic and conventional computation. We will begin by reviewing the existing parallel sorting algorithms. Parallel algorithms for the other symbolic computing operations will be the subject of future research.

Because of the real-time and large N constraints of symbolic computation, we will confine our discussions to parallel algorithms where the sorting time grows sublinearly with N : this immediately excludes linear arrays. Usually the uniform cost criterion is assumed when comparing algorithms, where all steps of the computation are of the same duration and processing, interconnect, and memory elements and operations are equally costly. Of course the uniform cost criterion is not applicable to systems with varying hardware

characteristics. In addition, all relative order arguments only apply for asymptotic values of N where constants and lower order terms are ignored.² However, N is bounded by technology constraints so the magnitude of the constants can be important when comparing real systems. Clearly technology and architecture dependent constants and relative costs are critical in a meaningful trade-off analysis between sorting systems. The following analysis will focus on these subtleties and result in the specification of optimal sorting systems with regard to the requirements of symbolic computing applications.

Sorting can be performed on 2-D array of processing elements in sublinear $O(N^{1/2})$ time.³ The nearest-neighbor communication of meshes allows the minimum temporal increment to be extremely small. Moreover, the 2-D topology makes them particularly well suited for implementation with 2-D technologies like electronics. While simple matrix operations can be pipelined for high throughput, most complex mesh algorithms like sorting are not pipelinable; and therefore, the system throughput equals the latency. In the high performance sorting applications found in symbolic computing, achieving a modest sublinear temporal complexity without pipelining is inadequate. Hence we must consider alternative sorting architectures.

The most powerful class of parallel algorithms are based on reconfigurable global communications between parallel processing elements and a common memory; hence they are called shared memory machines.⁴ The processing elements are fully connected through the memory and allow varying degrees of simultaneous memory reads and writes. Of the three general classes of parallel algorithms, shared memory computer algorithms can perform operations with the lowest number of time steps. For instance, sorting can be performed in $O(\log N)$ time and many graph problems benefit from shared memory. Abstract shared memory machines can also simulate both the mesh and network computers with no time delay.

While the temporal complexity of shared memory algorithms may be low, they usually require significantly more spatial resources than the mesh algorithms. In addition to limiting the size of the shared memory implementations, there is also a breakdown of the uniform cost criterion when comparing shared memory machines with meshes due to the globally reconfigurable interconnect. Global reconfiguration takes much more time in real systems than a simple nearest neighbor communication: the reconfiguration time increases substantially with N , the number of processing elements in the architecture. Thus shared memory implementations will be limited by practical

considerations to small-scale parallel architectures, which will not be optimal for sorting applications in symbolic computing.

Fortunately there is an alternative to the mesh and shared memory approaches to parallel computer architecture which we call network architectures. Network architectures are characterized by fixed, global communications between simple parallel processing elements. Sorting algorithms exist for network architectures that can be pipelined, have low delay $O(\log^2 N)$, and moderate spatial complexity $O(N \log^2 N)$.⁵ However in electronic network implementations, the global communications limit the maximum N and the minimum temporal interval from above and below respectively. Optical network implementations on the other hand are capable of building very large networks where the minimum temporal interval is limited by the propagation distance and the speed of light. At one nanosecond per foot, connection occurs quite rapidly in either fiber optic, bulk optical,⁶ or holographic systems. Thus the minimum temporal increment and N of optical networks can approach that of electronic meshes. The network architectures also obtain a low temporal complexity at the cost of reasonable spatial complexity while retaining the ability to pipeline the sorting operation.

From the preceding discussion it appears that network sorting algorithms are optimal for symbolic computing applications. In addition, optical networks are favored over their electronic counterparts because of their large size and bandwidth. The remainder of this report is devoted to issues concerning the optical implementation of network sorting algorithms. The first section details the design of optical implementations of the active portion of the network sorting algorithms, the compare-and-exchange operation. We propose using a distinctive feature of optical devices, namely bistability, that enables the construction of simple, hardwired circuits. At the end of this section we show how the properties of optical device families can be used to project the application domains of the resulting sorting networks. This paper has been accepted to a special issue of Applied Optics on optical computing.

In the second and last section we demonstrate an all-optical implementation of the compare-and-exchange operation using ZnS interference filters. This was a collaborative effort between BDM and the Optical Circuitry Cooperative of the Optical Sciences Center at the University of Arizona. This paper has been accepted to a special issue of the IEEE Journal of Selected Areas in Communications on photonic switching.

REFERENCES

1. D. E. Knuth, The Art of Computer Programming: Sorting and Searching, Vol. 3, Addison-Wesley, Reading Mass. 1973.
2. Aho, A. V., Hopcroft, J. E. and Ullman, J. D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA (1974).
3. S. G. Akl, Parallel Sorting Algorithms, Academic Press, NY (1985).
4. A. Borodin and J. E. Hopcroft, Routing, Merging, and Sorting on Parallel Models of Computation, J. of Comput. and System Sciences 30, 130 (1985).
5. K. E. Batcher, "Sorting Networks and their Applications," in Proceedings of the 1968 Spring Joint Computer Conference, Vol. 32. AFIPS Press, Reston Va., 307-314.
6. A. Lohmann, What Classical Optics can do for the Digital Optical Computer, Applied Optics, Vol. 25, 1543 (1986).
7. B. K. Jenkins, P. Chavel, R. Forchheimer, A. A. Sawchuk and T. C. Strand, Architectural implications of a digital optical processor, Applied Optics Vol. 23, No. 19, pp. 3465-3474 (1984).

SORTING WITH OPTICAL COMPARE-AND-EXCHANGE MODULES

C. W. Stirk and R. A. Athale

The BDM Corp.

7915 Jones Branch Dr.

McLean, VA 22102

ABSTRACT

Sorting is central to the solution of many knowledge-based and switching problems in advanced computation and communication systems. Parallel-pipelined sorting algorithms are appropriate for applications that demand high throughput, low delay and many data channels. One such algorithm, the bitonic sort, can be implemented with passive perfect shuffle interconnects between active stages of compare-and-exchange elements. In this paper we focus on optical hardware to implement the C&E operation and show that by taking advantage of a distinctive feature of optical logic, namely bistability, comparison circuits of remarkable simplicity are attainable. We describe implementations of C&E in a variety of optical device technologies capable of performing latching and nonlatching logic. Based on the device characteristics we outline potential application areas for each technology.

INTRODUCTION

In the early seventies it was estimated that 25% of all computer time was devoted to sorting.¹ With the widespread application of dedicated micro-controllers it is unlikely that this is still true; however, sorting remains one of the most common tasks in general-purpose computation. For instance, databases and expert systems often sort the elements of a data structure to simplify searching and the addition of new elements. Furthermore, data manipulation operations like projection, set union and intersection can be directly implemented by modified sorting algorithms.² Typically, knowledge-based systems operate on large numbers of related elements of information. As a general rule the number of parallel steps necessary to sort a data structure depends on the number of elements and the faster a sorting algorithm is, the more resources it requires. In other words the temporal complexity of the sorting problem is reduced at the expense of increased spatial complexity. Thus faced with a maximum amount of spatial resources and a minimum switching delay allowed by device-physics considerations, the time it takes to sort large structures in knowledge-based systems can prohibit real-time applications.

In addition to its widespread use in computation, sorting is also important in communications.³ In particular, parallel processor architectures can be interconnected by pipelined sorting networks

serving as message passing systems.⁴ Similarly, telecommunication packet switches can be based on sorting networks.⁵ Just as in knowledge-based systems, the hardware sorters for massively parallel architectures and subscriber loop communications must process large numbers of parallel channels with low delay. More importantly however, the sorting networks in communications must keep up with the data and packet generation rates--which can be considerable in large-grained parallel architectures and in trunk and video telecommunications. Hence, demands on the throughput of the sorting hardware mandate the use of parallel, pipelined sorting algorithms.

A sorting algorithm that fulfills the combined requirements of low temporal and spatial complexity along with high throughput is the bitonic network.³ The bitonic network can be pipelined in a multistage architecture that requires global interconnects and active compare-and-exchange (C&E) modules. It has been recognized that optical interconnects can provide the global connections needed between the stages.^{6,7,8} In addition, previous research described optical implementations of the exchange portion of the active modules using polarization switches,⁹ directional couplers¹⁰ and hybrid optoelectronic circuits.¹¹ In this paper we show the feasibility of simple, hardwired implementations of C&E in which all the processing is performed either optically or electrooptically. Specifically we propose using a family of devices for the comparison operation that employs bistability to combine logic and memory in a single device--obviating the need for external feedback as in flip-flops. In the next section we review the properties of the bitonic algorithm and its implications to electronic and optical implementations. In the last section we describe implementations of C&E using all-optical, hybrid optoelectronic and electrooptic logic devices. Based on the device characteristics, we outline the properties of the ensuing networks and their potential application domains.

PIPELINED BITONIC SORTING NETWORKS

A bitonic sequence of length N is composed of two sorted subsequences of length $N/2$, one monotonically increasing, the other decreasing. The bitonic merge combines the subsequences into a sorted sequence of length N using $\log N$ stages, each stage composed of compare-and-exchange modules and fixed interconnections between stages. The bitonic sorting algorithm for an arbitrarily ordered input list uses a divide-and-conquer strategy which begins by applying the bitonic merge to bitonic sequences of length 2, generating sorted sequences of length 2 and bitonic sequences of length 4. By repeated application of the bitonic merge, sorted and bitonic sequences of twice the length of those in the previous stage are formed. Thus, it takes $\log N$ applications of the bitonic merge to produce a sorted sequence of length N . Since the k th merge takes $\log k$ steps, the time complexity of the bitonic sort is $O(\log^2 N)$. If we have a pipelined bitonic sorting network then there are $N/2$ compare-and-exchange modules per stage: thus, the spatial complexity is $O(N(\log^2 N))$.

Pipelined bitonic networks are difficult to implement with electronic technology for several reasons. Because of the length and complexity of the interstage interconnects, 2-D layouts of the networks require area and communication distances that grow faster than N , the number of data channels.¹² In such wire-limited architectures the signal propagation delays are dependent on the number of channels and contribute to the overall sorting delay. Similarly, the long wires require large, high-power drivers that dominate the total system power for large values of N . Moreover the maximum signal bandwidth, and thus the throughput, is proportional to the difference in length of the wires in a stage or their RC time constants, whichever is larger. With the addition of high-speed buffers the time-skew limited throughput can be increased at the expense of increased delay. In conclusion, pipelined bitonic sorting networks in electronics are limited to applications with small numbers of data channels and low signal bandwidths.

Optical technology, on the other hand, is well-suited to implement the interconnects needed in sorting networks.^{6,7,8} Each bitonic interstage connection pattern can be emulated by a number of perfect shuffles with global, space-variant communications.¹³ Free-space optical implementations of the perfect shuffle have been demonstrated^{14,15} that exploit the third dimension for non-interacting communication channels. Since the active devices must share area only

with the connections' input and output transducers rather than the connections themselves. sorters with large numbers of channels can be fit into small areas and volumes. Besides the area and volume advantages, 3-D interconnects permit the interstage delay to be independent of the number of channels. Thus in contrast to electronics, the overall sorting delay grows only with the number of stages. Moreover for the moderate distances present in these architectures, the optical drive power is independent of the communication distance. These passive optical systems also have minimal time skew and may communicate information at optical media bandwidths; thus, the sorting throughput can be quite large and is limited in practice by the response time of the active devices. Finally due to the prevalence of optical technology in mass storage and communication environments, the data to be sorted may already be in optical form. In conclusion, optical technology will be competitive for sorting applications with large numbers of channels and/or high bandwidth signals.

The limiting feature of optical multistage sorting networks, in contrast to electronic implementations, is not the passive interconnection network, but the active processing performed in parallel between each communication step. The advantages of optical interconnections for sorting that we outlined above are dependent on the existence of optically compatible 2-channel sorting elements, the

C&E module shown in Figure 1. The compare operation determines the relative magnitude of the information on the two input channels. Depending on the result of the comparison operation, the exchange operation directs the larger and smaller input data to the output channels marked high and low, respectively. A pipelined bitonic sorter on a perfect shuffle connected network is presented in Figure 2.

Condition 1) if $\text{high} \geq \text{low}$ then $\text{high} \rightarrow \text{high}$ and $\text{low} \rightarrow \text{low}$

Condition 2) if $\text{high} < \text{low}$ then $\text{high} \rightarrow \text{low}$ and $\text{low} \rightarrow \text{high}$

Fig. 1 Compare & Exchange Rules

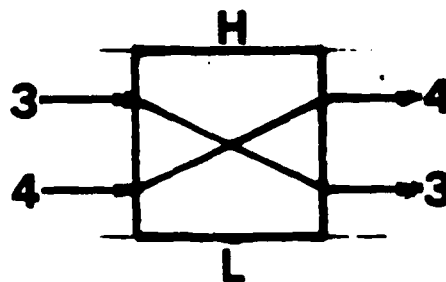
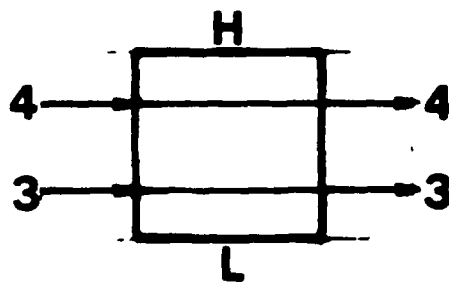


Figure 1. Compare-and-Exchange Module.

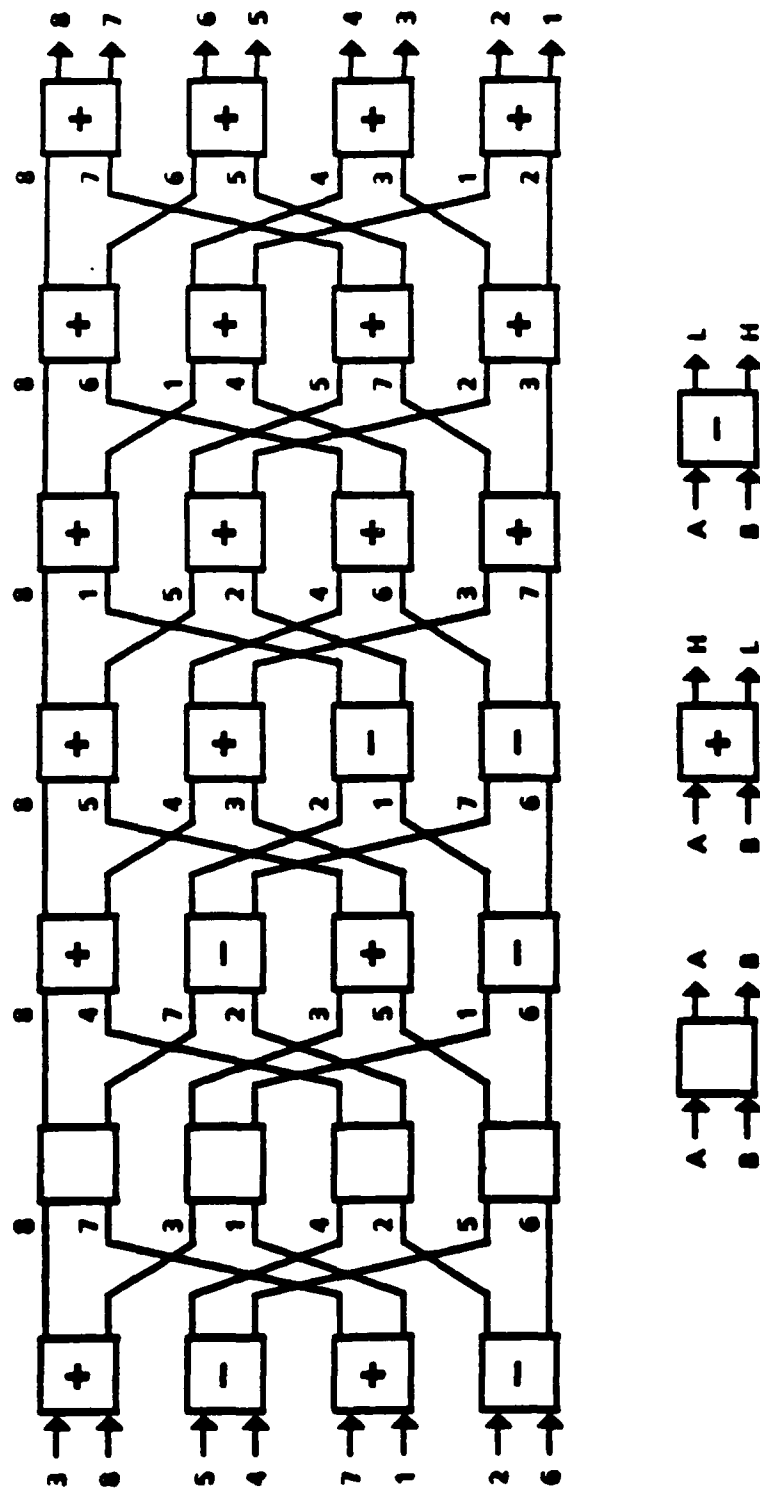


Figure 2. Pipelined Bitonic Sorter on a Perfect Shuffle Connected Network.

ANALOG IMPLEMENTATION OF COMPARE AND EXCHANGE MODULE:

Analog implementations of C&E have been proposed for associative memories and self-organizing systems ¹⁶. In this application, their role is to identify the element of a vector with the maximum value using a binary-tree architecture. Knowledge-based systems that do not involve high accuracy data could potentially use analog C&E and related operations for sorting and logical set operations. Unfortunately, analog sorting systems require system dynamic range much larger than the dynamic range of inputs. For the moment, let us assume that we desire error-free sorting by multistage analog C&E. Then the finite accuracy of each analog comparison calculation in the first stage places an initial upper bound on the allowable dynamic range of the inputs. In addition, multistage analog systems lacking signal restoration accumulate noise which further limits the useful dynamic range. Hence, noise introduced by non-uniform gain and crosstalk during or between the C&E processes is the most serious limiting factor since it will increase with each stage of the calculation. Clearly, the lowest signal-to-noise ratio is present at the last stage of the calculation and places the tightest restrictions on the allowable dynamic range of the inputs. If a specific dynamic range is desired for the inputs then the noise introduced by the system limits the number of possible stages. Since the number of stages is the logarithm or the logarithm squared of the number of inputs in deterministic sorting and selection networks, respectively.

noise also limits the number of data channels. Because of these apparent problems with analog approaches, we now turn to digital implementations of C&E.

DIGITAL IMPLEMENTATIONS OF COMPARE-AND-EXCHANGE MODULE:

Digital implementations of C&E have several advantages over analog approaches.¹⁷ A digital representation of data permits any finite dynamic range for the input values by simply specifying the number of bits. In addition digital logic can restore signal levels, and hence, the C&E units can be cascaded indefinitely. If crosstalk noise in the network is low and independent of the number of data channels, indefinite cascadability implies that the number of data channels is limited only by device-physics constraints like space and power. In contrast to analog implementations, for bit-serial data the internal complexity of the digital C&E modules remains constant regardless of the network size and the dynamic range of the inputs. Moreover, the low fan-in and -out of the bitonic network compensate for the low contrast and gain, respectively, in the active optical devices. Thus, the device requirements of the digital approach to multistage sorting networks appear to be compatible with the characteristics of optical and electrooptical technology. In this section we will show that C&E has simple, hardwired implementations that benefit from the bistable nature of many optical devices.

In a digital C&E module the comparison operation can be considered as a search for the most significant bit mismatch between the binary representations of the input data. The input word mismatch occurring closest to the most significant bit determines which datum is larger, and thus, the switch configuration. A rough outline of a serial algorithm for digital C&E schematically shown in Figure 1 is as follows:

- 1) input the data streams A and B most significant bit first into the high/low channels of the C&E module;
- 2) compare the two channels bit by bit; at the first occurrence of a mismatch between the strings, proceed to step 3 ;
- 3) if the mismatch is such that the A channel contains the larger datum, place the switch in the barred configuration (i.e. non-exchange position), otherwise the B channel contains the larger datum and place the switch in the crossed configuration (i.e. exchange position);

The time evolution of the switch position is illustrated in Figure 3 for typical input streams. The input data streams can be routed by the exchange switch subsequent to or even concurrent with comparison since up until the first mismatch the data streams are identical. Once the most significant mismatch has been detected and the exchange switch configuration determined, for correct operation the exchange switch must be insensitive to any subsequent mismatches. This property can be achieved in a number of ways, the most common

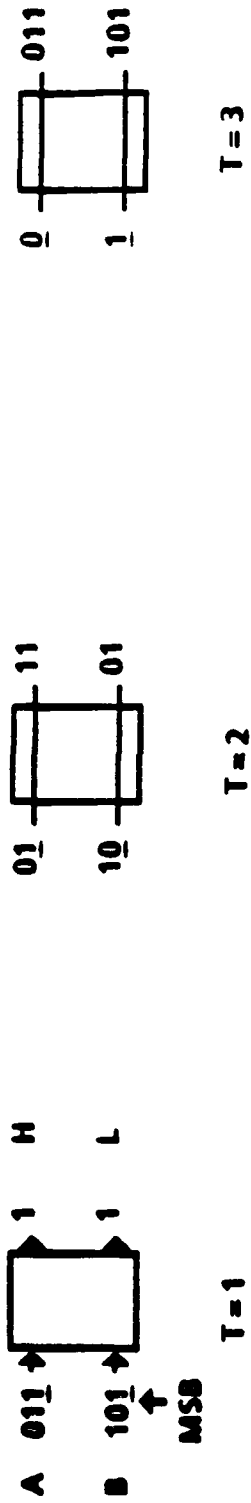


Figure 3. Time Evolution of Digital Compare and Exchange.

being feedback of the previous result of the comparison operation. Likewise, external flip-flops can record whether or not the mismatch has occurred and in which direction the switch should be set. Hence at the beginning of each word comparison the feedback signal or flip-flops must be reset to signify the mismatch has yet to occur.

LATCHING LOGIC DESIGN OF COMPARISON UNIT

Setting the exchange switch in a particular configuration until reset consists of remembering whether or not a mismatch has occurred and into which state, crossed or barred, the switch should be fixed. However, to reduce the complexity of the circuit we propose making the memory function inherent to the logic devices that perform comparison. This can be accomplished by using the bistability present in nonlinear logic devices with internal feedback. For example the transfer function of a latching AND gate is shown in Figure 4(a). While operating in latching mode, the device is biased up into the bistable loop. When the AND condition is first met, the state of the switch shifts to the upper part of the bistable loop. Since subsequent removal of all the inputs except for the bias does not change the output of the gate, the gate is effectively latched into the logical state true. Previously the only proposed uses for bistability in optical computers was for delay lines or memory elements.^{18,19} By extending the techniques we have presented here, it can be shown that latching NAND, OR, and NOR gates are possible with appropriate bistable loops and bias levels.

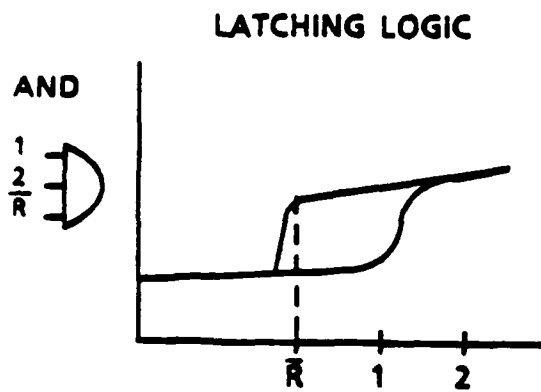


Figure 4(a). Transfer Function of a Latching and Gate.

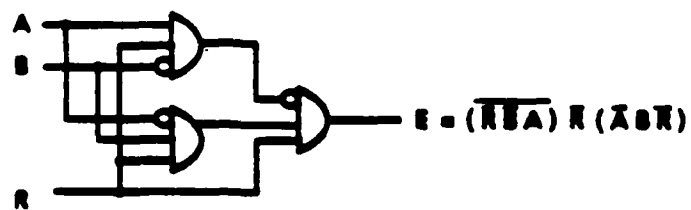


Figure 4(b). Latching and Implementation of Compare Operation.

Once one understands how latching logic works, the next task is to build latching logic circuits that perform a useful function. Unfortunately, we do not know of any general methods to design latching logic circuits based on a description of the intended circuit function. However, we succeeded in designing the latching AND circuit for comparison shown in Figure 4(b) that operates in the following manner. The top and bottom first layer AND gates are designed to latch at the first occurrence of the mismatches ($A_i > B_i$) and ($B_i > A_i$), respectively. If ($A_i > B_i$) occurs before ($B_i > A_i$) then the top gate latches to true while the output of the bottom gate is unlatched at false. The latched output of the top gate is then inverted to false, preventing the second layer gate from ever latching to the state (exchange = true), regardless of changes in the state of the bottom gate. Thus the output of the comparison module is effectively latched to the state (exchange = false). Conversely, if ($B_i > A_i$) occurs before ($A_i > B_i$) the bottom gate latches to true while the top gate remains unlatched at false. Thus the second layer gate and output of the comparison modules is directly latched to the state (exchange = true). Since removal of the bias causes all the latching gates to relax to the false state, the complement of the inter-word reset signal should be used as the bias to all of the latching gates. Since each gate latches at most once per word, the

switching duty cycle--and hence power dissipation in the comparison module--decreases with increasing word length. Other latching logic families, like those based on latching NAND gates, form the basis of alternative comparison circuits.²⁰

OPTICAL IMPLEMENTATIONS OF LATCHING LOGIC FOR COMPARISON

Latching logic gates can be fabricated using a variety of optical logic technologies. Each device technology has associated with it a set of performance characteristics that are crucial to the selection of the relevant application domain. Among the critical characteristics are switching speed, power, wavelength of operation, size and technological maturity. In this section, we will highlight a few of these device technologies and show how their individual characteristics limit their intended applications.

Bistable Fabry-Perot etalons can implement latching AND gates for the compare operation. The latching circuit we outlined above tolerates the low gain and contrast of etalons because the fan-out and -in required of the latching gates is at most one and three, respectively. Because of their high speed²¹ nonlinear etalons are well suited for switching broadband signals. The speed of the latching circuits based on etalons may be limited by the cavity build-up time required to reach the stable state. A device with a nonsymmetric cycle time²² (fast switch-on and slow switch-off) is useful if the packet frequency is small compared to the bit frequency

Since at a fixed bandwidth the power dissipation in each comparison module decreases with increasing header length, large networks based on etalon comparison may be feasible. However as we shall see in the next section, large networks demand signal restoration whose total power dissipation grows with the bandwidth and network size. Anyhow, at a fixed bandwidth the power dissipation in the comparison module decreases with decreasing packet frequency. This is especially useful for applications that generate very long packets relatively infrequently such as inter-computer communications and video telecommunication.

Slightly slower speeds for comparison are possible with symmetric SEED devices serving as the latching AND gates.²³ Just as with bistable etalons, the latching SEED devices must wait for the positive feedback (in this case electrical) to build up to place the output in a stable state. Comparison units based on SEED devices offer a variable speed/power tradeoff: ²⁴ therefore, higher levels of integration may be possible for low bandwidth signals before thermal dissipation becomes a problem. Thus, SEED arrays appear well suited for subscriber loop and intra-computer communications where the data rates are relatively lower but the number of channels is higher than the previous applications.

Bistable laser diodes also possess the necessary properties to implement latching logic. The high gain and contrast of laser diodes make them particularly well suited for environments where the

interstage connections create considerable losses and crosstalk. In addition, laser diode manufacturing technology has demonstrated its maturity, single mode fiber compatibility and ability to form 2-D arrays of devices. However the physical size and total power dissipation of the devices can be quite large, preventing their incorporation into very large integrated structures. Thus, they seem best suited for the trunk and inter-computer communications applications which involve a small number of high bandwidth channels.

There are alternatives for comparison implementation that are based on hybrid logic device designs. These designs detect the incoming light signals and then modulate one of the input signals or a bias signal to produce the desired logic operations.²⁵⁻²⁷ The primary advantage of this approach is that sophisticated electronic processing can be performed on the detected signal before it is applied to the modulator. For instance, complex switching nodes for store-and-forward packet switches may be attainable. The use of special modulating materials with intrinsic memory characteristics such as the Ferroelectric Liquid Crystals and other inorganic ferroelectric electrooptic materials will lead to latching logic devices. This technology is, however, immature and most device response times are on the order of milliseconds, making them too slow for the applications under consideration. New developments in

materials research to enhance the response time and successful incorporation of fast materials into functional devices will necessarily lead to a re-evaluation of this technology.

OPTICAL IMPLEMENTATION OF THE EXCHANGE UNIT

Spatial position encoded exchange units built with conventional non-latching logic can restore signal levels. Thus only signal to noise, crosstalk, uniformity, power and other systems engineering considerations limit the number of channels per stage and the total number of stages. Because noise does not propagate between stages, restoring exchange applies to deep networks. The schematic circuit diagram of the exchange unit is shown in Figure 5. The AND operation can be performed by any nonlinear optical device (all-optical or hybrid) with a sigmoidal input-output response and proper biasing. The two OR gates in the second stage receive signals that mutually exclusive, and hence can be implemented by passive combiners. As the logic expressions in Figure 5 indicate, the output H will be equivalent to A (and output L equivalent to B) if the exchange signal E is "0", and the signals at the output port will be interchanged if E is "1". Like the technology that is available for comparison, the networks based on restoring exchange devices span the spectrum from narrowband and large to broadband and small.

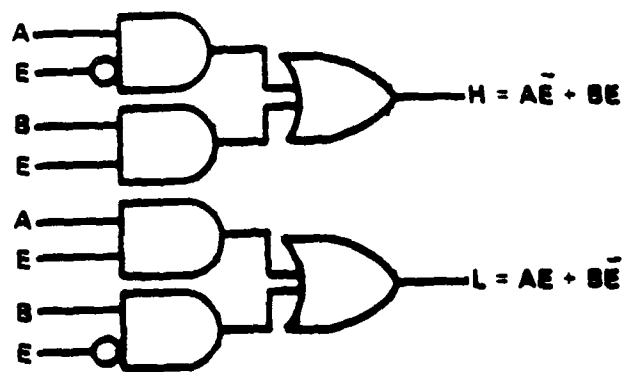


Figure 5. Regenerative Exchange Circuit.

The potential throughput per channel is greatly increased if the exchange module uses passive switches. In this case, the bandwidth of the message header is determined by the response time of the comparison logic while the trailing information can propagate at optical media bandwidths within the signal-to-noise limits imposed by losses in the passive switches. Polarization encoded switching using Wollaston prisms and controllable half-wave plates² is one technology that performs passive routing. A photoactivated, polarization encoded exchange unit is shown in Figure 6. A photodiode, photoconductor or phototransistor receives the exchange signal and produces an electric field dependent change in the polarizability of the dynamic half-wave plate through the electrooptic effect. When activated, the dynamic half-wave plate rotates the polarization of the orthogonally polarized signal beams through 90°, thereby acting as a passive switch. The Wollaston prism or polarizing beamsplitter subsequently separates the high and low channels. The advantage of polarization switching, in addition to its passive nature, is that exchange occurs in one stage and the data may occupy the same spatial channel. Similarly, the data can be wavelength multiplexed for further increases in bandwidth. In addition, the fan-out of the previous comparison module only has to be one. However, the frame rate of optically controlled, dynamic half-wave device arrays is presently constrained to the millisecond regime by the material characteristics and the combined optical and electrical switching power dissipation limitations. Since exchange based on polarization-tagging is non-regenerative, the the number of

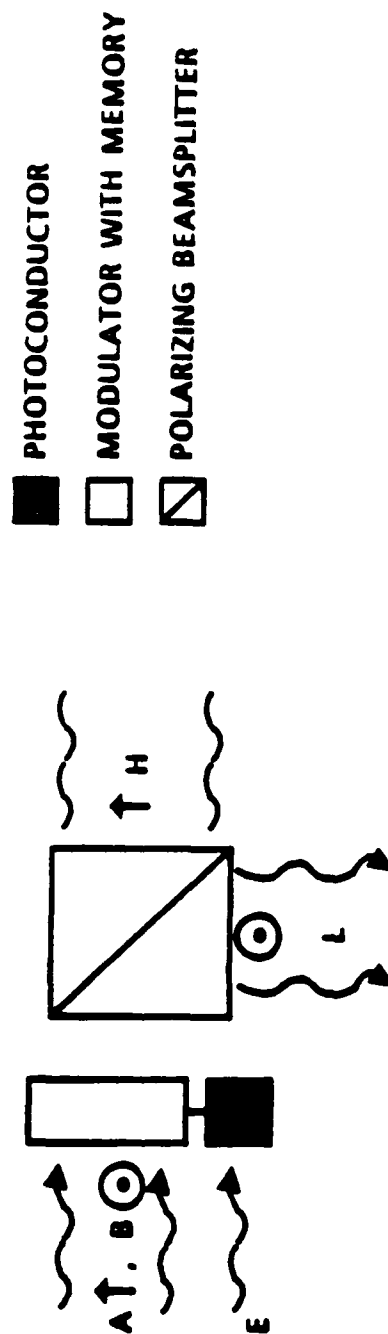


Figure 6. Polarization Encoded Exchange.

stages and thereby the number of inputs is primarily limited by absorptive, diffractive, sampling and scattering losses. Thus, they are limited to small networks with low packet frequency but high data rates like inter-computer communications or video telecommunications.

SUMMARY AND CONCLUSIONS

In this paper we reviewed why optical interconnects are appropriate to implement pipelined sorting networks for telecommunication and parallel-processing applications. We went on to propose optical implementations of the active compare-and-exchange operation that are essential to the sorting networks. In particular, we described a class of Boolean logic devices called latching logic which permits the design of simple, hardwired comparison modules. Latching logic significantly reduces the interconnect and gate complexity of the compare module over the non-latching logic approach. Based on the available device characteristics we outlined the application domains of sorters utilizing a variety of optical technology. Which technology one chooses depends on the requirements of the application of interest. One application where optics will compete most favorably with electronics is when the packets are long and infrequent, and where low delay and high throughput are paramount - for example video telecommunications and inter-processor message routing. Optics also appears competitive at the other extreme of intra-processor and subscriber-loop communication where the signals are much slower but involve very large numbers of channels.

ACKNOWLEDGEMENTS

The authors would like to thank C. Friedlander of BDM and H. M. Gibbs, R. Jin, G. Khitrova, K. Wagner and L. Zhang of the Optical Circuitry Cooperative at the Optical Sciences Center of the University of Arizona for their contributions to the understanding of compare-and-exchange and its implementations.

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-86-C-0030. An unrefereed version of this paper was presented at SPIE 754 in Jan. 1987.¹⁶

REFERENCES

1. D. E. Knuth. The Art of Computer Programming, Vol. 3, Sorting and Searching. Addison-Wesley, Reading Mass. 1973.
2. J. T. Schwartz, Ultracomputers. ACM Trans. Prog. Lang. and Sys. 2(4), 484-521 (1980).
3. K. E. Batcher, Sorting Networks and their Applications, Proceedings of the 1968 Spring Joint Computer Conference, Vol. 32. AFIPS Press, Reston Va., 307-314.
4. H. P. Moravec, Fully Interconnecting Multiple Computers with Pipelined Sorting Nets, IEEE Trans. Comput. Vol. C-28, No. 10, 795 (1979).
5. A. Huang and S. Knauer, Starlite: A Wideband Digital Switch, Proc. IEEE Global Telecommunications Conference, Atlanta, Georgia, Vol. 1, 121, November 1984.
6. J. W. Goodman, F. J. Leonberger, S. Y. Kung and R. A. Athale, Optical Interconnections for VLSI Systems, Proc. IEEE, Vol. 72, No. 7, 850-866 (1984).
7. J. E. Midwinter, 'Light' Electronics, Myth or Reality?, IEE Proc., Vol. 132, Pt. J, No. 6, 371-383 (1985).

8. A. Huang, The Relationship Between STARLITE, a Wideband Digital Switch and Optics, Proc. International Conference on Communications, Toronto, Canada, June 22, 1986.
9. A. W. Lohmann, What Classical Optics can do for the Digital Optical Computer, Applied Optics 25, 1543-1549 (1986).
10. H. S. Hinton, Applications of the Photonic Switching Technology for Telecommunications Switching, To appear in Proc. International Conference on Communications, June 7-10, 1987.
11. J. E. Midwinter, Novel approach to the design of optically activated wideband switching matrices, IEE Proc., Vol. 134, Pt. J, No. 5, 261-268 (1985).
12. C. D. Thompson, The VLSI Complexity of Sorting, IEEE Trans. Comput. C-32(12), 1171-1183 (1983).
13. H. S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE Trans. Comput. C-20(2), 153-161 (1971).
14. A. Lohmann, W. Stork and G. Stucke, Optical Implementation of the Perfect Shuffle, Proc. OSA Topical Meeting on Optical Computing, Lake Tahoe, NV, paper WA3 (1985).
15. C. W. Stirk, R. A. Athale and M. W. Haney, The Folded Perfect Shuffle Optical Processor, to appear in Applied Optics, Jan. 1, 1987.

16. R. Lippmann, "An Introduction to Computing with Neural Networks", IEEE ASSP Magazine, 1 April, 1987.
17. C. W. Stirk, R. A. Athale and C. B. Friedlander, Optical implementation of the compare-and-exchange operation for applications in symbolic computing, to appear in Proc. SPIE 754-27.
18. B. S. Wherrett, All-optical computation: a design for tackling a specific problem, Applied Optics, Vol. 24, No. 17, 2876-2883 (1985).
19. A. C. Walker, Application of bistable optical logic gate arrays to all-optical digital parallel processing, Applied Optics, Vol. 25, No. 10, 1578-1585 (1986).
20. L. Zhang, R. Jin, C. W. Stirk, G. Khitrova, R. A. Athale, H. M. Gibbs, H. M. Chou, R. W. Sprague and H. A. Macleod, All-Optical Compare-and-Exchange Switches, submitted to IEEE Journal on Selected Areas in Communications Special Issue on Photonic Switching, Oct. 87.
21. Lee, Y. H., Gibbs, H. M., Jewell, J. L., Duffy, J. F., Venkatesan, T., Gossard, A. C., Wiegmann, W., and English, J. H., Speed and Effectiveness of Windowless GaAs Etalons as Optical Logic Gates, Appl. Phys. Lett. 49(1), 486-488 (1986).
22. A. Migus, A. Antonetti, D. Hulin, A. Mysyrowicz, H. M. Gibbs, N. Peyghambarian and J. L. Jewell, One-picosecond optical NOR gate at room temperature with a GaAs-AlGaAs multiple-quantum-well nonlinear Fabry-Perot etalon, Appl. Phys. Lett., 48(1), 70 (1985).

23. A. L. Lentine, H. S. Hinton, D. A. B. Miller, J. E. Henry, J. E. Cunningham, L. M. F. Chirovsky, The Symmetric Self Electro-optic Effect Device, CLEO 87, post deadline paper THT12, pg 249.
24. D. A. B. Miller, D. S. Chemla, T. C. Damen, T. H. Wood, C. A. Burrus Jr., A. C. Gossard, and W. Wiegmann, The Quantum Well Self-Electrooptic Effect Device: Optoelectronic Bistability and Oscillation, and Self-Linearized Modulation, IEEE J. Quant. Elect. QE-21(9), 1462-1476 (1985).
25. R. A. Athale, "Studies in Digital Optical Processing", Ph.D. Thesis. University of California, San Diego, 1980..
26. S. H. Lee, S. C. Esener, M. A. Title and T. J. Drabik, Two-dimensional Si/PLZT Light Modulators: Design Considerations and Technology, Opt. Eng. Vol. 25, No. 2, 250 (1986).
27. P. Wheatley, P. J. Bradley., M. Whitehead, G. Parry, J. E. Midwinter, P. Mistry, M. A. Pate and J. S. Roberts, Novel Nonresonant Optoelectronic Device, Elect. Lett 23(2), 92-93 (1987).

All-Optical Compare-and-Exchange Switches

Lei Zhang, Ruxiang Jin, C.W. Stirk, G. Khitrova, R.A. Athale,
H.M. Gibbs, H.M. Chou, R.W. Sprague, and H.A. Macleod

Abstract—All-optical compare and exchange is experimentally demonstrated using ZnS bistable optical devices. The compare-and-exchange demonstration utilizes polarization multiplexing and filtering, and latching and bidirectional logic. The combination of 2-D arrays of compare-and-exchange modules with optical perfect-shuffle interconnections leads to pipelined optical sorting networks that can process large numbers of high-bandwidth signals in parallel. Optical sorting networks with these characteristics are applicable in telecommunication switches, parallel processor interconnections and database machines.

The Arizona portion of this research was supported by DARPA/RADC, SDIO and OCC. The BDM portion was funded by DARPA/AFOSR under Contract Number F49620-86-C-0030.

Lei Zhang, was a visiting student at Optical Sciences Center, University of Arizona, Tucson, AZ 85721, from the Department of Applied Physics, Harbin Institute of Technology, Harbin, People's Republic of China.

Ruxiang Jin, G. Khitrova, H.M. Gibbs, H.M. Chou, R.W. Sprague, and H.A. Macleod are with Optical Sciences Center, University of Arizona, Tucson, AZ 85721

C.W. Stirk, and R.A. Athale are with The BDM Corporation, 7915 Jones Branch Drive, McLean, Virginia 22102-3396

I. Introduction

Sorting is one of the most common and well-understood topics in computer science. It is known that serial sorting algorithms require at least $O(N \log N)$ temporal complexity [1]. Hardware based on parallel sorting algorithms offers enhanced performance on problems that must rapidly sort large quantities of information. Since the number of clock cycles, devices and interconnects are limited resources in any processing environment, we need parallel algorithms with sublinear temporal and practical spatial complexity. In addition, the algorithms we choose must be optimum with respect to our specific implementation technology. For instance the mesh algorithms developed for VLSI require only nearest neighbor connections and are sublinear $O(N^{1/2})$ in temporal complexity [2]. Unfortunately, mesh algorithms must finish sorting one sequence before beginning another; thus their throughput is limited by their latency. On the other hand, the shared memory [3] and some network [4] algorithms have the lowest temporal complexity $O(\log N)$ of all sorting algorithms, but are not practical with current technology since they require globally reconfigurable interconnects and excessive spatial resources, respectively.

Network algorithms based on the bitonic sort [5] have sublinear temporal complexity $O(\log^2 N)$. Moreover, they can be pipelined in stages for high throughput; and thus, are useful in problems where throughput is as critical as latency. But the bitonic sorting network requires at least one globally-connected interstage communication pattern. For instance the perfect-shuffle [6] connection pattern transmits half the information present in the top half of a list to the bottom half and vice versa. Because VLSI is confined to the 2-D surface of a chip and electrons in wires are capacitively coupled, practical electronic perfect-shuffles are limited to small numbers of channels and low data rates. In contrast, the noninteracting nature of photons and 3-D connection capability of optics allows optical perfect-shuffle networks to have large numbers of parallel channels and high data rates [7]-[9]. Thus, optical sorting networks based on the perfect-shuffle interconnection and bitonic algorithm are desirable when the number of communication channels or the data rates exceed the capabilities of electronic systems.

In particular, optical sorting networks are applicable in telecommunication switches that route high-bandwidth optical data packets [10]. Telecommunication switches must handle many parallel channels, have low latency and keep up with the packet generation rates. Similarly, high-throughput sorters serve as the communication

fabric of electronic multiprocessors [11]. In these parallel processors the number of processing elements, and thereby the computational power, is governed by the number of parallel data channels. Furthermore, the throughput of each processing element is limited by the interconnection latency and throughput. In addition, sorting hardware may serve as dedicated subsystems for parallel database operations [12] in conjunction with optical memories [13]. Parallel and independent memory access can generate data rates beyond the capabilities of electronic systems.

Network sorting algorithms need, in addition to perfect-shuffle interconnections, 2x2 self-routing crossbar switches where each routing decision depends on the relative magnitude of the local information. Hence, we desire implementations of the 2x2 self-routing crossbars that are compatible with optical interstage connections and fulfill the requirements of bandwidth and parallelism in sorting applications. The function of such self-routing crossbars can be separated into the operations of comparison and exchange: comparison determines the relative magnitude of the local data; exchange configures the crossbar switch dependent on the outcome of the comparison.

In all subsequent discussions we assume a binary representation for the data. Overscores represent the invert operation; thus $\bar{R}_1$ and $\bar{R}_2$ are the logical complements of the system reset. Brackets "[]" contain a latching condition which we will explain shortly.

An algorithm for compare and exchange proceeds as follows: we label the synchronous input channels A and B, and operate serially from most to least significant bit. If $A_i > B_i$ occurs before $B_i > A_i$, where i represents the bit position, then the switch latches into the "don't-exchange" position. Conversely if $B_i > A_i$ occurs first then the switch latches into the exchange position (Fig.1). Latching implies that once an inequality has been detected, the exchange switch becomes set into one particular configuration until the system is reset.

Optical bistable devices have the potential for high-speed optical signal processing and computing [14]-[15]. ZnS and ZnSe bistable interference filters have already been used to demonstrate simple digital optical circuits, pattern recognition, symbolic substitution, and one-bit addition, because they can be operated in the visible spectrum and are relatively easy to fabricate [16]-[17]. In this paper we experimentally demonstrate a circuit that performs compare and exchange with ZnS interference filters as bistable devices. Here the ZnS interference filters are used in less common modes of operation including latching and bidirectional logic. In addition

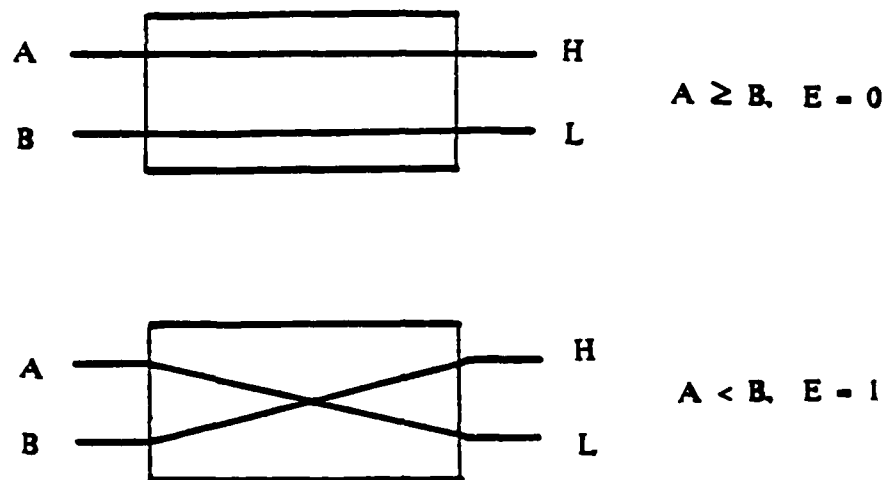


Figure 1. The compare-and-exchange module. E represents the exchange signal, A, B represent the two input numbers, and H, L represent the higher number and the lower number, respectively.

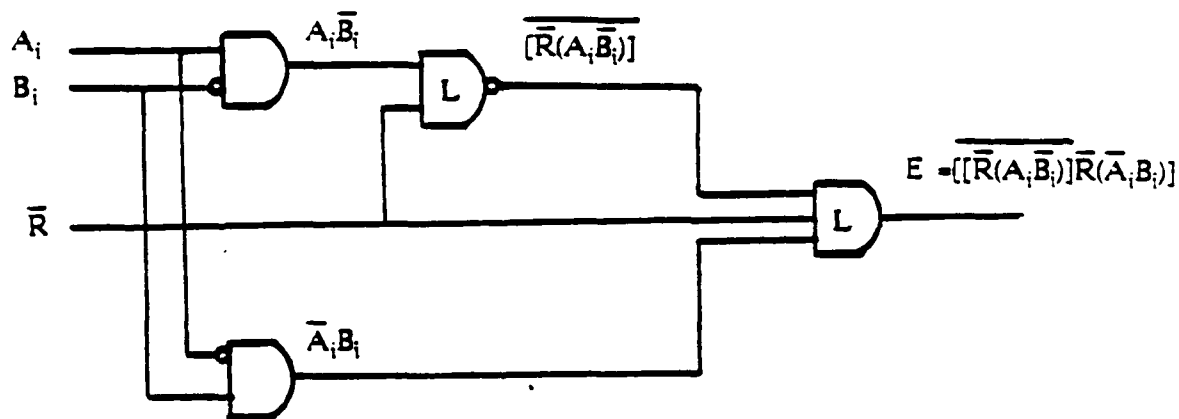


Figure 2. Compare circuit in which L indicates the latching logic gate.

we employ polarization multiplexing and filtering to achieve channel isolation. 4-port bidirectional devices and reduced feedback. In the next section we outline the design of one possible compare-and-exchange circuit without regard to the implementation technology. We also illustrate the expected operation of each part of the circuit. In the following section we present the general layout and operation of the compare-and-exchange design using ZnS interference filters along with polarization multiplexing and filtering. In the discussion section we compare the experimental and expected results. We conclude with some general comments.

II. Compare-and-Exchange Circuit Design

More than one circuit design is possible for comparison [18]. The circuit diagram for the comparison circuit demonstrated in this paper is shown in Fig.2. It consists of three parts. The first part is a comparator to distinguish between the cases where $A_i > B_i$ or $A_i < B_i$. Fig.3 shows how this can be done by generating $A_i \bar{B}_i$ and $\bar{A}_i B_i$. In the second and third parts, two latching gates are employed, so that when $A < B$, $\bar{A}_i B_i = 1$ comes first, and one latching gate will be switched-on to give an exchange signal $E = 1$. It remains in the on-state until all the bits of A and B are transmitted. Similarly, when $A > B$, $A_i \bar{B}_i$ comes first, and another latching gate will be switched-on to prevent the exchange. From Fig.3 we see that $A_i \bar{B}_i = 1$ and $\bar{A}_i B_i = 1$ never occur simultaneously, making it possible to separate the state of the latching gates.

Like comparison, however, there is more than one way to implement exchange. The appropriate choice depends on the application requirements, the technology characteristics and the corresponding comparison circuit. For demonstration purposes we will construct an active exchange module. If the exchange signal $E = 1$ is present, it sends B to the H-output and A to the L-output. Otherwise if the exchange signal is 0, it sends A to H-output and B to the L-output.

The above discussion shows that our circuit design needs a comparator, two latching gates, and an exchanger. Fig.4 shows that $A_i \bar{B}_i$ and $\bar{A}_i B_i$ can be generated from a single bistable etalon by using its reflections from both sides, so that a single bistable device can be used as a comparator. The latching operation is also natural for bistable devices, so that the two latching gates are just two bistable devices. Another bistable device is used as the exchanger with its transmission determined by the exchange signal. The details of their operations are discussed in the following section.

A_i	B_i	$A_i \bar{B}_i$	$\bar{A}_i B_i$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Figure 3. The use of $A_i \bar{B}_i$ and $\bar{A}_i B_i$ to compare A and B. When $A < B$, $\bar{A}_i B_i = 1$ will appear first. When $A > B$, $A_i \bar{B}_i = 1$ will appear first.

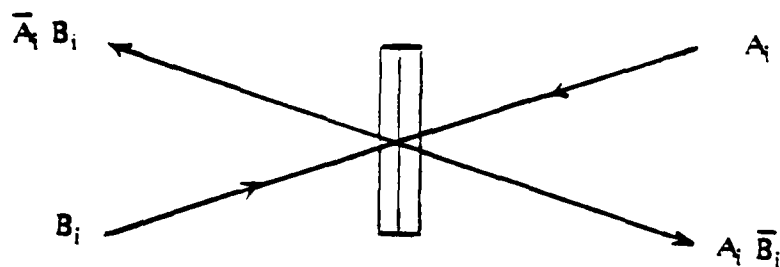


Figure 4. Generation of $A_i \bar{B}_i$ and $\bar{A}_i B_i$ using a single bidirectional, reflection-mode Fabry-Perot etalon.

III. Experimental Demonstration

The experimental layout for the compare-and-exchange circuit is shown in Fig.5. An Argon laser and a phase grating generate four optical beams, each having a peak power of about 40 mW. A chopper modulates beams A and B with the test sets, and blocks the holding beams $\bar{R}_1$ and $\bar{R}_2$ between each test set to allow the latching gates to reset. A half-wave plate gives the two holding beams $\bar{R}_1$ and $\bar{R}_2$ s polarization. A quarter-wave plate gives the beams A and B circular polarization.

For the experimental demonstration of all-optical compare and exchange we choose test vectors of $A = 110001$, $B = 101011$ and $A = 100011$, $B = 110101$. In the former set of test vectors $A_i > B_i$ occurs first; all four permutations of $A_i B_i$ follow to ensure that the switch is properly latched. Similarly, for the latter group of test vectors we find that $B > A$ and demonstrate the exchange stability to further permutations. Up until the first mismatch the position of the exchange switch is not important to first-order approximation since the output data streams are identical. In Fig.6 we depict the expected operation of the latching compare and passive circuits described above for both test sets. All data used in the simulations are based on the structure of each filter. The curves are drawn upside down to be consistent with the experimental photographs. We see that whether $A > B$ or $A < B$, the larger number always goes to the H-output. We did not fit the simulations with the experimental results because we wanted to show the ideal results with suitable devices. The transfer functions shown in Fig.7 used the same data.

The compare circuit operates in the following manner. The circularly polarized data beams, A and B, are incident on two polarizing beam splitters (PBS's). These PBS's serve two functions. One function is to sample the data beams for the compare operation: the p-polarization from the A channel propagates through the PBS for comparison, the s-polarization is reflected to the exchange switch; conversely, the s-polarization of the B beam is reflected for comparison and the p-polarization propagates through the PBS to the exchange switch while its polarization is rotated by the half-wave plate to match that of A. The orthogonally polarized data beams that were injected into the compare circuit are converted to circular polarization by two quarter-wave plates. The circularly polarized data beams are incident on the first interference filter (IF₁).

IF₁ operates in reflection mode as a bidirectional comparator. If B_i is zero and if A_i is one, then the circularly polarized A_i is reflected by IF₁, converted to the s-polarization by the quarter-wave plate and reflected by the PBS to produce the signal

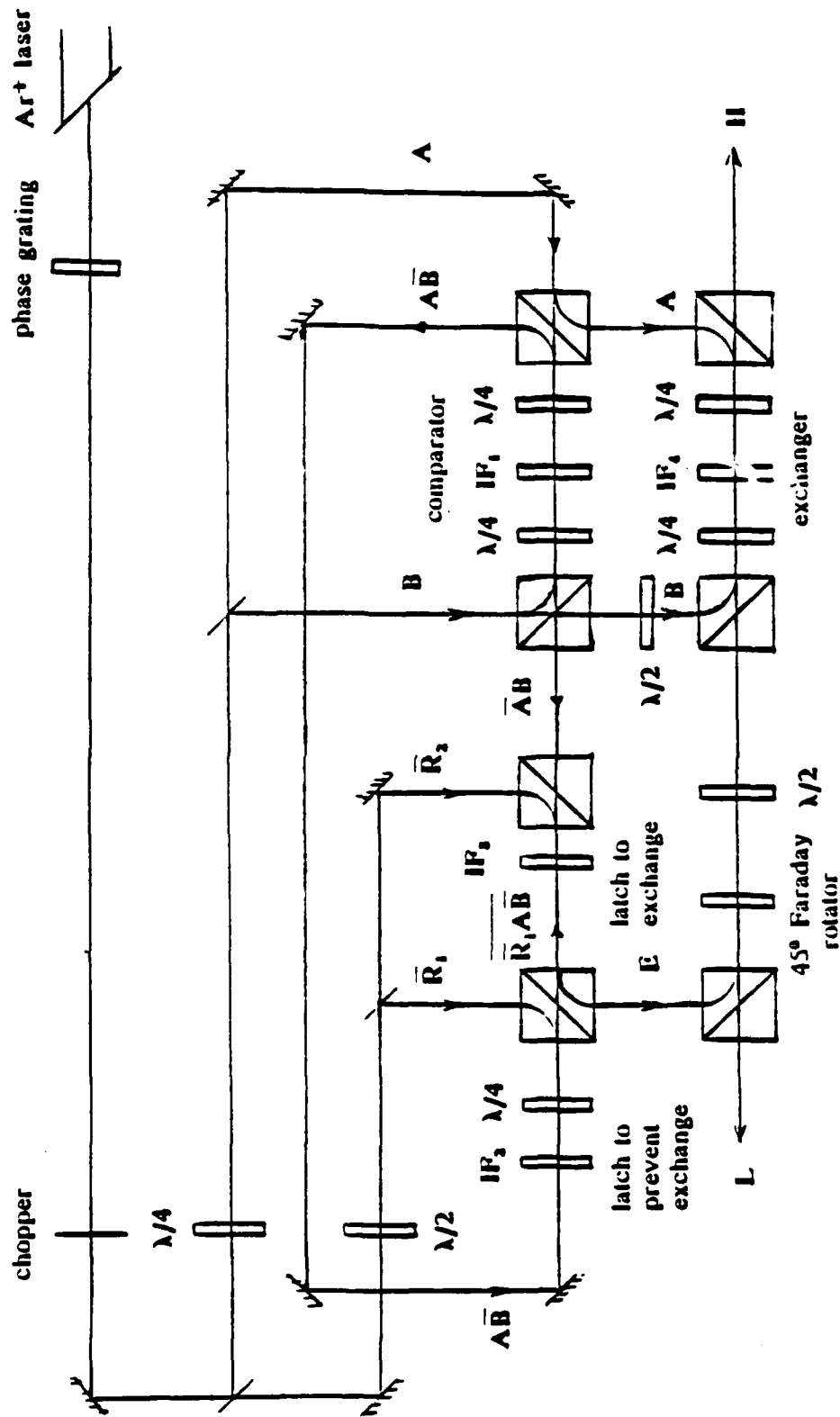


Figure 5. Experimental layout for all-optical compare and exchange with IF- ZnS interference filter, $\lambda/2$ - half-wave plate, $\lambda/4$ - quarter-wave plate. E represents the exchange signal, A, B represent the two binary encoded numbers, and H, L represent the outputs of the larger number and the smaller number, respectively.

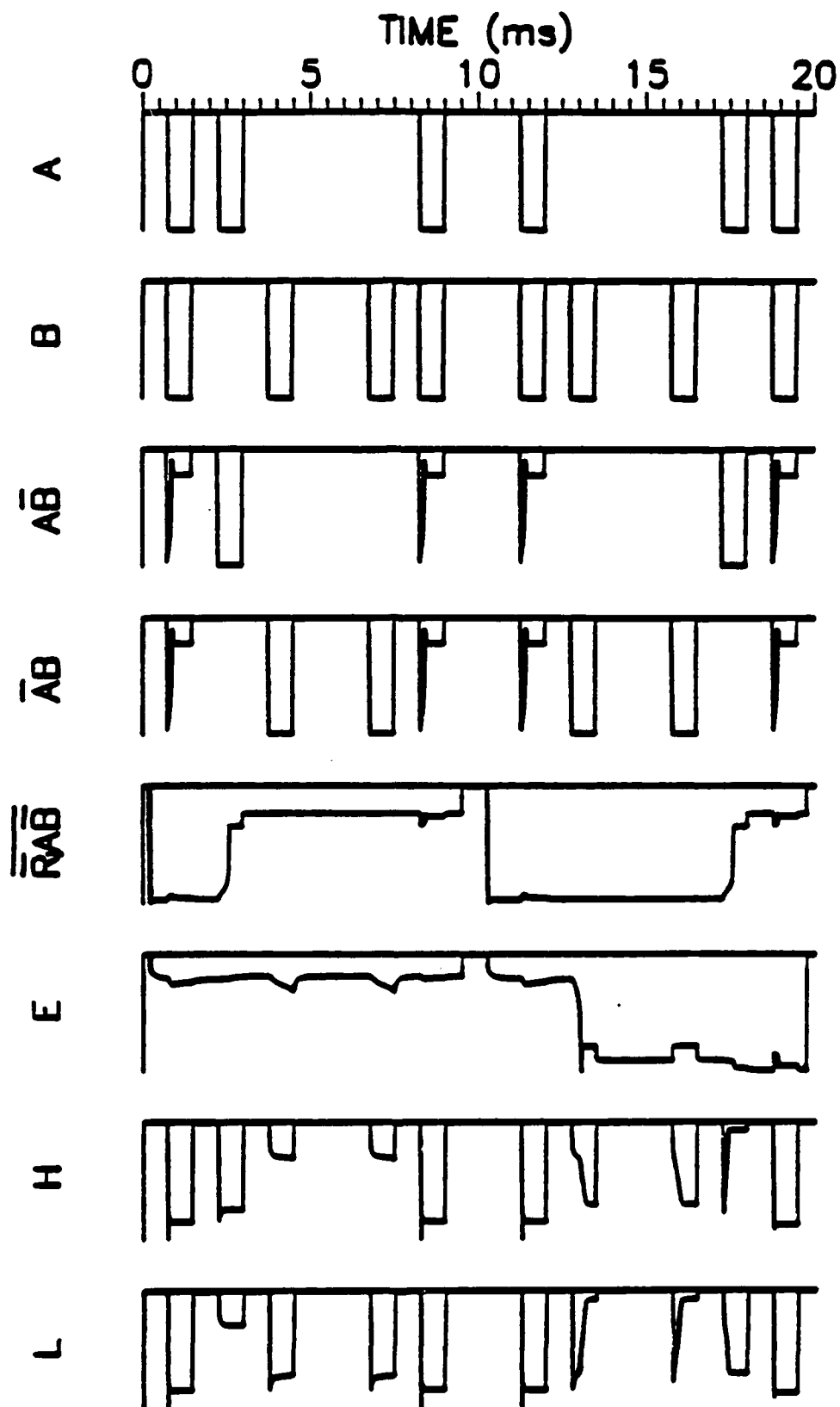


Figure 6. Expected operation of compare-and-exchange circuits test set 1 and 2 on horizontal axis. All curves are drawn upside down consistent with the experimental photographs. The vertical axes are in arbitrary units.

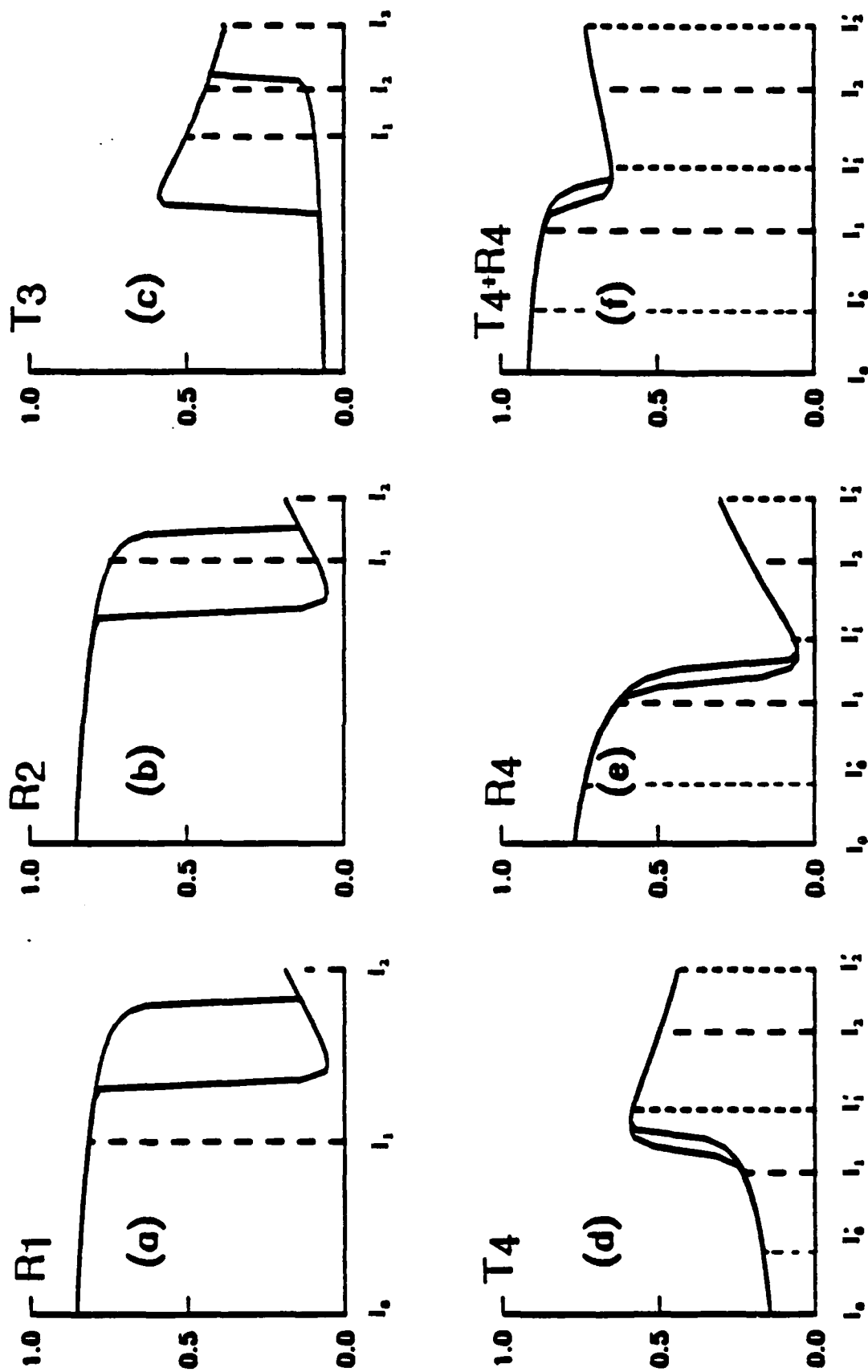


Figure 7. Computer simulated transfer functions for the ideal interference filters. The horizontal axes are input power in arbitrary units and the vertical axes are reflectivity or transmissivity of the filters. (a). Reflectivity R_1 of the comparator IF1 with $I_0=0$, $I_1=A_i$ (or B_i), and

$I_2 = A_i + B_i$; (b). Reflectivity R_2 of the latching **NAND** gate
 IF_2 with $I_1 = \bar{R}_1$, and $I_2 = \bar{R}_1 + A_i \bar{B}_i$; (c). Transmissivity
 T_3 of the latching **AND** gate with $I_1 = \bar{R}_2$, $I_2 = \bar{R}_2$
 $+ \bar{R}_1 A_i \bar{B}_i$ (or $\bar{R}_2 + \bar{A}_i B_i$), and $I_3 = \bar{R}_2 + \bar{R}_1 A_i B_i$
 $+ \bar{A}_i B_i$; (d). Transmissivity T_4 of IF_4 with $I_0 = 0$, $I'_0 = E$,
 $I_1 = A_i + B_i + E$; (e). Reflectivity R_4 of IF_4 with
 definitions as in (d); (f). The sum of (d) and (e).

$A_i \bar{B}_i$. In a similar fashion if A_i is zero and B_i is one, then the circularly polarized B_i is reflected by IF_1 , converted to the p-polarization by the quarter-wave plate and transmitted through the PBS to produce the signal $\bar{A}_i B_i$. Thus the PBS's also function as part of a bidirectional switch. Since the filter inputs are A_i and B_i and the outputs are both $\bar{A}_i B_i$ and $A_i \bar{B}_i$, IF_1 is a 4-port device.

$[\bar{R}_1 \overline{AB}]$ is the exchange prohibited signal from IF_2 , which works in reflection mode as a latching NAND gate (See Fig.7b). As long as $A_i \bar{B}_i = 0$, the reflection of $\bar{R}_1$ is high, which has been polarization rotated so it passes through PBS helping IF_3 to switch on and latch to have a high transmission when $\bar{A}_i B_i$ becomes 1 (See Fig.7c). This is the exchange situation with $E = 1$. If $A_i \bar{B}_i$ becomes 1, IF_2 switches on and latches to have a low reflectivity; the reflection of $\bar{R}_1$ will be low thereafter. If this takes place before the first occurrence of $\bar{A}_i B_i$ equal to 1, IF_3 will never have enough input power to switch on, and the output of exchange signal E will always be low (See Fig.10).

The final filter IF_4 is the exchanger and works in both transmission and reflection modes (See Figs.7d-7f). If both A_i and B_i are 0, both outputs are 0 independent of the exchange signal. In the case that $E = 0$ and only one of A_i and B_i is 1, IF_4 will not switch on; beam A_i reflects to the high output on the right; beam B_i reflects to the low output on the left. If both A_i and B_i are 1, IF_4 switches on and has a high transmissivity and low reflectivity. Both sides have a high transmission independent of the exchange status. The exchange control signal E will move the transmission curve closer to the laser frequency. When $E = 1$, either signal (or both) can switch on the gate; then A_i and B_i are transmitted to the opposite sides, in other words they are exchanged.

Fig.8 shows the results of the comparator. It demonstrates clearly that as soon as there is a difference between A_i and B_i , the comparator has a high output to the following corresponding gate which makes the appropriate decision. At the first bit, numbers A_i and B_i are equal. The output goes from a high reflection rapidly to a low reflection and produces a sharp peak pulse at the rising edge of the output. If the signal pulse width is large enough compared to the width of the sharp pulse, this sharp pulse will not have enough power and would not switch the next stage. Before each comparison of the input numbers, $\bar{R}_1$ and $\bar{R}_2$ are reset to 1. When the compare and exchange is over, $\bar{R}_1$ and $\bar{R}_2$ are shut off. The system is waiting for the next operation. Fig.9 shows the exchange prohibited signal $[\bar{R}_1 \overline{AB}]$. Upon the first occurrence of $A_i > B_i$, this signal latches to a low output. Fig.10 shows the exchange

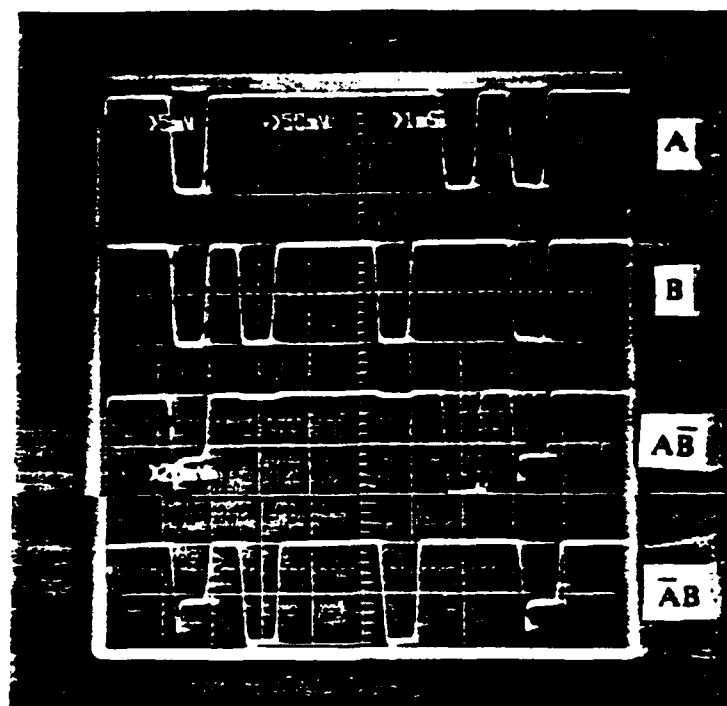


Figure 8. Experimental results of the inputs A, B and the logic outputs AB , $\overline{AB}$. The input powers are 11 mW each, and the output power is about 5 mW.

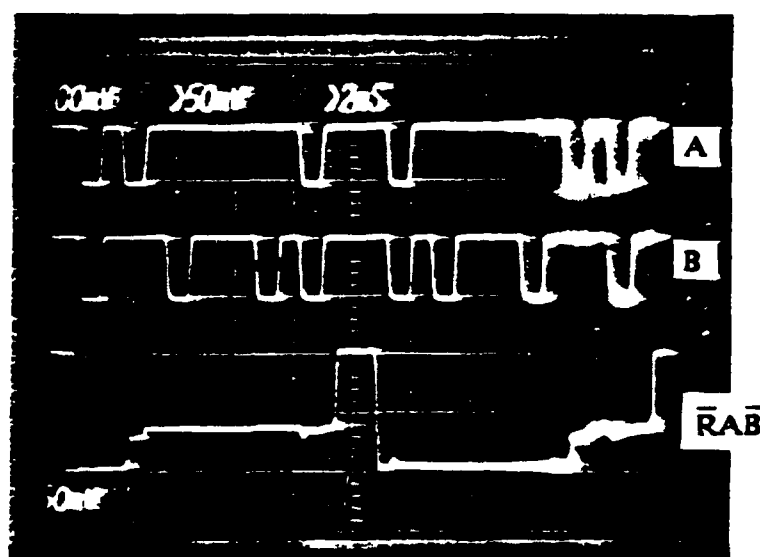


Figure 9. Output of the exchange-prohibited signal $\overline{R_1 AB}$. The upper two traces show the two groups of numbers coming into the system. In the first group A is larger than B and in the second one, B is larger than A. The power of the holding beam is 19 mW, and the output power is about 6 mW.

signal E which is the transmission of $\bar{R}_2$. On the left part of Fig.10, although two cases of $B_i > A_i$ occur, E remains in its low state because the earlier occurrence of $A_i > B_i$ latched the exchange prohibited signal to 0. On the right part of Fig.10, A is larger than B. Upon the first occurrence of $A_i > B_i$, E is latched to 1, and therefore all the following bits exchange their positions. The time delay at the rising edge of E is caused by the switching speed of the device. Figs.11 and 12 show the results of compare and exchange in the two cases of $A > B$ and $A < B$, respectively.

IV. Discussion

From the experimental results of the all-optical compare-and-exchange circuit one can see that the contrasts are not as good as those in the simulations. This is because the filters used in the experiment were not specially designed for reflection-mode operation. Therefore the low state of the reflection is higher than we expected. However even with such non-optimal filters, the system worked. The contrasts of the outputs could be better by using specially designed reflection-mode filters. This would also decrease the power required. Another bistable optical device, with its threshold set half way between the worst case levels 0 and 1, could amplify the outputs of the exchanger as well as enhance the contrast. Then the outputs could be used to drive the next compare-and-exchange module in a self-routing optical network.

It was not easy to obtain stable operations of all of the four interference filters simultaneously long enough to test the system, especially since the contrasts of the devices are not so good. The data in Figs.8-12 were taken with only the relevant section working. Figs.8-10 were taken from the compare unit consisting of IF_1 - IF_3 . And Figs.11-12 were the results from the exchange unit of IF_4 . While we were doing the exchange, we used a third beam having the power consistent with the exchange control signal E. Therefore, the experimental results do not show the delay as seen in the simulations. We also did the experiment with IF_1 and IF_3 producing a real exchange signal for the last gate to show that when $A < B$, the exchanger works.

It was also not easy to focus A, B and E onto IF_4 and have H and L come out without energy losses when the respective polarization directions are considered. A 45° Faraday rotation glass and a half-wave plate placed between the exchange control signal E and IF_4 might solve this problem. A plane-polarized light beam passing through the glass will have its polarization direction rotated through an angle θ relative to the polarization direction of the incident beam. A beam coming from the

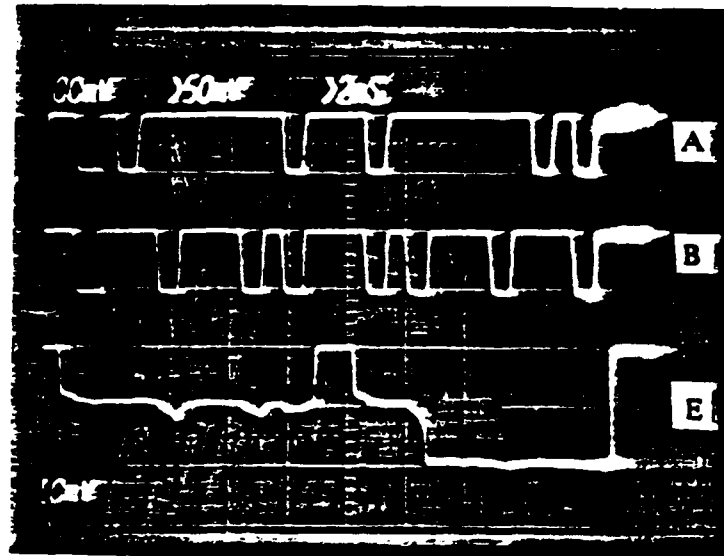


Figure 10. Output of the exchange signal E. The holding power is 20 mW, and the output power is 5 mW.

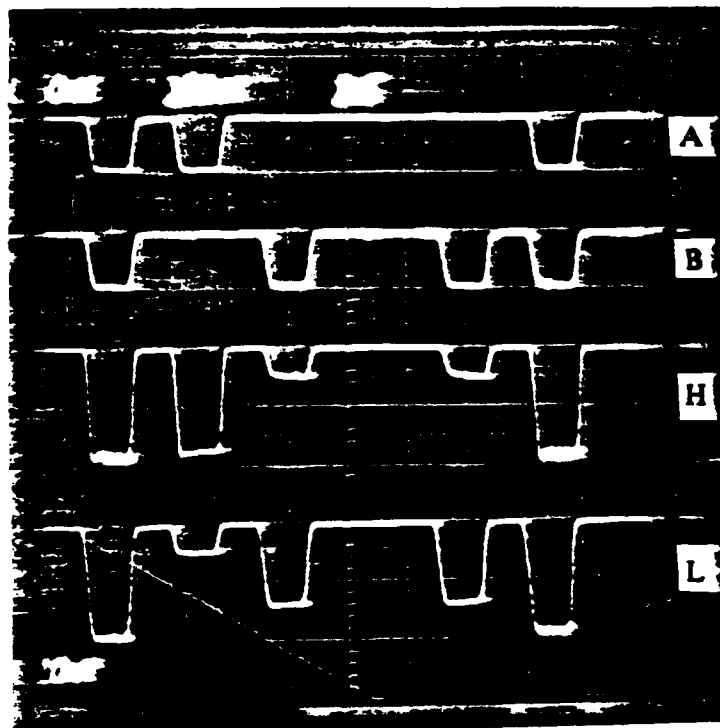


Figure 11. The high output and the low output of the system with $A > B$. The input power is 14.5 mW, and the output power is 6.5 mW.

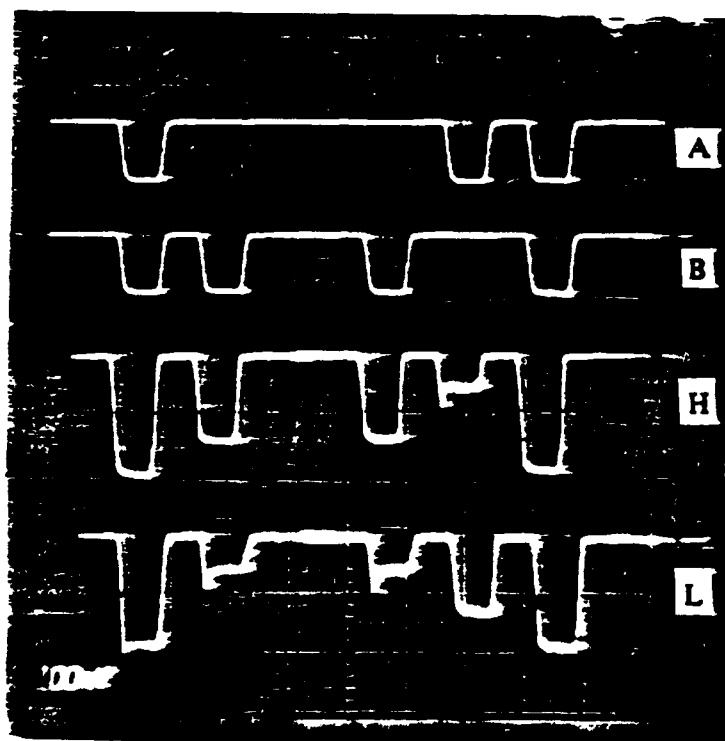


Figure 12. The high output and the low output of the system with $B > A$. The input power is 14.5 mW, and the output power is 6.5 mW.

opposite direction will have its polarization rotated in the same direction. Let θ be 45° and the fast axis of the half-wave plate be 67.5° with respect to the incident plane of vibration of E. Then the polarization direction of E propagating to the right will be rotated 90° , while that of L which propagates to the left will be unchanged. A Faraday rotator as shown in the experimental layout was not available during the experiment. Instead, only a half-wave plate was used to rotate the plane of the vibration of E by 90° and make E go to IF_4 , so that the system could work. By slightly detuning the half-wave plate, a small transmission of L is detected.

The power of the exchange control signal E in our experiment is small compared to those of numbers A and B. So the exchange operation in our case has to be active. That is when the exchanger is set to exchange status, it has to be switched on for each following bit. The operation speed is then limited by this switching speed. However, if E could be twice as large as A and B, the exchanger could operate as follows: While $E = 0$, IF_4 can not be switched on, and always keeps a high reflectivity, so that the input signals are reflected back, the exchanger operates like a mirror. While $E = 1$, IF_4 switches on and keeps a high transmissivity, so that the input signals are transmitted to the opposite sides and exchanged. The advantage is that after the exchanger is set, the following data could have a extremely high-speed transmission, since everything is linear after the exchange decision has been made .

The polarization encoding is the key to the compare-and-exchange realization. It not only reduces the energy losses in combining signals but also reduces the influences of crosstalk and feedback. The effect of the unused transmissions and reflections has been reduced to a minimum using polarization filtering. Transmissions of A and B through IF_1 and the transmission and reflection of E from IF_4 propagate back toward the source of A and B. The reflection of R_2 from IF_3 reflects directly back. Half of the high transmission of R_1 goes to IF_1 . But in this case, a decision has been made, and IF_1 is no longer used until the next operation begins. The transmission of $\bar{A}B$ through IF_3 can propagate to IF_2 which can only be high when an exchange decision is made. And half of the transmission of $A\bar{B}$ from IF_2 can propagate to IF_3 . But this can be eliminated with an additional Faraday rotator which also prevents $\bar{R}_1$ from feeding back to IF_1 .

The use of the on-axis, normal incidence makes the system extendable to operation on arrays, so that two-dimensional inputs could be compared and exchanged at the same time in parallel.

By using 2-D arrays of bistable devices and folded perfect-shuffle interconnections [18], optical sorting networks may be feasible for large numbers of channels, but first, system engineering issues must be addressed like cascadability, uniformity, crosstalk, reliability and heat dissipation.

The ZnS interference filters have relatively slow switching times (milliseconds) because they are based on thermal nonlinearities, making real-system applications unlikely. On the other hand, much faster compare-and-exchange modules based on GaAs Fabry-Perot etalons [19] may increase the throughput of the sorting networks. GaAs embodiments of the compare-and-exchange designs demonstrated here appear ideal for packet-switching telecommunication networks because GaAs etalons are diode-laser compatible [20] and allow rapid reconfiguration of very high-speed data channels.

V. Conclusions

All-optical compare and exchange has been demonstrated using bistable optical devices. The ZnS interference filters used required a speed of 3 ms per bit and a total four-filter power of about 100 mW. The experimental setup is extendable to operation on arrays and to other bistable optical devices. The switching times might be reduced to picoseconds using GaAs etalons, making the system more competitive with alternative approaches.

Acknowledgements

The authors appreciate helpful discussions with Kelvin Wagner. Special thanks to L. Wang for the phase grating.

References

- [1] D.E. Knuth, "The Art of Computer Programming: *Sorting and Searching*," Vol. 3, Addison-Wesley, Reading Mass. 1973.
- [2] S.G. Akl, "Parallel Sorting Algorithms," Academic Press, Orlando, 1985.
- [3] A. Borodin, and J.E. Hopcroft, "Routing, Merging, and Sorting on Parallel Models of Computation," *J. of Comput. and System Sciences*, vol.30, p.130, 1985.
- [4] T. Leighton, "Tight bounds on the complexity of parallel sorting," *Proc. 16th Annual ACM Symposium on the Theory of Computing*, p.71, 1984.
- [5] K.E. Batcher, "Sorting Networks and their Applications," *Proceedings of the 1968 Spring Joint Computer Conference*, vol.32, AFIPS Press, Reston Va., pp.307-314.

- [6] H.S. Stone, "Parallel Processing with the perfect shuffle," *IEEE Trans. Comput.*, vol.C-20, no.2, pp.153-161, 1971.
- [7] A.W. Lohmann, "What classical optics can do for the digital optical computer," *Appl. Opt.*, vol.25, pp.1543-1549, 1986.
- [8] J.E. Midwinter, "'Light' Electronics, Myth or Reality?," *IEE Proc.*, vol.132, Pt. J, no.6, p.371, 1985.
- [9] C.W. Stirk, R.A. Athale, and M.W. Haney, "The folded perfect shuffle optical processor," *Applied Optics*, vol. 27, No. 2, pp. 202-203 (1988).
- [10] A. Huang, "The relationship between STARLITE, a wideband digital switch and optics", *Proceedings of the International Conference on Communications*, Toronto, Canada, June 22, 1986.
- [11] H.P. Moravec, "Fully interconnecting multiple computers with pipelined sorting nets," *IEEE Trans. Comput.* vol.C-28, no.10, p.795, 1979.
- [12] W.L. Shu and A.K. Sood, "Parallel processor implementation of relational algebra operations," *Proc. of Vector and Parallel Processors in Computational Science II*, Oxford, U.K., August 1984.
- [13] C.W. Stirk, R.A. Athale, and C.B. Friedlander, "Sorting with optical compare-and-exchange modules," submitted to *Applied Optics*.
- [14] H.M. Gibbs, "Optical Bistability: *Controlling Light with Light*," Academic Press, New York, 1985.
- [15] H.M. Gibbs, and N. Peyghambarian, "Nonlinear etalons and optical computing," *Proc. SPIE*, vol.634, p.142, 1987
- [16] S.D. Smith, I. Janossy, H.A. MacKenzie, J.G.H. Mathew, J.J.E. Reid, M.R. Taghizadeh, F.A.P. Todey, and A.C. Walker, "Nonlinear optical circuit elements as logic gates for optical computers: the first digital optical circuits," *Opt. Eng.*, vol.24, p.569, 1985.
- [17] L. Wang, H.M. Chou, H.M. Gibbs, G.C. Giglioli, G. Khitrova, H.-M. Kulcke, R. Jin, H.A. Macleod, N. Peyghambarian, R.W. Sprague, and M.t. Tsao, "Symbolic substitution using ZnS interference filters," *Proc. SPIE*, vol.752, 1987.
- [18] C.W. Stirk, R.A. Athale, and C.B. Friedlander, "Optical implementation of the compare-and-exchange operation for applications in symbolic computing," *Proc. SPIE*, vol.754, p.175, 1987.
- [19] Y.H. Lee, H.M. Gibbs, J.L. Jewell, J.F. Duffy, T. Venkatesan, A.C. Gossard, W. Wiegmann, and J.H. English, "Speed and effectiveness of windowless GaAs Etalons as optical logic gates," *Appl. Phys. Lett.*, vol.49, no.1, p.486-488, 1986.

- [20] A. Migus, A. Antonetti, D. Hulin, A. Mysyrowicz, H.M. Gibbs, N. Peyghambarian, and J.L. Jewell, "One-picosecond optical NOR gate at room temperature with a GaAs-AlGaAs multiple-quantum-well nonlinear Fabry-Perot etalon," *Appl. Phys. Lett.*, vol.46, no.1, p.70, 1985.

PART IV

TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
	INTRODUCTION	1
	ABSTRACT	5
I	INTRODUCTION/BACKGROUND	6
II	CONSISTENT LABELING PROBLEM WITH BINARY CONSTRAINTS	8
	1. Graphs and Tree Structures	8
	2. Data Representation	10
	3. Forward-checking	11
III	OPTICAL MATRIX MANIPULATIONS FOR PRUNING THE SEARCH TREE	14
	1. Boolean Matrix Operations	14
	2. Optical Implementation	16
IV	DISCUSSION	17
V	ACKNOWLEDGEMENTS	19
VI	REFERENCES	19

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	Constraint Relations: Example	21
2	Initial Search Tree for Example Problem	22
3	Constraint Matrices for Example Problem	23
4	Constraint Matrices After Propagation of Initial Unary Constraints	24
5	Search Tree After Computation of Arc Consistent Network	25
6.	Constraint Matrices After Forward-checking Some Paths of Length 2	26
7	Search Tree After Forward-checking	27

APPLICATION OF OPTICS TO PROBLEMS IN
SYMBOLIC COMPUTATION

ANNUAL TECHNICAL REPORT - JANUARY 1988

INTRODUCTION

The goal of the program "Application of Optics to Problems in Symbolic Computation" at BDM is to examine the impact that optical computing may have on outstanding problems in AI. During the period covered by this report, two new efforts were undertaken that demonstrate how optical techniques can alleviate specific communication and processing bottlenecks in symbolic computation. In particular, the first effort developed the folded perfect shuffle optical processor to serve as the communication hardware for parallel processors intended to speed up the solution of AI problems. The second effort addressed the application of parallel optical Boolean matrix operations to prune the search space of AI problems that are represented by the consistent labeling formulation. In the following paragraphs, the salient features and impact of these two efforts will be discussed. The next two sections of this report contain journal articles that describe these efforts in more detail.

The perfect shuffle (PS) is an important interconnection pattern in parallel processing. It has been shown to be capable of speeding up computation of the FFT and matrix operations along with routing messages in a centralized or distributed manner for parallel computer architectures. However 2D implementations of the PS in either VLSI or printed circuit board technology, must consume at least $O(N^2/\log^2 N)$ surface area. For large N , the amount of area that the network requires can be prohibit single chip or single board implementations. Thus, large PS's must cross at least one level of the computer organization hierarchy, seriously degrading the network performance in terms of physical size, power consumption, and signal delay. Another limitation of 2D technologies is as the chip area grows, for the area-optimal layouts so does difference in lengths of the longest and shortest communication paths. This implies that there will be a size-dependent signal skew which limits the signal bandwidths of synchronous systems.

To alleviate the performance limitations imposed by 2D electronic technology, BDM developed the folded perfect shuffle optical processor. The folding strategy begins by raster encoding the 1D list into a 2D array. The shuffling operations are also transformed into 2D and applied to the 2D-formatted data. This approach uses the 2D formatting and 3D connection capabilities of optics to perform large perfect shuffles that require only $O(N)$ area; allow the per channel power to be independent of N ; have low delay; and eliminate signal skew, allowing high bandwidths. These properties will allow future parallel processors to be larger and to operate at higher speeds, significantly reducing the time it takes parallel processors to solve computationally intensive AI problems. The details of our architecture are given in the attached reprint of a paper published recently in *Applied Optics*.

THE BDM CORPORATION

The second task undertaken during this reporting period was an investigation of optical techniques for application to search problems in Artificial Intelligence (AI). The tree search or graph matching problem is ubiquitous in AI. Applications areas include: scheduling, theorem proving, and scene labeling/interpretation for computer vision. In general these problems have exponential time complexity and become intractable rapidly as the number of variables grows. A large body of research has been dedicated to developing "tree-pruning" techniques, which use forward checking to increase the efficiency of the search. These techniques attempt to avert the combinatorial explosion by using the relational constraints of the problem in local graph operations (arc and path consistency checks) to reduce the complexity of the search tree. Under worst case assumptions, forward checking itself requires exponential time; however, for many real world problems, it does increase the efficiency of the search.

A tree search can be formulated as a *consistent labeling* (CL) problem, in which the goal is to assign a *label*, from a set of L elements, to each *unit*, from a set of U elements. U corresponds to the number of levels in the search tree and L corresponds to the number of branches at each node of the tree. Not all of the L^U possible assignments are permitted by the problem constraints and the goal of a forward checking algorithm is to rule out, in advance, those partial labelings which cannot possibly contribute to a CL, where a CL is defined as a labeling of all U units in which all of the labelings are simultaneously compatible with the problem constraints.

The initial problem constraints are given as tuples of units which mutually constrain each other, along with the sets of allowed labels for each tuple. Here we restrict our attention to binary constraints. Many interesting problems in the application areas mentioned above can be cast as CL problems with binary constraints. For such problems the constraint data can be represented as $L \times L$ Boolean matrices, one for each pair of units that constrain each other.

In this task we investigated the potential for improving the efficiency of the search by applying highly parallel optical Boolean matrix operations to the set of constraint matrices. The purpose of these operations is two-fold. First, we want to remove, from the initial set of binary constraints, as many as possible of those that do not contribute to any consistent labeling. This improves the efficiency of the search by reducing the size of the domain of allowed pair labelings that must be checked during the search procedure. The second purpose in manipulating the constraint matrices is to make explicit those unary constraints that are implied by the initial set of binary constraints. These *induced* unary constraints can then be applied directly in the search process to prune the search tree. Our ideas are detailed in a preprint of a paper attached to this report which has also been submitted to the *Optical Engineering* special issue on optical computing.

Folded perfect shuffle optical processor

Charles W. Stirk, Ravindra A. Athale, and Michael W. Haney

BDM Corporation, 7915 Jones Branch Drive, McLean, Virginia 22102.

Received 25 September 1987.

Sponsored by J. W. Goodman, Stanford University.

0003-6935/88/020202-02\$02.00/0.

© 1987 Optical Society of America.

The perfect shuffle interconnection network (PS) consists of splitting a linear array of $N = 2^n$ items in half and interleaving the two halves. The PS was originally proposed as the interconnection primitive between local processors for parallel computation of the fast Fourier transform polynomial evaluation, sorting, and matrix transposition.¹ Interconnection permutations necessary for parallel matrix computations other than matrix transpose also were realized on the PS.² For the more general routing problems found in MIMD machines and telecommunications networks, an algorithm was presented to connect an arbitrary permutation of inputs to outputs with a limited number of PS stages [$O(\log N)$].³ A review of the parallel computation abilities of the PS is conducted in Ref. 4.

The wide utility of PS networks led to attempts to map the PS onto VLSI architectures.^{5,6} The inherent VLSI wiring characteristics of capacitive and inductive crosstalk, length-dependent power requirements, timing skew, limited crossovers, and chip area are ill-suited to the global and space-variant nature of the PS. Thus a trade-off of local processing complexity, the number of parallel channels and signal bandwidth due to the practical limitations of VLSI, limits the applicability of electronic PS implementations.

The ability of light beams to carry high-bandwidth data through free space without mutual coupling, the independence of drive power and interconnection length, the minimal timing skew, and the 3-D nature of optical systems led to suggestions of optical interconnects for electronic processors.⁷ The space-bandwidth product of optical systems (10^6) limits the number of parallel channels, while the modulation speed (10 GHz) and degree of multiplexing (wavelength and polarization) limit the channel bandwidth. By using optical communications the overall system throughput is increased by eliminating the interconnection network bottleneck. Hence free-space optics appears ideal for implementation of the PS in applications with high data rates or many parallel channels like telecommunications and fine-grained parallel processors.

The earliest proposed free-space optical architectures for the PS^{8,9} are capable of shuffling either the rows or columns of a matrix. If each data word is recorded as a column, shuffling between the columns results in a bit-parallel PS. In such spatially multiplexed architectures the maximum

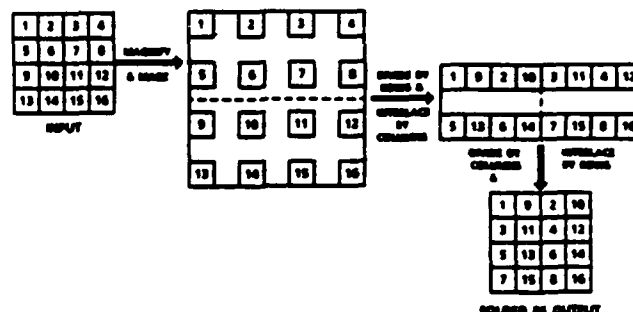


Fig. 1. Two-step approach to the folded PS.

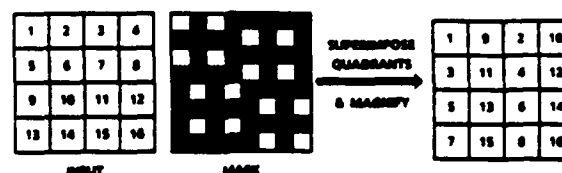


Fig. 2. One-step approach to the folded PS.

length of the data words must be known *a priori*. With time-multiplexed data streams, however, the word length is unlimited, and the other spatial dimension is free to encode additional data channels. Moreover, in many cases, the subsystems connected by the PS provide and require serial data. A 2-D optical PS was proposed¹⁰ and demonstrated¹¹ that rearranges a 2-D image of serial data streams by shuffling the rows and columns. The architecture divides and interleaves the matrix along each direction as separate operations. Since the 1- and 2-D PS are different interconnection primitives, not all the routing algorithms and hardware developed for the 1-D PS are compatible with the 2-D PS.

In this Communication we describe an algorithm that performs a 1-D PS on a long 1-D sequence that is raster-formatted on a matrix. We propose optical architectures and hardware to implement the algorithm and show experimental results for a particular system. The proposed algorithm and architectures retain compatibility with the well-developed 1-D PS algorithms while taking full advantage of the 3-D interconnection capabilities and 2-D space-bandwidth of free-space optics. Because a similar philosophy was employed in designing 2-D optical systems to perform spectrum analysis of very long (10^6) raster-recorded 1-D time signals in the work on the folded spectrum analyzer,^{12,13} we call our processor the folded perfect shuffle optical processor.

The 1-D PS algorithm is shown in Eq. (1) where N is the number of channels, i is the original data index, and i' is the shuffled index. (All indices start at 0.) Since the operations on the indices are linear and commute, they can be performed in any order or simultaneously:

$$\begin{aligned} \text{if } 0 < i < N/2 - 1, \text{ then } i' &= 2i; \\ \text{if } N/2 < i < N - 1, \text{ then } i' &= 2i + 1 - N. \end{aligned} \quad (1)$$

The details of the 1-D and 1-D folded PS algorithms differ because the data formats are fundamentally dissimilar. While the linear algorithm (and 2-D) requires the list (image) to be divided and interlaced along the same direction (both directions independently), in the folded algorithm the matrix is divided in half along the rows and interlaced along the columns. This results in a shuffled version of the input, but the rows are now twice as long and the columns half as high as those in the input matrix. Hence the output matrix must be further divided along the columns and interlaced along the rows for input/output compatibility (Fig. 1). Interchanging row and column operations in the preceding sequence produces the same folded output. These operations can also be performed simultaneously by dividing, magnifying, shifting, and interlacing the quadrants (Fig. 2). Masking is necessary for the area of the output pixels to be compatible with the input. In some architectures masking prevents pixel overlap.

Dividing a raster encoded matrix along both directions mandates that the dimensions be even. Optically dividing a matrix is possible using beam splitters, mirrors, Wollaston prisms with polarization encoding, gratings, or lenslet arrays. Addition of a constant, minus N or plus 1, is a simple shift of a quadrant in the vertical or horizontal directions, respectively. Practical shifting hardware includes tilted mirrors, off-axis lenses or lenslet arrays, prisms, or gratings. Multiplying the position by two involves imaging with 2X magnification. Figure 3 depicts a one-step architecture for the folded perfect shuffle. A simple mask at the input plane recodes the image for magnification and shifting. The four imaging lenses perform the magnification and shifting by overlapping the quadrants at the output plane. Due to the input recoding, the overlapped quadrants produce the desired PS output. The experimental results for 64-point perfect shuffles are shown in Fig. 4. The architecture is easily extendable to larger numbers of parallel channels.

In summary: We desire a folded perfect shuffle optical processor for several reasons. First, the 1-D perfect shuffle is an important interconnection primitive in communications and parallel computation. Second, formatting the 1-D data channels in two dimensions efficiently uses the 3-D interconnection capability of optics for potentially upward of 10^6 parallel channels. Moreover, parallel channels and serial data are compatible with the requirements of many preceding and subsequent information handling subsystems. Finally, the passive nature of the optical architectures presented allows shuffling times of the order of a nanosecond and are readily pipelinable. In this Communication, we described an algorithm that performs a folded perfect shuffle. We outlined the architectural and hardware approaches to implement the algorithm optically and demonstrated a particular architecture on a 64-point perfect shuffle.

This research was supported by the Advanced Research Project Agency of the Department of Defence and was monitored by the Air Force Office of Scientific Research under contract F-49620-86-C-0030.

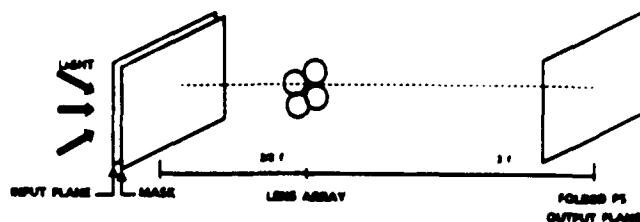


Fig. 3. One-step imaging architecture for the folded PS.

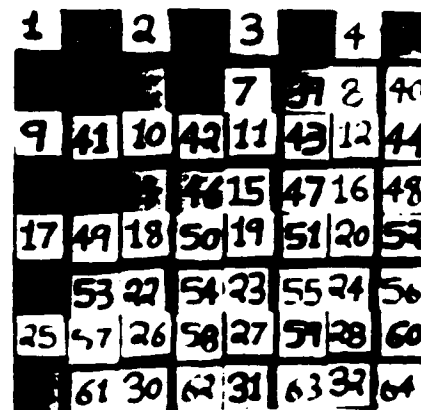


Fig. 4. Experimental results for sixty-four-channel folded PS. The variation in contrast is due to the nonuniformities in the input illumination.

References

1. H. S. Stone, "Parallel Processing with the Perfect Shuffle," *IEEE Trans. Comput.* C-20, 153 (1971).
2. D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Comput.* C-24, 1145 (1975).
3. C. L. Wu and T.-Y. Feng, "The University of the Shuffle-Exchange Network," *IEEE Trans. Comput.* C-30, 324 (1981).
4. J. T. Schwartz, "Ultracomputers," *ACM Trans. Program. Lang. Syst.* 2, 484 (1980).
5. F. T. Leighton, *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks* (MIT Press, Cambridge, 1983).
6. C. D. Thompson, "The VLSI Complexity of Sorting," *IEEE Trans. Comput.* C-32, 1171 (1983).
7. J. W. Goodman, F. J. Leonberger, S.-Y. Kung, and R. A. Athale, "Optical Interconnections for VLSI Systems," *Proc. IEEE* 72, 850 (1984).
8. A. Lohmann, W. Stork, and G. Stucke, "Optical Implementation of the Perfect Shuffle," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1985), paper WA3.
9. J. E. Midwinter, "Light Electronics, Myth or Reality?," *IEE Proc* 132, Pt. J, No. 6, 371 (1985).
10. A. Lohmann, "What Classical Optics can do for the Digital Optical Computer," *Appl. Opt.* 25, 1543 (1985).
11. S.-H. Lin, T. F. Krile, and J. F. Walkup, "2-D Optical Multistage Interconnection Networks," *Proc. Soc. Photo-Opt. Instrum. Eng.* 752, 209 (1987).
12. C. E. Thomas, "Optical Spectrum Analysis of Large Space Bandwidth Signals," *Appl. Opt.* 5, 1782 (1966).
13. T. M. Turpin, "Spectrum Analysis Using Optical Processing," *Proc. IEEE* 69, 79 (1981).

Optical Techniques for Increasing the Efficiency of Heuristic Search

Michael W. Haney, Ravindra A. Athale, and Roger A. Geesey

**BDM Corporation
7915 Jones Branch Drive
McLean, VA, 22102**

ABSTRACT

Many problems in Artificial Intelligence are intractable due to the exponential growth of the solution space with problem size. Often these problems can benefit from heuristic search or forward-checking techniques which attempt to prune the search space down to a manageable size before or during the actual search procedure. Many interesting search problems can be formulated as consistent labeling problems in which the initial problem information is given in the form of a set of binary constraints, for which Boolean matrices are a natural data representation. In this paper optical implementations of Boolean matrix operations are proposed for manipulating the constraint matrices to perform forward-checking and thereby increase the search efficiency. The high degree of parallelism afforded by using optical techniques and the relatively low accuracy requirements of Boolean matrix operations suggest that optical techniques are well matched to this problem.

I. Introduction/Background

Problems that require searching through very large solution spaces are ubiquitous in Artificial Intelligence (AI). Examples can be found in: expert systems, scheduling, theorem proving, database management, game playing, decoding, and computer vision. In general these problems have exponential computational complexity and solutions based on exhaustive search are impractical. These problems are characterized by a lack of structure in the solution space which precludes algorithmic solutions. Heuristic search strategies are therefore envisioned as the only hope for attacking these problems.

Many AI problems can be formulated as consistent labeling (CL) problems [1], in which the goal is to assign a label, from a set of L possible labels, to each unit, from a set of U units, while satisfying all the known constraints of the problem. The initial problem constraints are given as a set T of n -tuples of units which mutually constrain each other, along with a set R of $2n$ -tuples which list the allowed label assignments for each n -tuple in T . The constraint relations defined by T and R can, in general, consist of unary-, binary-, ternary-, ...- relations, depending on the number of elements in the n -tuples.

To make these notions less abstract, it is appropriate to describe the CL formulation in terms of a real-world example. Consider the problem of scheduling N speakers (denoted: $a, b, c, \dots$) into N timeslots (denoted: $1, 2, 3, \dots, N$). The set of units corresponds to the various timeslots and the set of possible labels for those timeslots corresponds to the set of

speakers. An example of a given unary constraint is: "Speaker 'a' is unavailable for timeslots 20 through 30". In the CL formulation this corresponds to having the set of individual units being self-constraining as 1-tuple members of the set T. The corresponding members of the set R are of the form: $(1,a)$, $(2,a), \dots, (19,a)$, $(31,a), \dots, (N,a)$. This set corresponds to the allowed label assignments taking into account the known unary constraint. An example of a binary constraint is: "Speakers 'b' and 'c' should not be scheduled back-to-back". The corresponding members of the set T are then all pairs of units (timeslots) that are in sequence, i.e., $(1,2)$, $(2,3), \dots, (N-1,N)$, and the corresponding members of the set R are of the form: $(1,i;2,j)$, $(2,i;3,j), \dots, (N-1,i;N,j)$, such that i and j are not simultaneously b and c. An example of a possible ternary constraint is: "Speakers 'a', 'e', and 'f' should be scheduled in sequence;" the appropriate members of T and R can be listed in an analogous manner to the binary constraints.

In this paper we restrict our attention to binary constraints -- the units directly constrain each other only in a pairwise manner. Many interesting problems in the application areas mentioned above can be cast as CL problems with binary constraints. Examples of such problems include the Satisfiability problem, which is an archtypical NP-complete problem related to theorem proving, and scene labeling in computer vision, among others [3].

II. Consistent Labeling Problem with Binary Constraints

1. Graphs and Tree Structures

When cast as a directed graph problem, the nodes of the graph correspond to the units and the connecting arcs of the graph correspond to relations between the nodes that define the allowed labels for each node. The search process consists of moving from node to node in the graph, assigning labels to each unit (node), and checking to determine whether the label assignments made so far are consistent with the given problem constraints.

Figure 1 illustrates an example of a binary constraint search problem involving 5 units and 3 labels. The initial set of pairs of units which constrain each other is given in set T and the allowed set of labels for these constraining pairs is given in the set R. Since only binary constraints are considered, the graph representation in Figure 1 has arcs connecting nodes in a pairwise manner. Furthermore, we assume for this simple example that the order of the units in the constraining pairs does not matter. In this case the graph representation is an undirected graph as shown.

The recursive nature of the graph search process is often represented as a tree structure to permit the time history of the search process to be preserved. For the CL problem formulation, U , the number of units, corresponds to the number of levels in the search tree, and L , the number of candidate labels, corresponds to the number of branches at each node of the tree. Not all of the L^U possible assignments are permitted by the

problem constraints and the goal of the search is to find which (if any) of the possible labelings are all simultaneously compatible with the problem constraints. Such a labeling is called a Consistent Labeling. The initial search tree associated with the problem depicted in Figure 1 is shown in Figure 2. It can be seen that even such a simple problem has $L^U = 243$ possible paths that might need to be checked for a consistent labeling in a brute-force search, revealing the combinatorial explosion.

In a standard back-tracking tree search procedure a trial label is assigned to the first level (unit) of the tree and checked to see if it is consistent with the given constraints. If the label is not consistent a new label is tried until one is found that is allowed. When a permitted label is found, a trial label is assigned to the second level of the tree, and the trial pair of labelings is checked for consistency with the given constraints. This procedure is maintained down through successive levels of the tree until either a CL (i.e., an allowed path) is found or a unit is discovered for which no allowed label can be found. In the latter case the procedure must back up one level of the tree (i.e., to the previous unit) and find another consistent label before continuing. If no label can be found at that level then the search must back up one more level until an allowed labeling is found that is different from the one that caused the procedure to halt and backtrack. If the search is forced to backtrack all the way to the first level without finding a CL then no CL exists. The problem with this procedure

is that it suffers from "thrashing" behavior [2] -- a trial labeling at a high level in the tree which is not part of a CL may not be discovered without checking down many paths and backtracking many times.

A large body of research has been dedicated to developing "tree-pruning" techniques, which use forward checking to forstall thrashing and increase the efficiency of the search. These techniques attempt to avert the combinatorial explosion by using the relational constraints of the problem in local graph operations (arc and path consistency checks) to reduce the complexity of the search tree. Under worst case assumptions, forward checking itself requires exponential time; however, for many real world problems, it does increase the efficiency of the search [3].

2. Data Representation

For problems with binary constraints the relation data can be represented as $L \times L$ Boolean matrices, $R(i,j)$, one for each pair of units (i,j) that constrain each other [2]. The L rows correspond to the labels of unit i and the L columns correspond to the labels of unit j . The presence of a "1" as the k th element of matrix $R(i,j)$ indicates that assigning the i th unit with label "k" is consistent with labeling the j th unit "1". Note that if two units are not given to constrain each other directly, then the initial constraint matrix corresponding to that pair would consists of all 1's and contains no useful information about how that pair of units might ultimately constrain each other through induced constraints. Figure 3 shows

the seven binary constraint matrices corresponding to the given allowed labelings in the example problem of Figure 1. Since the ordering of the constraining pairs is not important in this problem it is noted that $R(i,j) = R^T(j,i)$, where T indicates transpose of the matrix.

3. Forward-checking

In general, the set of given constraints is very large. For those search problems for which there are only a small number of possible CLs, this means that many of the initially allowed label assignments are superfluous -- they do not contribute to any of the CLs. Indeed, a large portion of the inefficiency in brute-force search strategies can be attributed to the repeated checks of these extraneous constraint relations during the search. An important objective of a forward-looking procedure is therefore to discard as many as possible of those constraints that play no role in a final solution [1], thereby avoiding the inefficiency of repeatedly checking them during the search.

Various levels of forward-checking can be achieved, corresponding to looking ahead to various levels in the search tree. An arc of the graph is consistent if the set of allowed labelings for the pair of nodes (units) connected by the arc is also consistent with the allowed labelings of all other pairs of nodes that contain one of the two initial nodes. If all arcs of the graph are consistent, then the arc consistent network [2] has been computed, i.e., we have looked-ahead to all paths of length 1 from each node of the graph. This concept can be generalized to paths of length 2 and higher [2].

In the next section we discuss the application of highly parallel optical Boolean matrix operations to the set of constraint matrices to perform arc and path consistency checks. The purpose of these operations is two-fold. First, as stated earlier, we want to remove, from the initial set of binary constraints, as many as possible of those constraints that do not contribute to any consistent labeling. This improves the efficiency of the search by reducing the size of the domain of allowed pair labelings that must be checked during the search procedure. The second purpose in manipulating the constraint matrices in forward checking operations is to make explicit those unary constraints (i.e., unit u_i cannot assume label l_k) that are implied by the initial set of binary constraints. The derivation of a unary constraint from the given binary constraints constitutes a significant reduction in the size of the search space. Each induced unary constraint for a unit u reduces the size of the search space by a factor of $1 - L_u/L$, where L_u is the current number of labels for unit u found to be not permitted by induced unary constraints, and L is the total number of labels. The induced unary constraints can thus be applied directly to the search tree.

As the forward-checking process proceeds, the constraint matrices become more sparse due to the deletion of extraneous constraints. Eventually, the sparsity of the matrices may become such that an entire row or column contains only zeros. This situation indicates an induced unary constraint. The operation of unary constraint detection is accomplished by examining each

of the current set of binary constraint matrices for the presence of a row or column containing only 0's. This situation indicates that the unit associated with the rows or columns of that matrix can never be assigned the label corresponding to that row or column. This can be detected by performing an OR operation across all rows, and then all columns, for each of the constraint matrices. If an all-zero row or column is found, then the resulting induced unary constraint can be propagated to all other constraint matrices that share the same unit by zeroing out the associated row or column associated with that label and unit. This may lead to the discovery of new induced unary constraints which can be detected and propagated until a fixed point is reached.

The first step in a breadth-first forward-checking procedure is to check for unary constraints that might be already present in the initial binary constraints. Examination of the matrices for the sample problem in Figure 3 reveals that $R(2,3)$ has a row and a column that consists of all zeros. The all-zero row corresponds to the unary constraint: unit 2 cannot have label b, and the all-zero column corresponds to the unary constraint: unit 3 cannot have label c. Propagation of these unary constraints results in the zeroing out of the third column of $R(1,3)$ and the third row of $R(3,4)$. The new set of reduced constraint matrices is shown in Figure 4. Applying the unary constraints to the search tree reduces the number of possible paths that might possibly need to be checked to 108 as shown in Figure 5.

The detection and propagation of all unary constraints hidden in the initial set of binary constraints corresponds to the computation of the arc consistent network for the problem. To achieve further benefit from forward-checking requires the use of path consistency checks for paths of length 2 or greater.

III. Optical Matrix Manipulations for Pruning the Search Tree

1. Boolean Matrix Operations

The Boolean matrix operations of intersection and composition have previously been suggested for use in forward checking [2]. Intersection is the element by element ANDing of corresponding entries of two matrices to yield a new matrix. Composition is a Boolean matrix multiplication which is obtained by replacing the multiplication and addition operations in conventional matrix multiplication by Boolean AND and OR operations, respectively. Since the multiplication between two binary elements is equivalent to their multiplication, the only operational difference is in the generation of binary output elements by replacing the analog addition with multi-input OR operation. Here we propose that these operations can be combined with unary constraint detection and unary constraint propagation, to compute path consistent networks which will increase the efficiency of the search. Furthermore, for increased speed, all of these operations can be implemented optically, using established low accuracy analog linear algebraic techniques, followed by simple nonlinearities at the detector stage to implement the OR operation.

The use of intersection and composition in a forward checking operation proceeds as follows. Given a constraint matrix relating units i and j , $R(i,j)$, and other constraint matrices $R(i,k)$ and $R(k,j)$, we can create a new constraint matrix:

$$R'(i,j) = R(i,j) * R(i,k) \& R(k,j), \quad (1)$$

where "*" indicates the intersection operation and "&" indicates the composition operation. Composition takes precedence over intersection. The induced relation R' replaces R and is a stronger constraint between units i and j because it now takes into account the influence of an intermediate unit (k) along the path and not just the single arc between the units. Even stronger constraints can be derived by intersecting $R'(i,j)$ with other induced constraint matrices derived from the composition of matrices along other paths between i and j . In practice, this operation would be performed on all constraining pairs of units to some level of path length.

In the example problem, the application of Equation 1 to compute new versions of $R(1,4)$ and $R(1,5)$, as shown in Equations 2 and 3:

$$R'(1,4) = R(1,4) * R(1,3) \& R(3,4), \quad (2)$$

$$R'(1,5) = R(1,5) * R(1,4) \& R(4,5), \quad (3)$$

These path consistency checks yield new unary constraints which, after propagation, result in the set of constraint matrices shown in Figure 6. With the new induced unary constraints (unit 1 cannot be labeled "b", unit 5 cannot be labeled "a") the search tree is now pruned such that only 48 paths remain to be checked in a search procedure, as shown in Figure 7. A check of these paths shows that there are 3 CLs for the example problem: (1a,2a,3a,4b,5b), (1a,2c,3a,4b,5b), and (1c,2c,3b,4a,5c).

2. Optical Implementation

Optical implementation of the operations of intersection and composition has previously been suggested in the context of inference machines [5]. The operation of intersection can be implemented optically via image multiplication by representing the constraint matrices as binary images, while composition can be implemented by analog optical matrix multiplication, followed by thresholding to restore the levels to 1 or 0.

Unary constraint detection is achieved by focusing the light passed through the matrices along each dimension separately and detecting the presence of light with a 1-D threshold detector array. This achieves the required OR operation across all rows or columns simultaneously. To propagate the induced unary constraint the resulting thresholded 1-D data, $R(i)$, is transformed into a light signal, spread out into a 2-D array, and multiplied by all other matrices, $R(i,m)$, which share the unit that has the unary constraint. For constraint matrices which involve the unit i as the column index, the transpose, $R^T(m,i)$ is used.

IV. Discussion

The forward-checking process could conceivably continue until all the superfluous binary constraints and resulting unary constraints are discovered. At this point the minimal network [2] has been computed, which means that every remaining binary labeling in the constraint relations is a part of at least one CL and forward checking will not reduce the set of binary constraints any further. It has been shown that the closer a network is to minimal then the less work the tree search will have [3]. However, computation of the minimal network for a general combinatorial problem is itself an NP-complete problem [4]. In fact forward-checking is not guaranteed to reduce the computational requirements of the search. However, some problems have proven to benefit greatly from the application of forward-checking [2].

Rather than carrying out the entire search procedure using optical forward-checking techniques, it is envisioned that the application of optical techniques for forward-checking is appropriate as a pre-filtering operation to remove as many of the superfluous binary constraints as possible and prune the search tree down as much as possible before turning the problem over to a conventional Prolog-type processor to complete the search. In this context, optical techniques would be applied to those computationally intensive forward-checking operations of the search that are amenable to highly parallel optical techniques, while the control intensive and sequential operations of the

search would be left to the more conventional electronic techniques.

The extent to which optical forward-checking techniques should be employed will depend on the size and nature of the problem and the heuristic search strategy used. Current conventional wisdom assumes that Boolean constraint matrices as large as 1000 x 1000 pixels can be handled optically. This would seem to suggest that an upper limit on the number of labels in the problem would be on the order of 1000. However, this limit may not be correct -- the large matrix operations can be decomposed into a set of smaller ones without loss of computational efficiency. The number of units that can be handled with optical techniques appears to be limited principally by the amount of memory needed to store the large number of constraint matrices -- the more units in a problem, then the more likely a binary constraint relation exists between a pair of units.

The number of unary constraints that can be detected and propagated, and hence the importance of these operations to the given problem, is determined by the number of CLs that are possible (something you may not have a handle on ahead of time) and the number of labels and units for the problem. Clearly if only one CL exists then the number of unary constraints that may be detected through forward-checking operations grows with the size of the sets of units and initial candidate labels. At the other extreme, if the number of CLs is large and the number of candidate labels is low, then it is possible for there to be no

unary constraints in the problem, which simply means that each units is allowed to be assigned each of the labels for at least one CL.

Although optical implementations of the Boolean matrix operations and manipulations on the set of constraint matrices offer the potential for great speed due to the parallel nature of the computations, the overall speed of a hybrid architecture may be limited due to bottlenecks associated with the required control and memory manipulation functions. Effective use of optical matrix operations may require an architecture which has a very limited amount of transductions between the optical and electronic domains to avoid the associated bottlenecks. In fact, the best approaches may keep the constraint data in an optical matrix format throughout the forward-checking procedure, and return to the electronic domain only during the actual search process. The ultimate utility of optical forward-checking techniques will be determined by the progress made in optical mass storage technology.

V. Acknowledgments

This work was supported by DARPA/DSO and monitored by Air Force Office of Scientific Research.

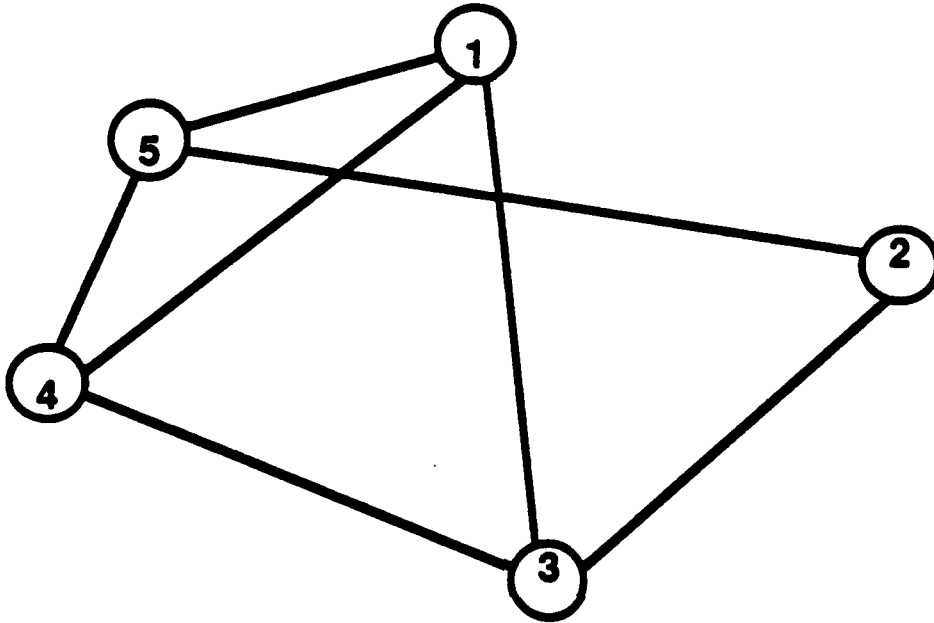
VI. References

- [1] Haralick, R. M. and Shapiro, L. G., IEEE Trans. PAMI, Vol. PAMI-2. No. 3, May 1980, pp. 193-203.

- [2] Mackworth, A. K., Artificial Intelligence 8 (1977), pp. 99-118.
- [3] Haralick, R. M. and Shapiro, L. G., IEEE Trans. PAMI, Vol. PAMI-1. No. 2, April 1979, pp. 173-183.
- [4] Montanari, U., Information Sciences 7 (1974), pp. 727-732
- [5] Caulfield, H. J., Optics Communications, Vol. 55, No. 4, Sept., 1985, pp. 259-260.

U: {1,2,3,4,5} L: {a,b,c}

T: {(1,3),(1,4),(2,3),(3,4),(4,5),(1,5),(2,5)}



R:

(1a,3a)	(1a,4a)	(2a,3a)	(3a,4b)	(4a,5a)	(1b,5a)	(2a,5b)
(1a,3b)	(1a,4b)	(2c,3a)	(3b,4a)	(4b,5b)	(1a,5b)	(2c,5b)
(1b,3b)	(1b,4b)	(2c,3b)	(3b,4c)	(4a,5c)	(1c,5c)	(2c,5c)
(1c,3c)	(1c,4a)		(3c,4a)	(4c,5b)		(2b,5c)
(1c,3b)	(1c,4c)		(3c,4c)			(2b,5b)
	(1b,4c)					(2b,5a)

Figure 1. Constraint Relations: Example

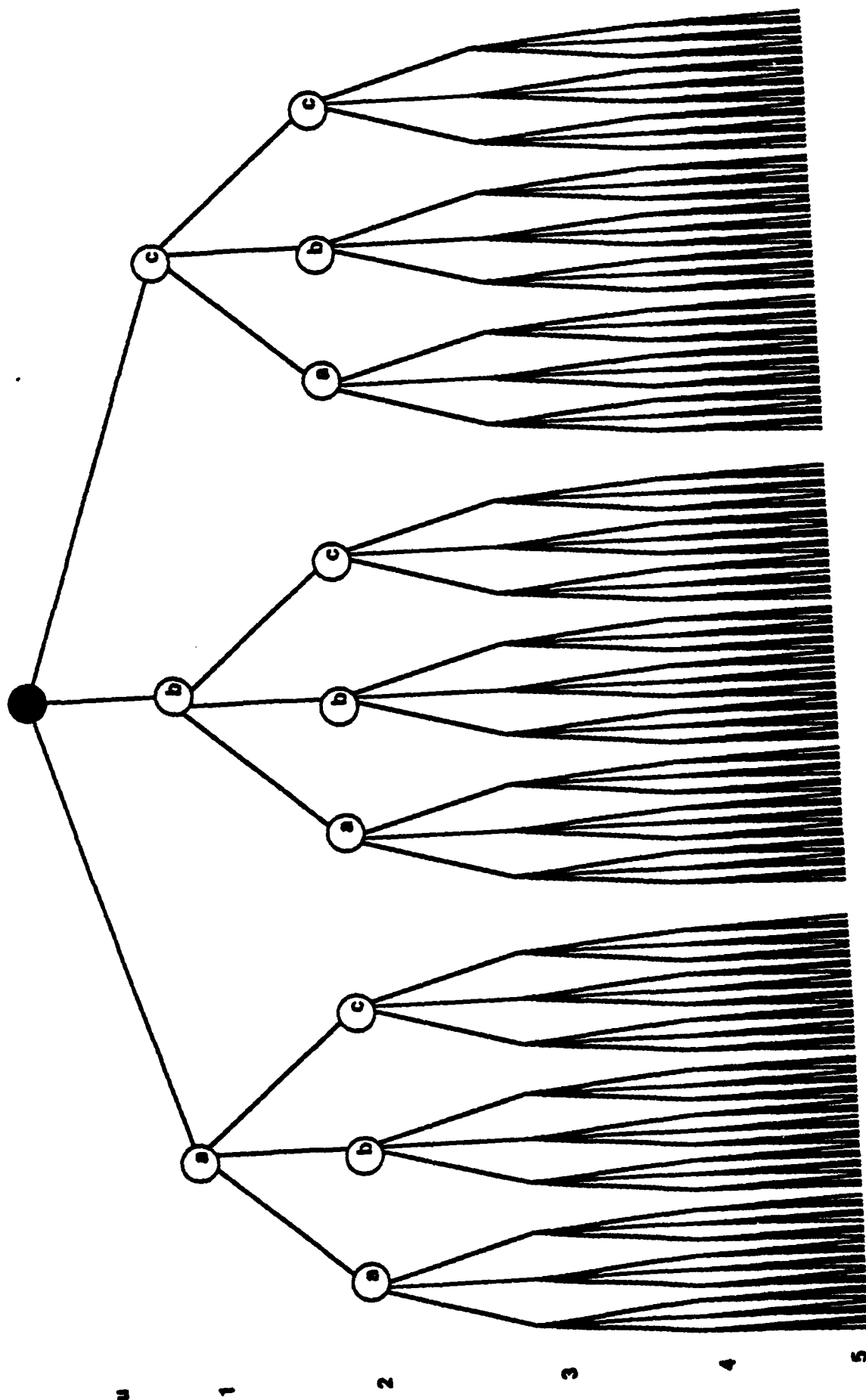


Figure 2. Initial Search Tree for Example Problem

$$R(1,3) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$R(1,4) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$R(2,3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$$R(3,4) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$R(4,5) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$R(1,5) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R(2,5) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Figure 3. Constraint Matrices for Example Problem.

$$R(1,3)' = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$R(1,4) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$R(2,3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$$R(3,4) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$R(4,5) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$R(1,5) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R(2,5) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Figure 4. Constraint Matrices After Propagation of Initial Unary Constraints

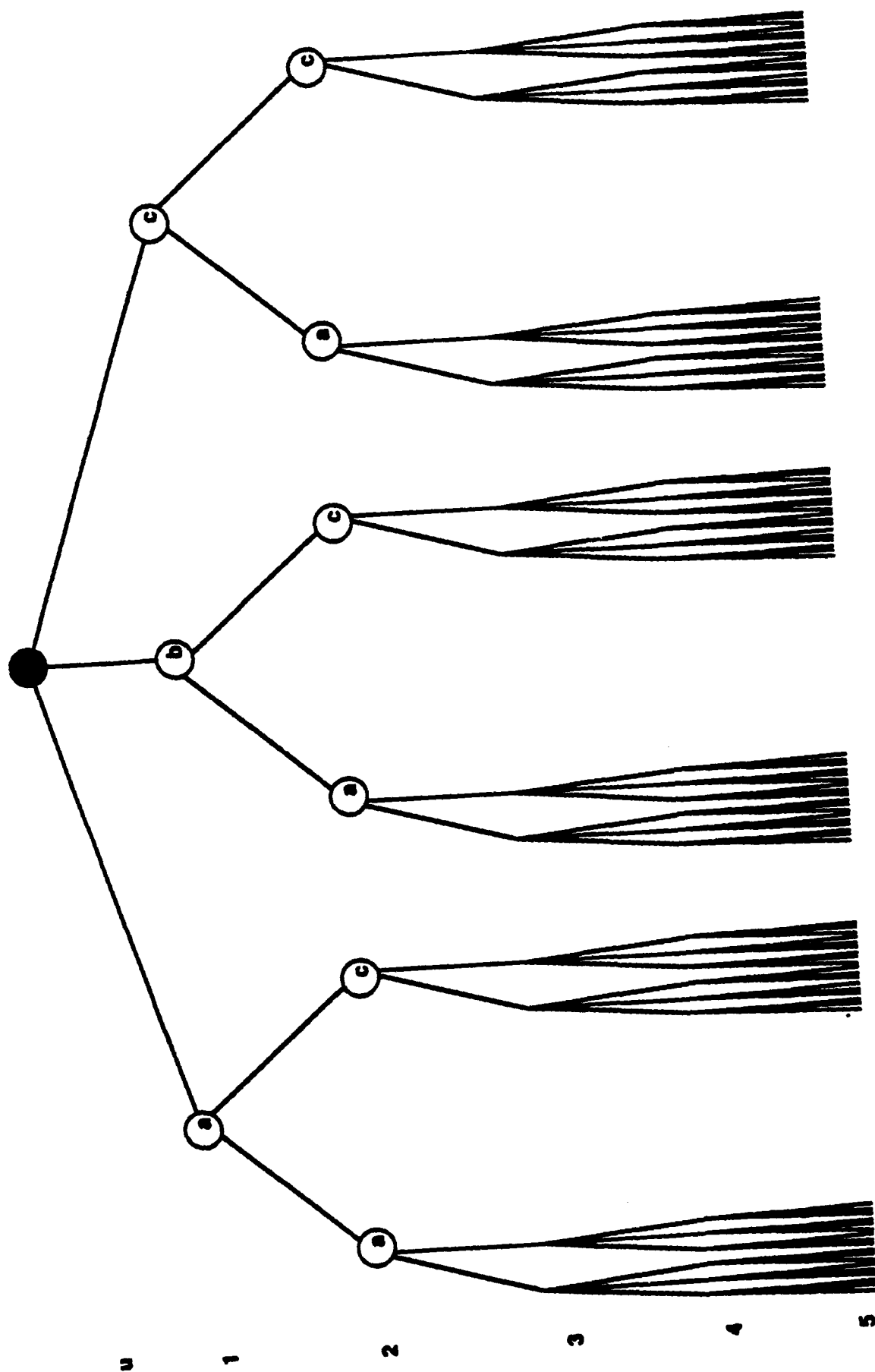


Figure 5. Search Tree After Computation of Arc Consistent Network

$$\begin{aligned}
R(1,3) &= \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \\
R(1,4) &= \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \\
R(2,3) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \\
R(3,4) &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\
R(4,5) &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \\
R(1,5) &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \\
R(2,5) &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}
\end{aligned}$$

Figure 6. Constraint Matrices After Forward-checking Some Paths of Length 2

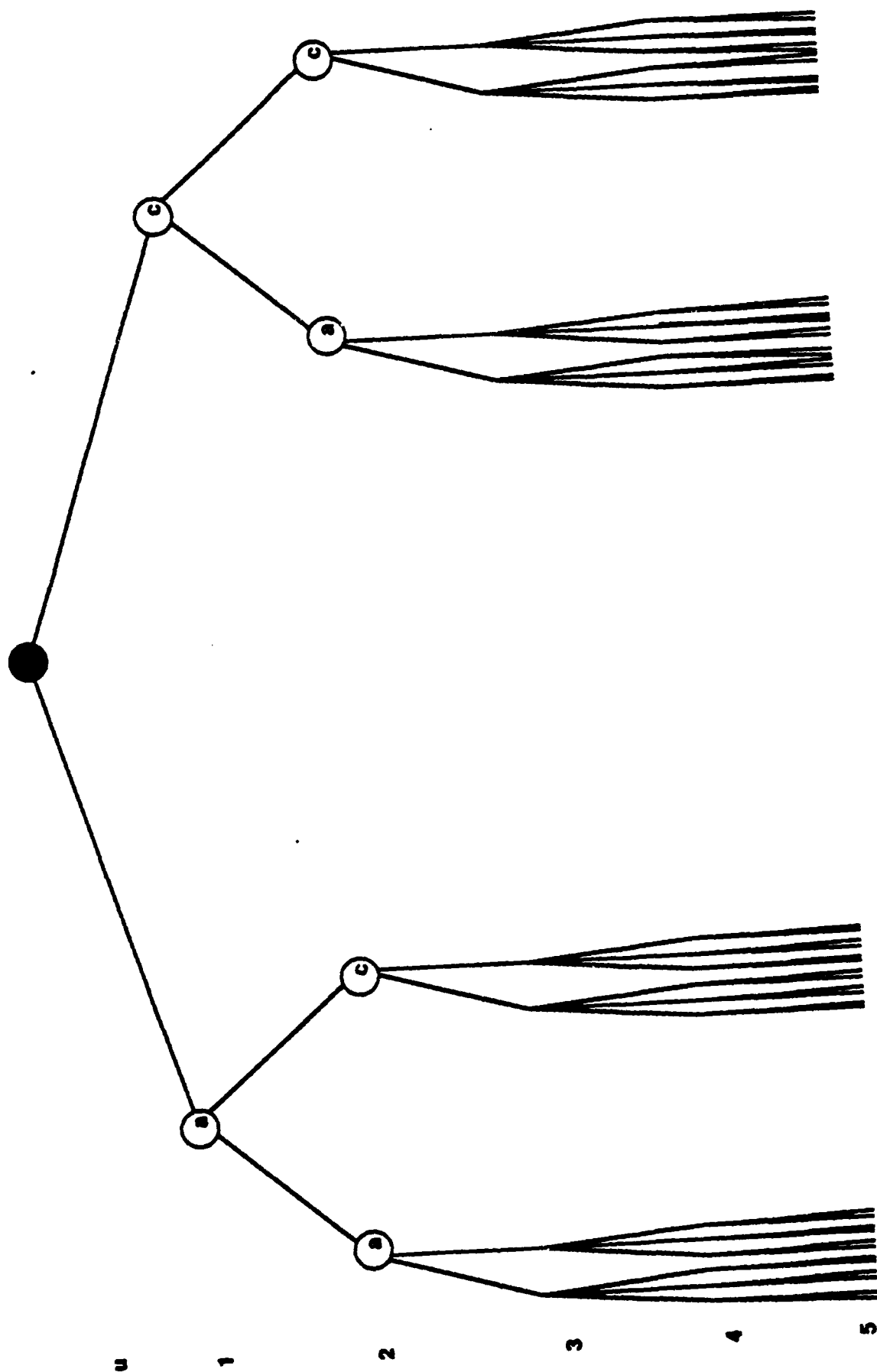


Figure 7. Search Tree After Forward-checking

PART V

FOLDED OPTICAL INTERCONNECTION NETWORKS

I. Introduction

A. Background

The spectacular advances in the semiconductor electronic technology since the invention of integrated circuits (ICs) has fueled corresponding advances in the computer industry. This interaction is evident in supercomputers (e.g., CRAYs) as well as in constantly decreasing price/performance ratios for smaller systems (e.g., Personal Computers). In seeking ever expanding computing power, computer scientists are also investigating parallel and distributed processing systems. Several parallel computing systems have been designed, demonstrated, and even manufactured over the past decade. The number of processing elements (PEs) and their complexity, the associated memory organization, the control strategies of the instruction streams, and the interconnection topologies employed vary significantly, thus leading to distinct architectural designs. The parallel architecture design tradeoffs are dependent on the current state of technology, on one hand, and the application requirements on the other, and hence are constantly changing. The advent of VLSI has dramatically lowered the cost of PEs and semiconductor memory, thereby making practical massively parallel designs, such as MPP made by Goodyear Aerospace (16K PEs) and the Connection Machine made by Thinking Machines Corp. (64K PEs). The performance of the parallel machines as well as their domain of applicability is, however, equally determined by the topology and performance of the interconnections within the machine. This central role of interconnects in parallel architectures is reflected in the amount of research on this topic.¹

B. Electronic Interconnects:

At present, the communication between different parts of a parallel computer is almost totally electronic, due to the direct compatibility between data representation (voltage signals) in

the logic and memory circuits and in the electronic conductor interconnects. The integrated circuits technology miniaturizes the interconnections along with the logic circuits. Use of CAD design tools simplifies the chip and printed circuit (PC) board layout. Automated manufacturing techniques are also routinely used at the chip and board level. The interconnect media, such as wires, strip lines, coaxial cables and corresponding connector technology have attained a high level maturity and are easily handled in production environments.

On the other hand, fundamental features of electrical signal propagation and interaction constrain the parallel computer architecture design. Because information is carried by charged particles, capacitive and inductive coupling, between closely spaced electrical channels carrying high bandwidth information, are everpresent physical phenomena. The density and bandwidth of electronic communication channels are therefore fundamentally limited. Also, the clock skew between signals arriving along different paths constrains interconnection topologies as well as the system clock speed. Electrical transmission lines have to be terminated if undesirable reflections at the end point are to be avoided. This termination increases the drive power requirements which consequently uses valuable chip area in driver circuitry.²

C. Optical Interconnects:

Optical techniques for long distance telecommunication are now firmly entrenched to the extent that it is now exclusively used for between city links. Fiber optic links can now be designed to communicate high bandwidth (Gbps) information over long distances (50 km) without repeaters. Fiber optic cables are also small and light weight. The optoelectronic components required in the transmitters, receivers, switches, multiplexers/demultiplexers for the fiber optic links are now readily available with required operational characteristics. These developments, combined with a growing realization of the fundamental disadvantages of electronic interconnects has spurred

a great deal of research in optical interconnects for intracomputer communications.³ Because lightwaves are chargeless and propagate in non-conductive media, the use of optics for intracomputer communication represents a fundamentally different approach to the interconnect problem and requires a reevaluation of the design tradeoffs.

Intracomputer communication can be divided into a hierarchy that is based on physical aspects rather than functional aspects of the electronic system. An electronic computing system consists of:

- (1) several digital circuits fabricated on a chip,
- (2) several chips connected on a PC board,
- (3) several PC boards connected to a common back plane in a rack or a card cage
- (4) several cabinets interconnected by cables or ribbons.

Functionally, a single PE could be as small as 1/10 of a chip or as large as a cabinet. The nature of signals between these physical components will therefore depend on the functions that are incorporated at that level. The performance of the interconnects will, however, be determined by the physical hierarchy (intra-chip, inter-chip, board-to-board).

Optical interconnects are being investigated for applications at all levels of interconnect hierarchy.² The limitations of electronic technology as well as the maturity of relevant optoelectronic technology depend very strongly on the physical level of interconnects. Hence the motivation and the specific technical approach adopted for optical interconnects will also depend on the physical level.

Intra-chip optical interconnects depend on integrated arrays of optical sources/modulators and receivers that are process-

compatible with Si or GaAs ICs. These are not yet available with required performance. In addition, the limitations of electronic interconnects at the intra-chip level become serious only when the interconnect lengths become longer (several mm). By clever circuit design and layout techniques, this limitation can be circumvented for many circuits thus obviating the need for optical interconnects at this level. Some inherently global functions, like clock distribution can potentially benefit from optical interconnects and this particular application has been investigated recently.⁴

Inter-chip communication in conventional IC packages takes place primarily along the edge of the chip. This leads to a pin-out limitation that becomes worse as the chips become bigger (the ratio of periphery/area of the chip decreases). Optical interconnects can potentially overcome this limitation by providing input/output ports on the chip surface. The relevant optoelectronic technologies are, however, immature as noted in the previous paragraph. Another approach to inter-chip optical interconnects is based multiplexing several electrical channels on a single high bandwidth optical link in order to reduce the number physical connections from a chip. The transmitters, receivers and multiplexers can be placed on a separate chip with fly-wire bonding between the electronic and the optoelectronic chips.⁵ For this approach, the packaging becomes a critical issue and is being addressed by a number of laboratories including MIT/Lincoln Laboratories.

The board-to-board interconnects are not amenable to automated design or manufacturing and are largely done manually. A powerful machine like a CRAY has thousands of wires that have to be hand trimmed to proper length to avoid clock skew and hand connected to the proper destination. Optical interconnects between boards can be implemented with fiber optic links or by employing free-space optical links. Several electrical signals can be multiplexed on a single high speed fiber link, thereby reducing the severity of the mechanical problem. Fiber links are

also immune to electromagnetic interference, cross-talk and ground loop problems that commonly plague twisted-pair links. The optoelectronic transmitters, receivers and drivers no longer have to be integrated and the highly developed discrete optoelectronic component technology can be exploited. This approach has been adopted in a project undertaken at Honeywell to insert board-to-board fiber optic links in the Connection Machine to improve its communication bandwidth and scalability. Another approach to optical board-to-board interconnects is based on free-space propagation of light. This approach allows direct communication between chips located in the interior of the two boards without going through the backplane. Multiplexing techniques can also be used to reduce the number of optical links. Standard optical transmitters and receivers in discrete device form can be employed. The primary advantage of this approach is scalability. A simple imaging system can easily handle 1000 optical channels and the complexity of the communication link (free-space, in this case) is independent of the number of channels. This is to be contrasted with fiber optic or twisted-pair communication links, where the complexity grows with the number of channels. The issues of mechanical stability and alignment requirements for board-to-board free-space optical point-to-point links are being addressed in a research project at MIT/Lincoln Labs.⁶ These issues are amenable to standard optical engineering solutions. Since a board can accommodate a large number of chips, a large range of PE complexity and numbers can be accommodated by this approach.

Optical links connecting two cabinets or racks fall typically under the umbrella of local area networks. The technology of choice in this case is most often fiber optics. The bandwidth and the immunity from crosstalk provided by fiber links are important at this particular level of hierarchy. The relevant optoelectronic technology for this application is well developed and the critical issues involve communication protocols and software interfaces. This aspect of optical interconnects is well advanced and closest to practical realization.

D. Multi-stage Interconnection Networks

Multi-processor computing architectures, which use parallel or distributed processing, require parallel communications. Consequently, as the number of nodes in an architecture grows, the overall system performance quickly becomes communications bound. The most obvious approach to arbitrary intra-processor interconnections is a crossbar switch, which may be thought of as a two-layer grid, horizontal paths in one layer, vertical paths in the other, and ON-OFF switches at the intersections. To connect N inputs to N outputs, the crossbar switch requires N^2 switches, resulting in high complexity. Because of this, computer scientists have turned to multistage interconnection networks for those parallel processing systems in which the number of interconnected nodes is large.

Generally, we define a stage of a multistage interconnection network (or permutation network) as containing a Link Interconnection Pattern followed by a set of switching elements (also called exchange units), as shown in Figure 1. The Link Interconnection Pattern is a set of N one-to-one data paths; a switching element between two adjacent paths can assume either the barred state, which simply passes the data along, or the crossed state, in which data paths are exchanged. (In some networks, the switching element is endowed with a third state which reverses the direction of propagation and sends data back along the adjacent path. See Figure 2.)

The perfect shuffle (PS) is probably the most studied of the various link interconnection patterns. It consists of a set of 2^n paths which shuffle the data path order as shown in Figure 3. One important property of the PS is that if you perform $\log_2 N$ perfect shuffles on N elements, they appear in their original order.

For networks of size $N=2^n$, the link interconnection patterns often include PSs of size N , $N/2$, $N/4$ etc. A number of such well-known networks, namely the data manipulator, flip network,

indirect binary n-cube network, omega network, and regular SW banyan network (with $S=F=2$), have been shown to be topologically equivalent.⁷

The shuffle-exchange is defined as containing a PS of size N followed by $N/2$ exchange units. An interconnection network consisting of several such identical stages can be implemented in an economical manner by using one stage and looping back the data signals.^{8,9} Beginning in the late 70s, computer scientists theorized as to the minimum upper bound to the number of passes (the sufficient number) through the shuffle-exchange a network needs to realize any arbitrary permutation¹⁰⁻¹⁴, depending on the efficiency of the routing algorithm. Wu and Feng¹⁵ proposed an algorithm which called for $3\log_2 N - 1$ passes. They also described a modified shuffle-exchange, which incorporates feedback, that calls for $2\log_2 N - 1$ passes. Also, note that Huang and Tripathi⁸, in 1986 set an upper bound of $2\log_2 N - 1$.

E. Optical Multi-stage Interconnection Networks

As discussed in the introduction, there are several reasons why one would want to use optics to implement an interconnection network.¹⁶ Optical beams can intersect in free space without crosstalk, are insensitive to EMI, possess inherent parallelism and bandwidth, and exhibit high propagation velocity.

The optical architectures proposed for multi-stage interconnections have been both 1-D and 2-D, which refers to the physical dimensionality of the data array. Thus, a 1-D interconnection network contains linear arrays of emitters and detectors, and a 2-D interconnection network contains planar arrays of the same.

The PS interconnection is essentially a data interlacing operation, and most 1-D optical PS architectures implement this in a straight-forward manner.¹⁷⁻¹⁹ The typical architecture incorporates cylindrical lenses to magnify each half of the input data array by a factor of 2. The images are then optically

overlapped so that the elements appear interlaced. If input data pixel arrays have 50% duty cycle (pixel separation equals pixel width), then each data pixel must be masked after magnification to avoid crosstalk.

Architectures built around planar input and output arrays take better advantage of optics' inherent parallelism and, in addition, open new interconnection possibilities. For example, the 2-D PS is described in detail in Reference 10 and the input and output planes for such an interconnection are shown in Figure 4 for $N=16$. The 2-D PS essentially amounts to shuffling the elements in the horizontal and vertical directions independently. The 2-D PS can be implemented in two sequential steps (divide-by-rows-interlace-by-rows, and divide-by-columns-interlace-by-columns) as well as by other constructs. For a given N , the data pattern repeats after $1/2 \log_2 N$ steps instead of $\log_2 N$ steps as in the 1-D PS. Adding switching elements (4×4 crossbars) enables the 2-D PS to perform arbitrary permutations in half as many steps as the 1-D PS; in addition, more than one route is available for the same permutation, giving "fault tolerance." The increased complexity of the switching elements is the price paid for these advantages.

II. Folded Optical Interconnections

A. Background

The basis of the approach developed under this program is the free-space optical interconnect scheme, developed under an earlier phase of this contract, based on encoding the 1-D array of nodes into a rastered or folded 2-D array. The term "folded" is borrowed from work on the optical folded spectrum analyzers in which very long 1-D signals were formatted into 2-D arrays and then processed optically.^{20,21} A key advantage of this approach is that a 1-D PS on a very large data array is performed with the benefits of 2-D optical data encoding, so that the size of the node array is not limited to the 1-D space-bandwidth product (SBWP) of the optical system, but by the 2-D SBWP. For example,

a 1-D optical implementation of a 1024 node PS would require a very stringent optical design with a SBWP product of >1024 ; whereas the folded ps implementation would require an optical system with SBWPs of only >32 along each dimension - a very easy criterion to meet.

Figure 5 shows the input and output planes of this architecture for $N=16$. Like the 2-D PS, the FPS must divide the entire array in half and interlace the individual elements. Only now, the two steps are divide-by-rows-interlace-by-columns and then divide-by-columns-interlace-by-rows. This two-step process is shown in Figure 6. One can observe that if the input array is divided into four quadrants which are then magnified and superimposed, the desired output data array results. This procedure, in one step, gives the same result as the 2-step approach.

The one-step procedure is achieved by masking the input plane and using four imaging lenses as shown in Figure 7. Proper overlay of the four images can be achieved by skewing the samples within each quadrant. In the BDM architecture described above, this skew is manifested in the object mask, so that an unskewed overlapping of the four quadrants is allowed. This required skew can equivalently be effected in the positions of the imaging lenses.²² The selection of the skewing method is governed by the format of the 2-D array (square or rectangular) as well as the convenience of skewing the source/detector pairs versus positioning the lenses

B. Application of Optical Folded Interconnections to Multistage Architectures

For multistage networks, the two issues which are of primary importance are the optical efficiency and input/output array compatibility. The optical efficiency influences the overall energy efficiency and hence scalability of the approach. Both the input/output compatibility and optical efficiency determine the cascadability of the various stages in the

interconnection architecture. The primary driver of both these issues is the amount of light lost at each stage due to the signal-carrying optical rays walking out of the system. The remainder of this section is concerned with the control of light losses in the folded perfect shuffle (FPS) architecture. The results are first derived for a 1-D PS, then extended to the FPS.

The losses in the mask encoded folded perfect shuffle architecture result from two mechanisms. First, since each stage magnifies the elements, masking at the input is needed to preserve element size. This creates a 75% magnification loss. Second, each of the four lenses image not only the nearest quadrant, but all four quadrants, creating a 75% replication loss. This is shown in Figure 8.

Figure 9 shows the approach for overcoming the magnification losses. A pair of lenslet arrays is placed at the output plane to individually demagnify each element by a factor of two. Thus, the elements are, in effect, magnified during the imaging and then demagnified to yield the original spot size. However, since the pair of lenslets also doubles the magnitude of the slopes of the rays incident upon it, rays will walk out of the network after only a few stages. These rays will compound the replication losses.

In this approach replication loss is overcome by having each point source emit a fan of light no wider than to fill the nearest imaging lens. This requires refractive wedglets to redirect beams in the next stage. In the following discussion we derive the required characteristics of the wedgelet array and show how, when combined with the lenslet arrays, they lead to a nearly lossless FPS implementation. The goal is to have both lenslet and wedgelet arrays that are amenable to mass production.

With the paraxial ray approximation, a wedgelet effects a uniform slope change to all rays in the fan that strikes it. With this assumption, only two slope changes are needed for the 1-D PS, regardless of data array size: zero, and $\pm 2(N-1)d/9f$,

where d is the inter-element spacing, and f is the imaging lens focal length. The results are similar for the 2-D folded architecture; only four slope change magnitudes are needed. The resulting architecture is depicted in Figure 10.

For systems in which the paraxial ray approximation is not valid, such as in highly compact, low f -number systems, beams may begin walk out of the system after several stages. We have determined the effects of non-paraxial rays through a digital ray-trace simulation which measures the amount of walk-off the signal-carrying optical rays will suffer. The following summarizes the encouraging results of this simulation.

We simulated the behavior of the 1-D PS architecture, and logically extend the results to the FPS. For any one element, the rays originate in the object plane of the first stage, pass through an imaging lens, converge upon the image plane, continue through the lenslets, and are bent at the object plane of the next stage by the wedgelet. The simulation determined the amount of bending of the non-paraxial rays that occurs at the wedgelet, and whether or not rays in the fan begin to miss the imaging lenses.

We simulated a 1-D 32-element perfect shuffle with an inter-element separation d of 1000 microns and a wedge index of 1.5. We varied the imaging lens focal length (and hence the "paraxiality") and the spread of the incident fans (the "fill factor"). In general we found that only minor losses occur, and generally these were after many stages and due to extremely nonparaxial conditions (low f -numbers). Naturally, smaller fill factors lead to fewer losses, as did longer focal lengths. The other very interesting observation we made is that the position set of the paths for any element eventually settles to an oscillatory equilibrium state. Rather than causing the beams to walk out of the system, the wedgelets seem to cause a convergence. This means that even if losses occur, they are not catastrophic in the sense that eventually all rays will walk out of the

optical system. Instead, the losses appear to be of a "one-time" variety, with all remaining rays converging to stable paths through the network, indefinitely. Even these minor losses appear to be correctable with suitable initial direction of the rays from the original sources. All the simulation data suggest that, if a suitable initial condition set is chosen, even short focal length systems can effect the perfect shuffle with no losses. These results will be presented in a paper accepted at the 1990 International Optical Computing Conference at Kobe, Japan. The submitted summary of this paper is attached as Appendix B.

C. Application of Optical Folded Interconnections to Hypercube Topologies.

Hypercube interconnection networks have seen intensive development in the parallel computer architecture community. There are commercial machines on the market (NCUBE) and the cosmic cube project at CalTech has been in existence for a number of years. In contrast to the perfect shuffle permutation (which is a 1-1 connection), hypercube network requires "m" I/O ports for each of "N" processing elements (PE) in the array, where m is equal to $\log_2 N$. Thus each PE connected to "m" other processing elements. The topology of interconnects is defined by considering the N PEs to occupy the corners of a hypercube in an m-dimensional hyperspace. Direct connections are made between PEs that are connected by edges in the hyperspace, or in other words, between PEs whose m-bit addresses differ in only one bit position. This topology is illustrated in Figure 11. The complexity of the wiring diagram is evident from the figure, although only a few connections are shown.

The PEs are implemented in custom VLSI technology and depending on the complexity of the PE and the number of PEs, a number of chips and the number of boards in the system will vary to build the hypercube processor. If the PEs are arranged in a 2-D array (as will be reasonable for electronic technology), the

arrangement shown in Figure 12 will result. It can be seen that the connections along the LSB edge are nearest neighbors along a row and hence readily implemented in electronics. Nearest neighbours along a column represents connections along the second MSB in this case. Connections along second LSB and MSB are no longer local even in this simple case of 16 PEs. These connections when implemented in electronics can lead to clock skew, cross talk and inefficient usage of valuable real estate on a Si chip. The symmetries in these connections can be easily noticed in Figure 11. This suggests that very simple free space optical imaging systems can be used to implement these nonlocal interconnects. Figure 12 shows the schematic diagram of an optical system implementing the second LSB connections. It can be seen from the previous figure that the second LSB involves a connection between the first and third column and second and fourth column. This can be achieved optically by inverting the whole matrix horizontally (by the first cylindrical lens) and then horizontally inverting the two vertical halves of the matrix independently by two cylindrical lenslets. The final matrix shows the desired permutation.

Connections along the MSB can be implemented by applying a similar procedure, except along rows and inverting the matrix vertically. For hypercubes of higher dimensions, a recursive procedure can be defined in which the matrix is divided in 2, 4, 8, 16... parts along columns/rows and inverted horizontally/vertically, respectively.

In summary, the hypercube interconnection topology is amenable to simplification by folding it in a 2-D array and further that simple optical systems can implement the desired interconnections between the PEs arranged in a 2-D array.

Appendix A: References

1. C-L Wu and T-Y Feng, "Tutorial: Interconnection Networks for Parallel and distributed Processing", IEEE Catalog Number EH0217-0, 1984.
2. J. W. Goodman, "Optics as an Interconnect Technology", in Optical Processing and Computing, H. H. Arsenault, T. Szoplik, and B. Macukow eds., Academic Press, (1989).
3. Special Issue on Optical Interconnection, Optical Engineering, October 1986.
4. B. D. Clymer, "Optical Clock Distribution for VLSI", Ph. D. Thesis, Dept. of Electrical Eng., Stanford University (1987).
5. A. Husain, "Optical Interconnect of Digital Integrated Circuits and Systems", Proc. SPIE, Vol. 466, p. 10, (1984).
6. D. Z. Tsang, "Alignment and Performance Tradeoffs for Free Space Optical Interconnects", paper TuC3, Topical Meeting on Optical Computing, February 27-March 1, 1989, Salt Lake City, Utah.
7. C.-L. Wu and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," IEEE Trans. Computers, vol. C-29, pp. 694-702, 1980.
8. S.-T. Huang and S. K. Tripathi, "Finite State Model and Compatibility Theory: New Analysis Tools for Permutation Networks," IEEE Trans. Computers, vol. C-35, pp. 591-601, 1986.
9. S.-H. Lin, T. F. Krile and J. F. Walkup, "2-D Optical Multistage Interconnection Networks," Proc. SPIE, vol. 752, pp.209-216, 1987.
10. H. S. Stone, "Parallel Processing with the Perfect Shuffle," IEEE Trans. Computers, vol. C-20, pp. 153-161, 1971.

11. H. J. Siegel, "The Universality of Various Types of SIMD Machine Interconnection Networks," Proc. 4th Annu. Symp. Comput. Arch., pp. 70-79, Mar. 1977.
12. T. Lang and H. S. Stone, "A Shuffle-Exchange Network with Simplified Control," IEEE Trans. Computers, vol. C-25, pp. 55-65, Jan. 1976.
13. T. Lang, "Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network," IEEE Trans. Computers, vol. C-25, pp. 496-503, May 1976.
14. D.S. Parker, "Notes on Shuffle/Exchange-type Switching Networks," IEEE Trans. Computers, vol. C-29, pp. 212-222, Mar. 1980.
15. C.-L. Wu and T.-Y. Feng, "The Universality of the Shuffle-Exchange Network," IEEE Trans. Computers, vol. C-30, pp. 324-332, May 1981.
16. J. W. Goodman, F. J. Leonberger, S.-Y. Kung, and R. A. Athale, "Optical Interconnections for VLSI Systems," Proc. IEEE, vol. 72, pp. 850-866, July 1984.
17. A. W. Lohmann, "What Classical Optics Can Do for the Digital Optical Computer," Applied Optics, vol. 25, pp. 1543-1549, May 1986.
18. G. Eichmann and Y. Li, "Compact Optical Generalized Perfect Shuffle," Applied Optics, vol. 26, pp. 1167-1169, April 1987.
19. K.-H. Brenner and A. Huang, "Optical Implementations of the Perfect Shuffle Interconnection," Applied Optics, vol. 27, pp. 135-137, Jan. 1988.
20. C. E. Thomas, "Optical Spectrum Analysis of Large Space Bandwidth Signals," Applied Optics, vol. 5, p. 1782, 1966. (or, Applied Optics 5:1782 (1966).)

21. T. M. Turpin, "Spectrum Analysis Using Optical Processing," Proc. IEEE 69, 79 (1981).
22. Sawchuck, A. A., Glaser, I., "Geometries for Optical Implementations of the Perfect Shuffle", Proc. SPIE, Vol. 963, p. 270, 1988.

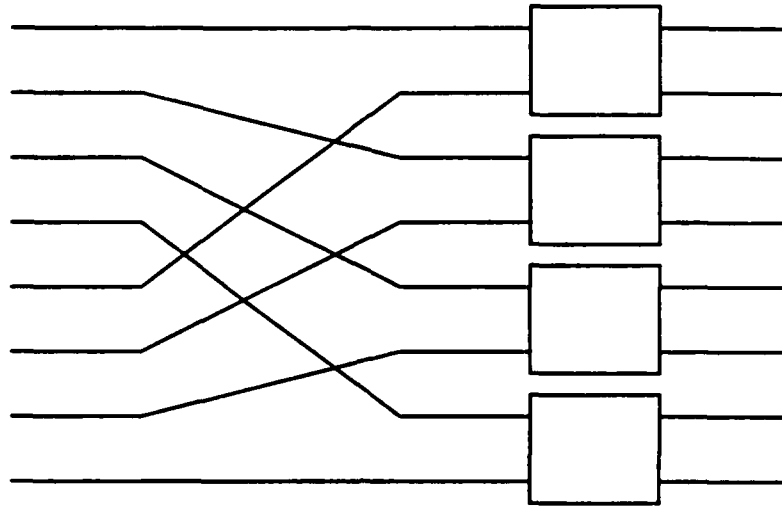


Figure 1. One stage of a multi-stage interconnection network.

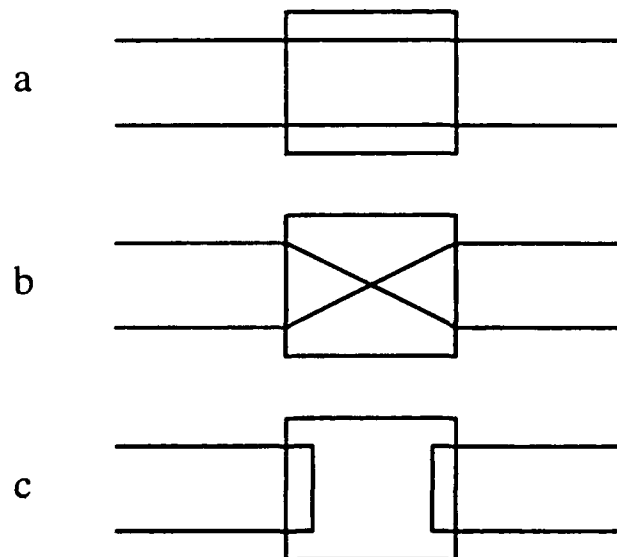


Figure 2. A switching element in: a. the barred state, b. the crossed state, c. the reject state.

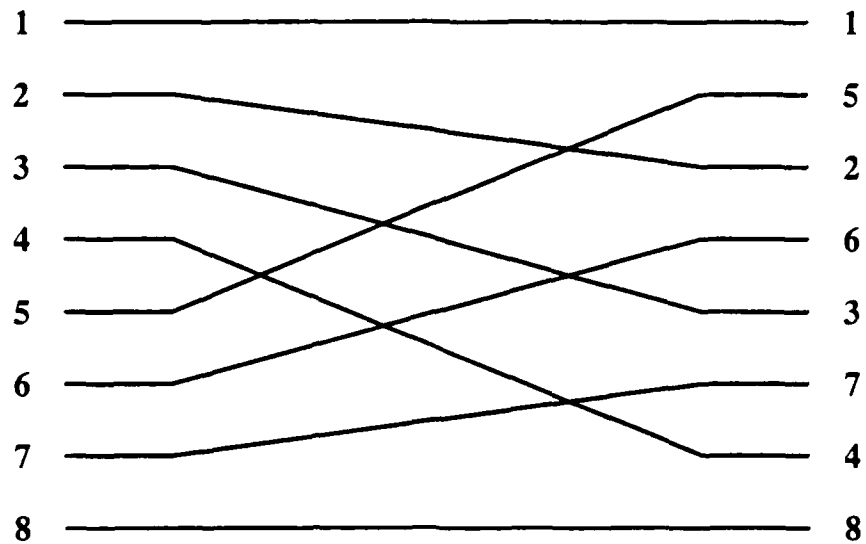


Figure 3. Perfect Shuffle for $N = 8$.

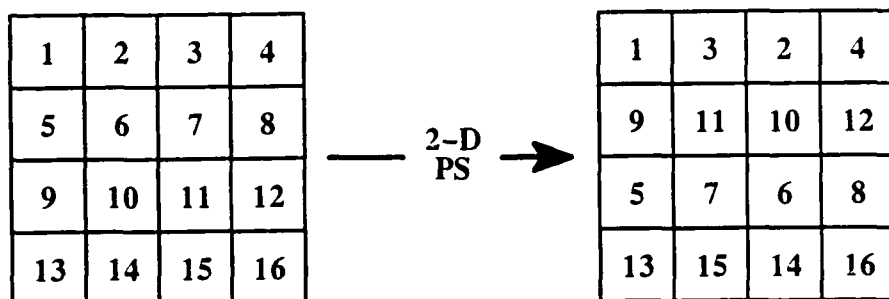


Figure 4. 2-D Perfect Shuffle.

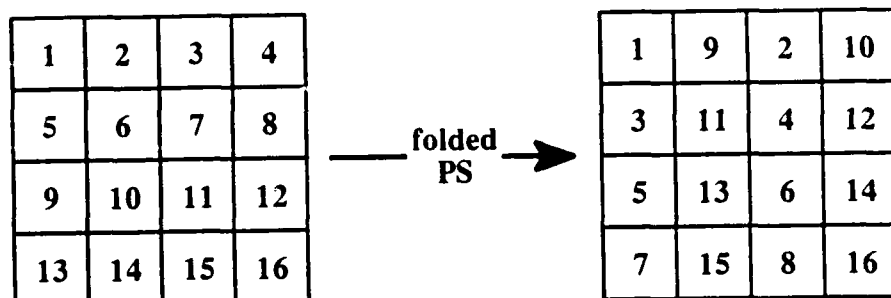


Figure 5. Folded Perfect Shuffle.

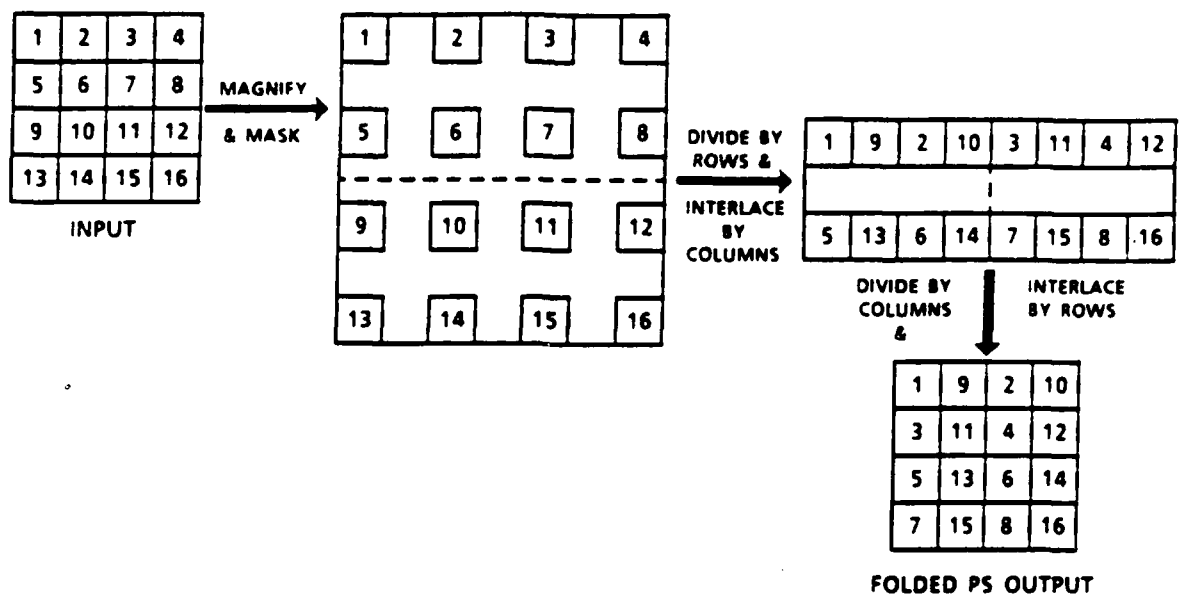


Figure 6. Two-step approach to the FPS.

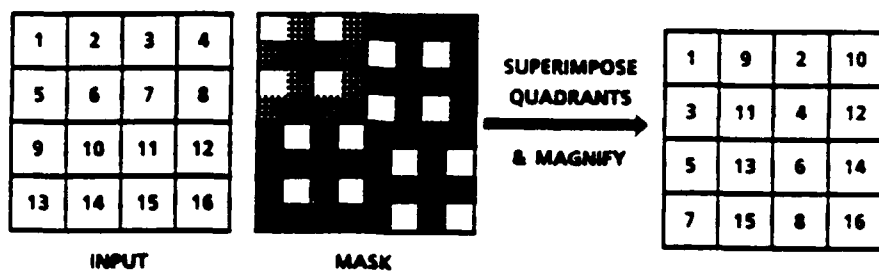
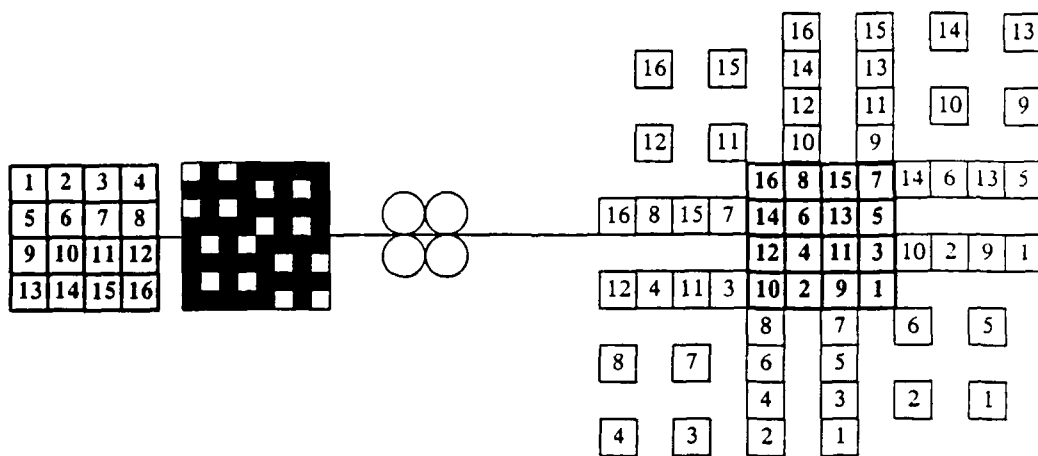


Figure 7. One-step approach to the FPS.



EXPLODED VIEW

Figure 8. Light loss mechanisms in mask-encoded FPS.

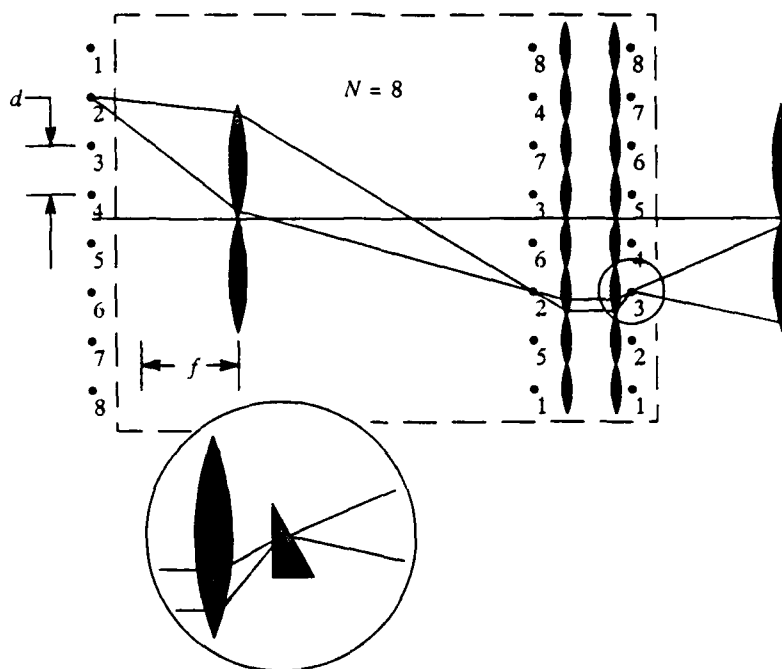


Figure 9. Remedies to magnification and replication losses.

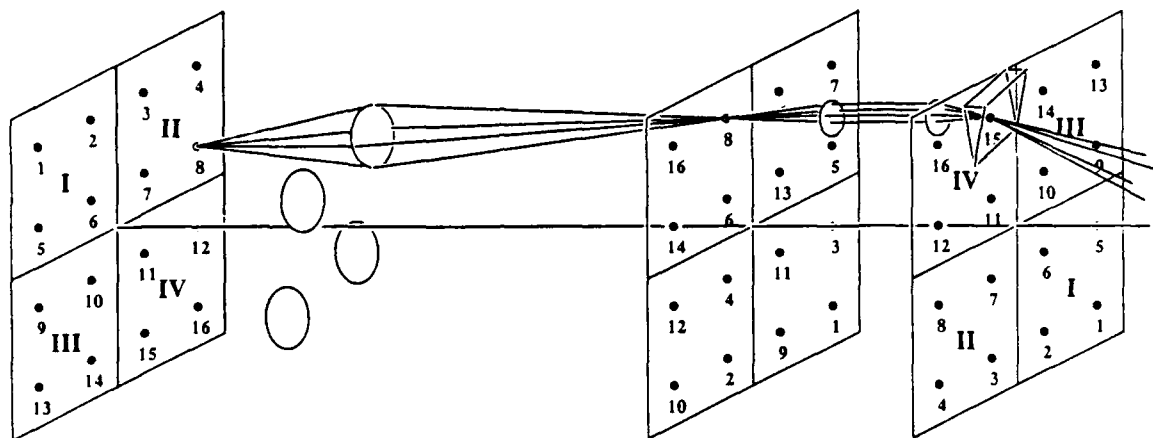


Figure 10. The resulting FPS configuration for $N = 16$. The lenslets and wedgelet for only one output element are shown.

Hypercube Interconnections : Definition

- N processing elements occupy vertices of a hypercube in n-dimensional space ($N = 2^n$).
- Processing elements whose binary address differs by only one bit are connected directly ($1010 \leftrightarrow 1011$).

Example: $N = 16, n = 4$ (only a few connections shown)

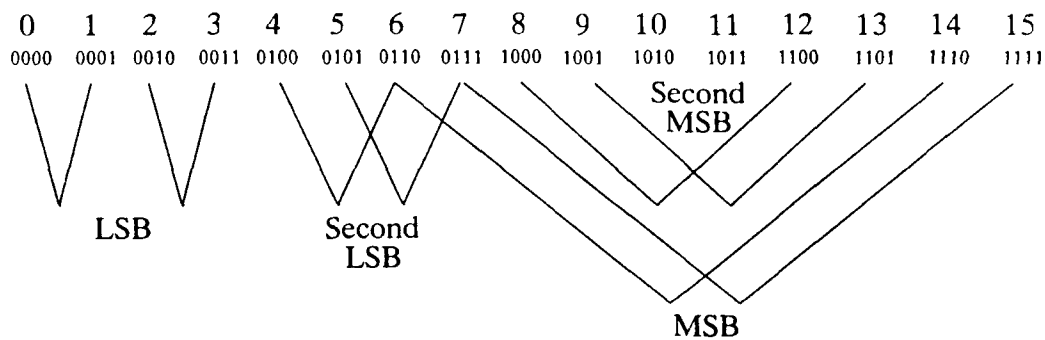


Figure 11. Hypercube Interconnections: Definition.

Hypercube Interconnections : Folded Implementation

- N elements folded into a 2-D array with conventional raster forming
- Interconnections along each edge of a n-hypercube implemented via separate optical system

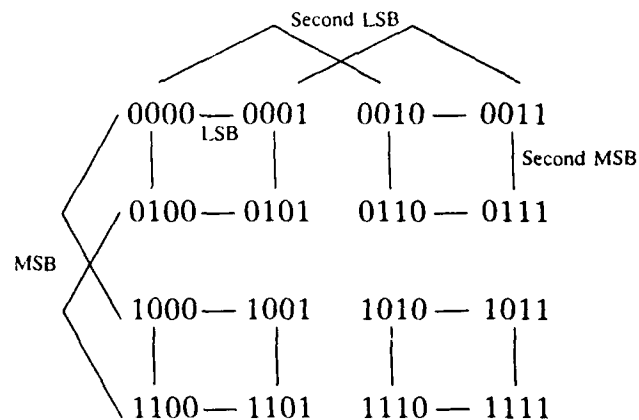


Figure 12. Hypercube Interconnections: Folded Implementation.

Hypercube Interconnections : Optical Implementation

- Nearest neighbor interconnects can be implemented electrically when the processing elements are electronic.
- Optical systems that perform image inversion along one dimension, operating on different subsections of the 2-D image, implement the necessary interconnects.
 - Example shown for second LSB interconnection for a 4 X 4 array. Second MSB interconnections occur between nearest rows. MSB interconnections are obtained by an optical system similar to the one shown, operating on rows.

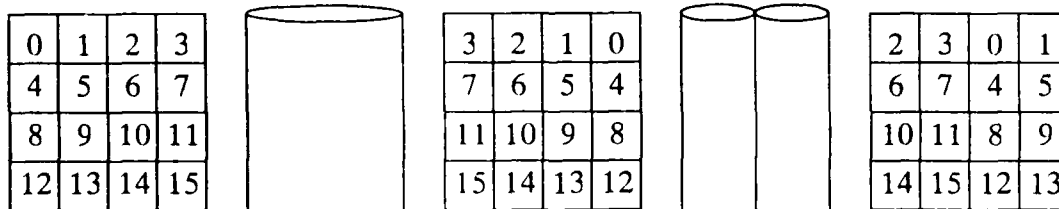


Figure 13. Hypercube Interconnections: Optical Implementation.

Appendix B: Preprint

Low Loss Free-space Folded Perfect Shuffle Network

M. W. Haney and J. J. Levy

BDM International, Inc.

7915 Jones Branch Drive

McLean, Virginia, USA 22102-3396

ABSTRACT: A light efficient and cascadable folded perfect shuffle network is described.

SUMMARY: Classical optics shows great potential for multiprocessor interconnection applications owing to its inherent advantages of high parallelism, low optical crosstalk, and low time skew.¹ The Perfect Shuffle (PS) is a useful communications topology in parallel computer architectures²; PS networks based on classical optics have been suggested.^{3,4,5} The Folded Perfect Shuffle⁶ (FPS) was proposed to overcome the 1-D space-bandwidth product (SBWP) limitations of optics in implementing the PS. In the FPS, the 1-D input processing element (PE) array is raster formatted (folded) into a skewed 2-D array. The skewing can be accomplished with a mask, as shown in Figure 1. Four imaging lenses magnify, overlay, and interleave the rastered array into an identically formatted shuffled array at the output. The number of PEs that can be shuffled is determined by the 2-D SBWP of the imaging system. The optics should therefore be relatively simple. For example, a multiprocessor architecture with 4096 PEs would require optical interconnection PS modules to image 64 x 64 arrays, rather than 1 x 4096 arrays.

It is desirable that the interconnection modules comprising a multistage network be optically cascadable to preclude the need for active signal regeneration at each stage. For cascadability, output and input arrays should have the same array and pixel dimensions, format, and light levels. The proper registration of the output array in the FPS is achieved via either a skewed, non-square format for the arrays⁶ (by either sub-pixel masking or shifting of input elements), or a skewing of the array of the four imaging lenses.⁷ Although, for both of these cases, the input/output formats and array sizes are matched, magnification and replication losses cause the pixel size and light levels to change from stage to stage. The magnification losses are due to the magnification that each pixel undergoes at each stage. The original FPS approach masked the input pixels to reduce their size, but this led to an unacceptable loss of light at each stage. The replication losses stem from each of the four lenses imaging the entire input array, while, for each lens, only one of the quadrants of the input array is overlapped at the output. The other three quadrants are imaged outside the region of interest, thereby losing light.

To achieve cascadability a 2-D lenslet array is added to correct for the magnification of each data pixel, and a 2-D microprism array generates the proper beam angles to overcome replication losses. These concepts are depicted in Figure 2 for a 1-D PS; they extend readily to the 2-D FPS. Lenslet array technology is now well developed. The important issues in the cascadable FPS, therefore, center on the characteristics of the

microprism array required to implement the proper correction to the ray paths, how easily they lend themselves to fabrication, and the optical losses due to nonidealities in the system.

We first calculated the wedge angles and orientations required, for each element of the micro-prism array, based on the assumption of paraxial rays. The surprising result is that there are only 16 wedge types required to implement correct ray bending in the FPS architecture -- this number is independent of the number of PEs being shuffled. Furthermore, of these, four have zero wedge angle, and the other twelve possess just three wedge angles at various rotations about the optic axes for each pixel. A ray-trace computer simulation was conducted to determine the effects of this wedglet array on the light efficiency of a FPS module due to non-paraxial rays. The results show that, in all but extremely low f-number cases (i.e., less than $f/2$), very high light efficiency is achieved. Furthermore, the architecture is tolerant to small variations in the individual wedglets' prism angles, with most of the the rays tending to reach a stable path through an arbitrarily large number of stages.

These results suggest that low-loss FPS modules for multi-stage interconnection applications are feasible. The combination of the FPS's ability to fully exploit 3-D optics, the robustness of the cascable features to light loss, and the relatively small array sizes required to have real practicality (128 x 128 pixels, or less), imply that the required lenslet and micro-prism arrays can be cheaply mass-produced, possibly using the most inexpensive stamped plastic techniques.

1. A. Lohmann, Appl. Opt. 25, 1543 (1986).
2. H. Stone, IEEE Trans. Comput. C-20, 153 (1971).
3. A. Lohmann, W. Stork, G. Stucke, 1985 Opt. Comp. Mtg., WA3.
4. J. Midwinter, IEE Proc. 132, Pt. J, No. 6, 371 (1985).
5. K.-H. Brenner and A. Huang, Appl. Opt. 27, 135 (1988).
6. C. Stirk, R. Athale, and M. Haney, Appl. Opt. 27, 202 (1988).
7. A. Sawchuk and I. Glaser, Proc. SPIE, vol. 963 (1988).

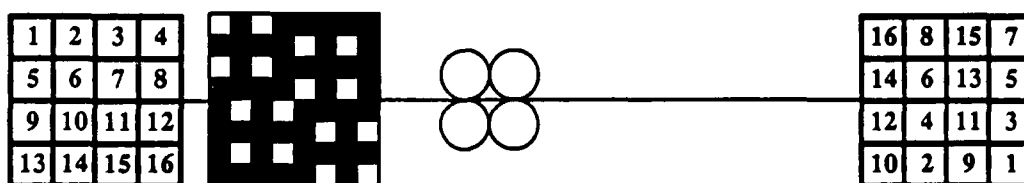


Figure 1. Basic Folded Perfect Shuffle concept (exploded view).

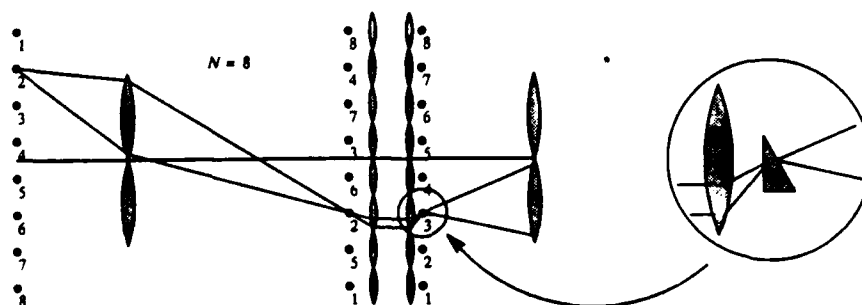


Figure 2. Lenslet and micro-prism arrays give low light loss.