Calhoun: The NPS Institutional Archive

Theses and Dissertations Thesis Collection

1986

A serial bus architecture for parallel processing systems.

Delaney, Kevin J.

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A SERIAL BUS ARCHITECTURE FOR PARALLEL PROCESSING SYSTEMS

by

Kevin J. Delaney

September 1986

Thesis Advisor: Larry W. Abbott

Approved for public release; distribution is unlimited.

T230328

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)* 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c ADDRESS *(City, State, and ZIP Code)*

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE *(Include Security Classification)*

A SERIAL BUS ARCHITECTURE FOR PARALLEL PROCESSING SYSTEMS

12 PERSONAL AUTHOR(S)
Delaney , Kevin J.

3a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1986 September	64

6 SUPPLEMENTARY NOTATION

7 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Parallel processing, Optoelectronic Multiplexer

9 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

One of the most serious deterrants to the development of multiple processor architectures has been the problem of providing adequate communication between the discrete processing elements. This paper examines two communications-based constraints.

The first constraint is related to the physical structure of the VLSI chip. The wider the communication path the more pins are needed to effect the data transfer. As Integrated Circuits grow in computational power, more communication capacity is needed, pushing designs closer to the pin limitations of the packaging technology.

The second constraint, somewhat related to the first, is the limited speed with which data can be transmitted via internal channels. Typical speeds one can achieve on a single wire are on the order of 1 Gbps. The

0 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
2a NAME OF RESPONSIBLE INDIVIDUAL Abbott, Larry W.	22b TELEPHONE *(Include Area Code)* 408-646-2379	22c OFFICE SYMBOL 62At

D FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

19. (continued)
recent development of an Optoelectronic Multiplexer may allow VLSI chips to communicate at rates up to 7 Gbps. An architecture for a parallel processing computer which takes advantage of this new capability is presented. The feasibility of a single-chip parallel processor based on the Optoelectronic Multiplexer is examined by projecting current trends in processor speed, power, and transistor count into estimates of throughput for a multi-processor IC.

Approved for public release; distribution is unlimited.

A Serial Bus Architecture for Parallel Processing Systems

by

Kevin J. Delaney
Lieutenant, United States Navy
B.S.E.E., United States Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

263
.1

ABSTRACT

One of the most serious deterrants to the development of multiple processor architectures has been the problem of providing adequate communication between the discrete processing elements. This paper examines two communications-based constraints.

The first constraint is related to the physical structure of the VLSI chip. The wider the communication path the more pins are needed to effect the data transfer. As Integrated Circuits grow in computational power, more communication capacity is needed, pushing designs closer to the pin limitations of the packaging technology.

The second constraint, somewhat related to the first, is the limited speed with which data can be transmitted via internal channels. Typical speeds one can achieve on a single wire are on the order of 1 Gbps. The recent development of an Optoelectronic Multiplexer may allow VLSI chips to communicate at rates up to 7 Gbps. An architecture for a parallel processing computer which takes advantage of this new capability is presented. The feasibility of a single-chip parallel-processor based on the Optoelectronic Multiplexer is examined by projecting current trends in processor speed, power, and transistor count into estimates of throughput for a multi-processor IC.

TABLE OF CONTENTS

I. INTRODUCTION 10

A. THE NEED FOR PARALLEL PROCESSING 10

B. PARALLEL PROCESSORS DEPEND ON COMMUNICATION 11

1. Exhaustive Communications 11

2. Limited Communications 12

C. THE OPTOELECTRONIC MULTIPLEXER CONCEPT 13

1. Optical Switching Yields High Speed 13

2. A Suitable Architecture Sought 14

II. OPTIMUM ARCHITECTURE OF LARGE INTEGRATED CIRCUITS 17

A. PARTIONING SILICON FOR MAXIMUM THOUGHPUT 17

1. Transistor Constraints 18

2. Power Constraints 25

B. MINIMUM CHIP SIZE FOR OM APPLICATION 28

1. Minimum Transistor Count 28

2. Minimum Power Dissipation 32

III. THE NEED FOR A HIGH-SPEED MULTIPLEXER 34

A. PROCESSOR POWER LIMITED BY COMMUNICATION PATH 34

IV. SYSTEM ARCHITECTURE BASED ON SERIAL COMMUNICATION 37

A. ON-CHIP DATA FLOW ARCHITECTURE 37

1. Pipeline Architecture 37

2. Reuse Architecture 38

3. Interleaving Data Sets 46

B. DATA DISTRIBUTION 50

1. Pipeline Architecture 50

2. Reuse Architecture ... 51
C. RECEIVER TASKS ... 53
D. TRANSMITTER TASKS ... 55
E. CONCLUSIONS AND LIMITATIONS OF THIS RESEARCH ... 58
1. Conclusions ... 58
2. Limitations and Recommendations ... 59

LIST OF REFERENCES ... 60

INITIAL DISTRIBUTION LIST ... 63

LIST OF TABLES

I. SPECIFICATIONS OF SOME ACTUAL PROCESSORS 18
II. EXPERIMENTAL CONSTANTS 20
III. SPECIFICATIONS OF SOME ACTUAL PROCESSORS 25
IV. EXPERIMENTAL CONSTANTS 28
V. INTER-PROCESSOR COMMUNICATIONS 4 X 1 REUSE ARCHITECTURE 44
VI. INTER-PROCESSOR COMMUNICATIONS 2 X 2 REUSE ARCHITECTURE 44
VII. INTER-PROCESSOR COMMUNICATIONS MODIFIED 4 X 1 REUSE ARCHITECTURE 46
VIII. PRESET SCHEDULE FOR DATA DISTRIBUTION PIPELINE ARCHITECTURE 52
IX. PRESET SCHEDULE FOR DATA DISTRIBUTION PIPELINE ARCHITECTURE 53
X. PRESET SCHEDULE FOR DATA DISTRIBUTION REUSE ARCHITECTURE 54
XI. PRESET SCHEDULE FOR DATA DISTRIBUTION REUSE ARCHITECTURE 55

LIST OF FIGURES

1.1 Exhaustive Communicatons 11

1.2 Limited Communications--Dedicated Path Loop 12

1.3 Limited Communications--Dedicated Path Regular Network 13

1.4 Limited Communications--Shared Path 14

1.5 Optoelectronic Multiplexer Block Diagram 15

2.1 Processor Speed and Complexity (Experimental) 19

2.2 Processor Speed and Number of Processors Based on a Constant Number of Transistors 22

2.3 Relationship Between Processor Capability And System Requirements (Speed a Strong Function of Complexity) 23

2.4 Relationship Between Processor Capability And System Requirements (Speed a Weak Function of Complexity) 24

2.5 Processor Speed and Power (Experimental) 27

2.6 Processor Speed and Number of Processors Based on a Constant Chip Power Level 29

2.7 Relationship Between Processor Capability And System Requirements (Speed a Weak Function of Power) 30

4.1 Sixteen Point Fast Fourier Transform 39

4.2 Sixteen Point Fast Fourier Transform Pipline Implementation 40

4.3 Sixteen Point Fast Fourier Transform Performed by 4 × 1 Chips 41

4.4 Sixteen Point Fast Fourier Transform Performed by 2 × 2 Chips 42

4.5 Sixteen Point Fast Fourier Transform 4 × 1 Reuse Architecture 43

4.6 Sixteen Point Fast Fourier Transform 2 × 2 Reuse Architecture 45

4.7 Sixteen Point Fast Fourier Transform Modified 4 × 1 Reuse Architecture 47

4.8 Sixteen Point Fast Fourier Transform Modified 4 × 1 Reuse Architecture Non-Interleaved 48

4.9 Sixteen Point Fast Fourier Transform Modified 4 × 1 Reuse Architecture Interleaved 49

4.10 Sixteen Point Fast Fourier Transform Distributed Among Four Multi-Processor Chips 51

4.11 Sixteen Point Fast Fourier Transform Distributed Between Two Chips Using a Reuse Architecture 56

4.12 Receiving Bus Interface Unit Architecture 57
4.13 Transmitting Bus Interface Unit Architecture 58

I. INTRODUCTION

> Farmers once used oxen to plow their fields. And when the task got too big for one ox they did *not* try to grow a bigger ox. They got *two* of them! [Ref. 1]

A. THE NEED FOR PARALLEL PROCESSING

So too have we often found that one computer is not enough, or at least, not *fast* enough for many applications. While progress on producing faster single processor computers continues, it is the orders of magnitude leap in speed possible in multiple-processor computers that promises to lead computing into its Fifth Generation.

> [Multiple-processor computers became] necessary because a limit to higher speed had been reached with brute-force approaches employing faster switching devices. Faster components made with gallium arsenide or Josephson junction devices can increase computer speed only 10 times if current uniprocessor architectures are used; however with the new architectures, there is hope of increasing speed 100 to 1000 times. [Ref. 2]

Such dramatic increases in computer speed would be of great benefit to researchers working on computationally-intensive and/or real time problems such as adaptive antenna control, weather prediction, or fusion reactor design. It is not merely a question of having the answers in seconds instead of minutes--once machines can perform calculations in *real time*, whole new applications suddenly become possible.

As an example, consider a computer system which calculates the power spectral density of intercepted radar emitters. A system which takes an hour to analyze a few seconds' worth of data may be useful to compile electronic intelligence data back at fleet headquarters--it produces answers long after the event is over. However, if the system could perform its analysis in real time it could be used onboard ship or in an aircraft to recognize hostile missile seekers and dispense chaff or activate jammers--that is, to respond to events as they happen. Increased speed alone could make this new application possible.

B. PARALLEL PROCESSORS DEPEND ON COMMUNICATION

> When using a number of processors on a single problem, the exchange of data between processors becomes a critical bottleneck. [Ref. 3]

Extensive research has already been conducted in many areas related to parallel processing, such as task distribution and software development. The research reported in this paper focused on the architecture of parallel-processing systems, especially with regard to inter-processor communications.

A system which uses more than one processor to perform a task must provide communication paths between the processors. There are essentially two approaches to this requirement:

- provide a path from every processor to every other processor--"exhaustive" communications
- provide paths between each processor and only some of the other processors--"limited" communications.

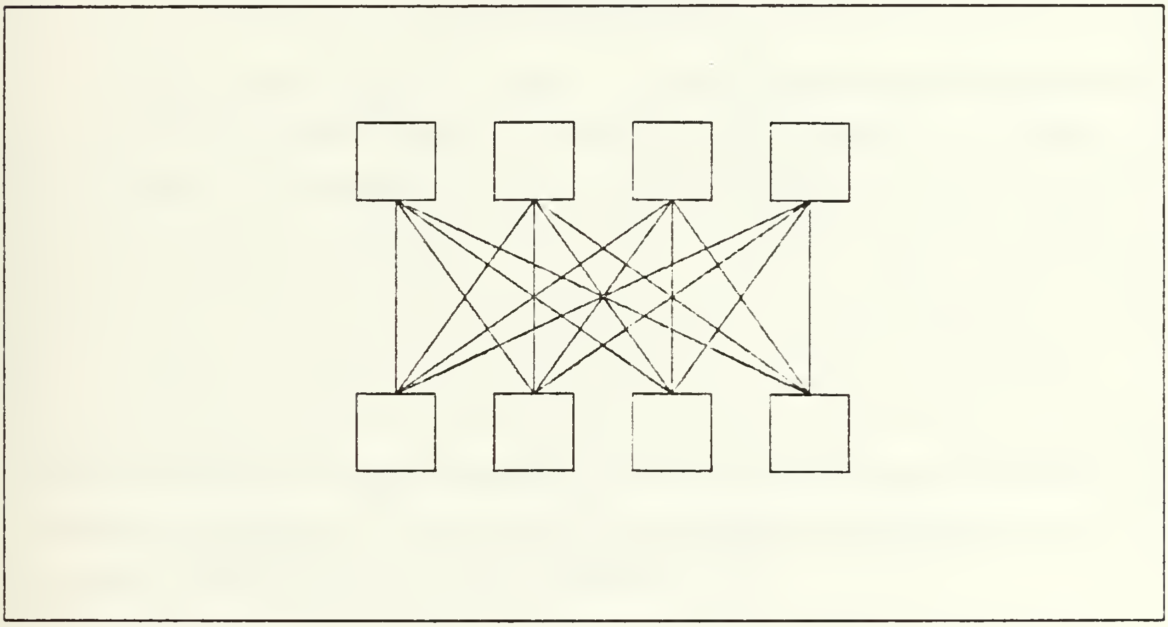

Figure 1.1 Exhaustive Communicatons.

1. Exhaustive Communications

An exhaustive communication architecture (Figure 1.1) provides direct data exchange without bus contention or waiting. However, as the number of processors rises, the number of communication paths in an exhaustive architecture becomes impractically large, leading to high costs. In addition, expansion of the network may be limited by the inability of the existing processors to accept another communication port. These difficulties with exhaustive communciation architectures have led many researchers to consider architectures based on limited communications.

2. Limited Communications

In limited communication architectures, [Ref. 4] identifies two major groups: dedicated path and shared path structures. Limited architectures employing dedicated paths enable a processor to exchange data without bus contention or waiting, but only with a limited number of processors. Figures 1.2 and 1.3 show two examples of a limited communication architecture employing dedicated paths.

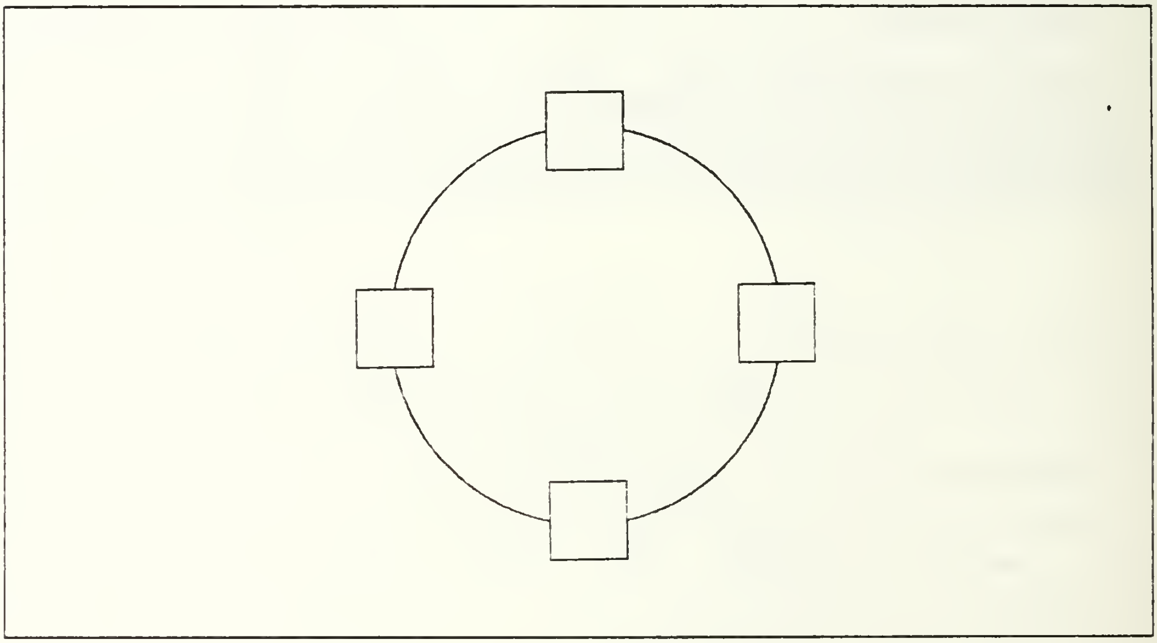

Figure 1.2 Limited Communications--Dedicated Path Loop.

Parallel-computing systems built around a limited communications-dedicated path concept can take advantage of the immediate communication between a given processor and the processors adjacent to it. Yet if a problem requires communication between non-adjacent processors, the message must be passed along by all the intermediate processors. Should the message reach a busy node, it may be delayed or even discarded, forcing a re-transmission. The resultant communication overhead could tie up the system and severely slow its operation.

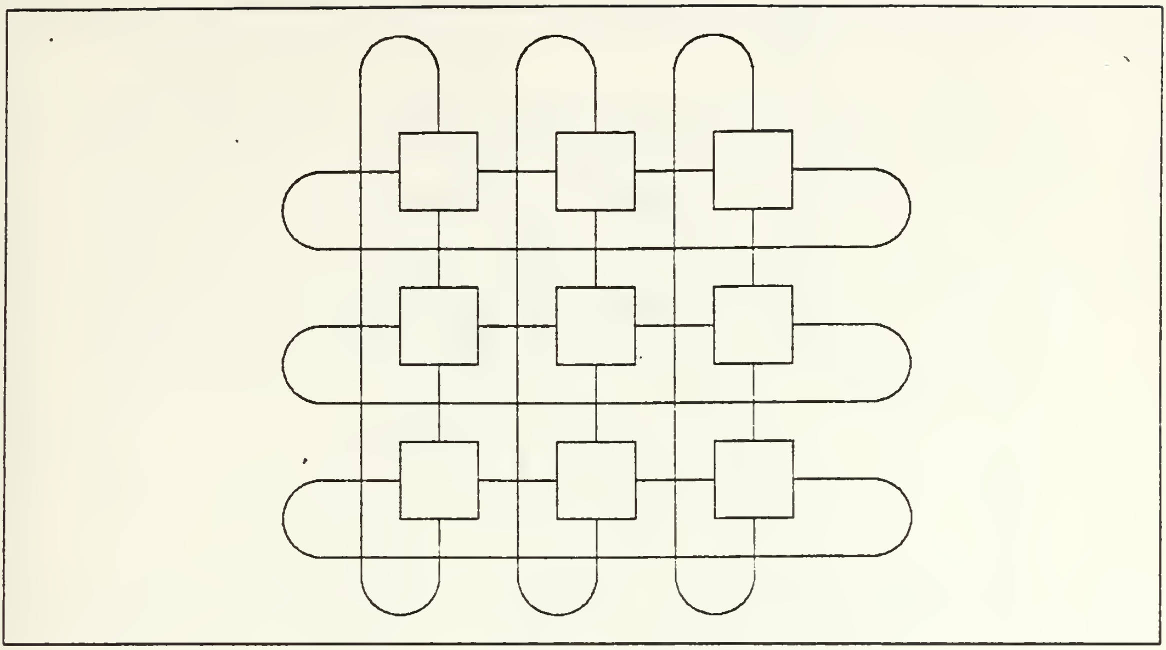

Figure 1.3 Limited Communications--Dedicated Path Regular Network.

Using a shared path (as in Figure 1.4) eliminates the need to relay data from one processor to another, because an uninterrupted path already exists between any two processors. For this reason, limited shared-path architectures are more flexible in the kinds of data flows which can be achieved and in the types of problems which can be solved than limited dedicated-path architectures. However, because processors must wait their turn to use the common communication path, system throughput may suffer. That is, unless the common bus runs at such a high speed that the processors can barely keep up with the bus. Such a high speed bus design would require a multiplexer on each chip capable of speeds considerably in excess of the speeds associated with conventional multiplexers. The Optoelectronic Multiplexer (OM) developed by the Naval Ocean Systems Center, San Diego, is such a device.

C. THE OPTOELECTRONIC MULTIPLEXER CONCEPT

1. Optical Switching Yields High Speed

The Optoelectronic Multiplexer employs optically-activated junctions to sequentially link parallel data lines onto a serial bus. [Ref. 5] A laser pulse, fed to the junction by optical fiber, activates the junction, allowing conduction from the input line onto the main data transmission line. By using a different length of optical fiber

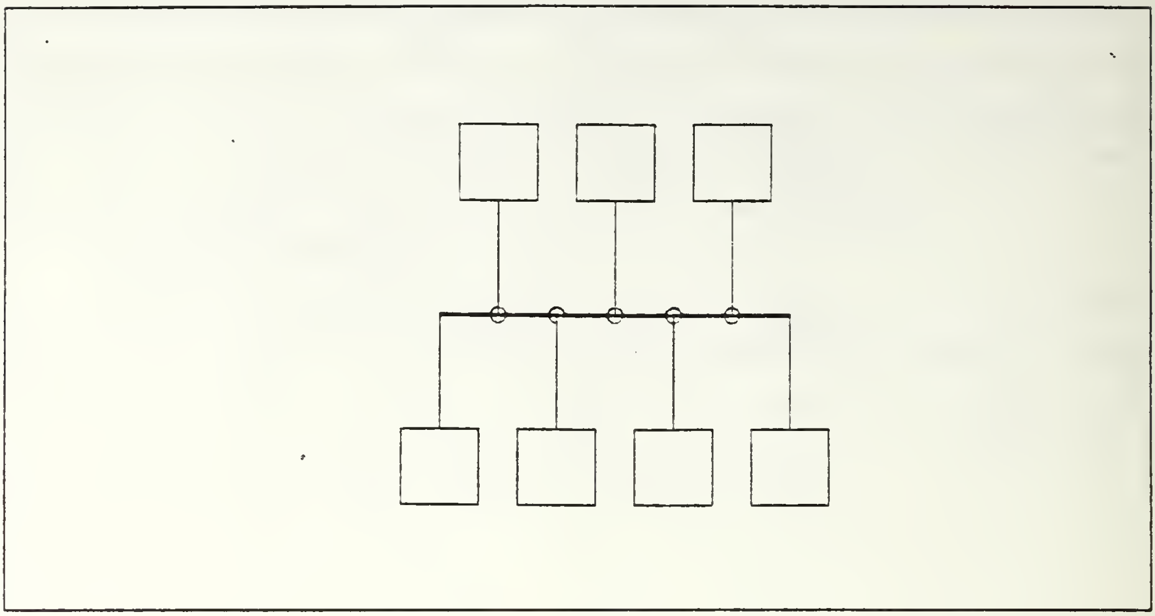

Figure 1.4 Limited Communications--Shared Path.

for each junction, the laser pulses will arrive at the junctions at different times. Consequently, the junctions are activated one at a time, which converts the parallel data waiting on the input lines to serial data pulses travelling along the output transmission line. The short pulsewidths generated by the laser allow extremely high pulse repetition frequencies--researchers have tested a prototype laser multiplexer at speeds as high as 7 Gbps. [Ref. 5]

2. A Suitable Architecture Sought

Current research [Refs. 6 - 10] is especially rich in parallel-processing architectures based on limited communication dedicated-path concepts, because shared path communications typically involve delays which could detract from the high performance otherwise achievable by parallel-processing designs. Prompted by the development of the high-speed Optoelectronic Multiplexer, which promises an increase in serial communication speed of at least one and perhaps two orders of magnitude, this project evaluated the impact of using a shared bus and serial communication in a parallel processing computer architecture. Specifically, the following questions were posed: With current technology, is it feasible to fabricate an Optoelectronic Multiplexer-based multiple processor chip? What new architectures are made possible by the OM's high speed? Which architecture makes optimum use of this new capability?

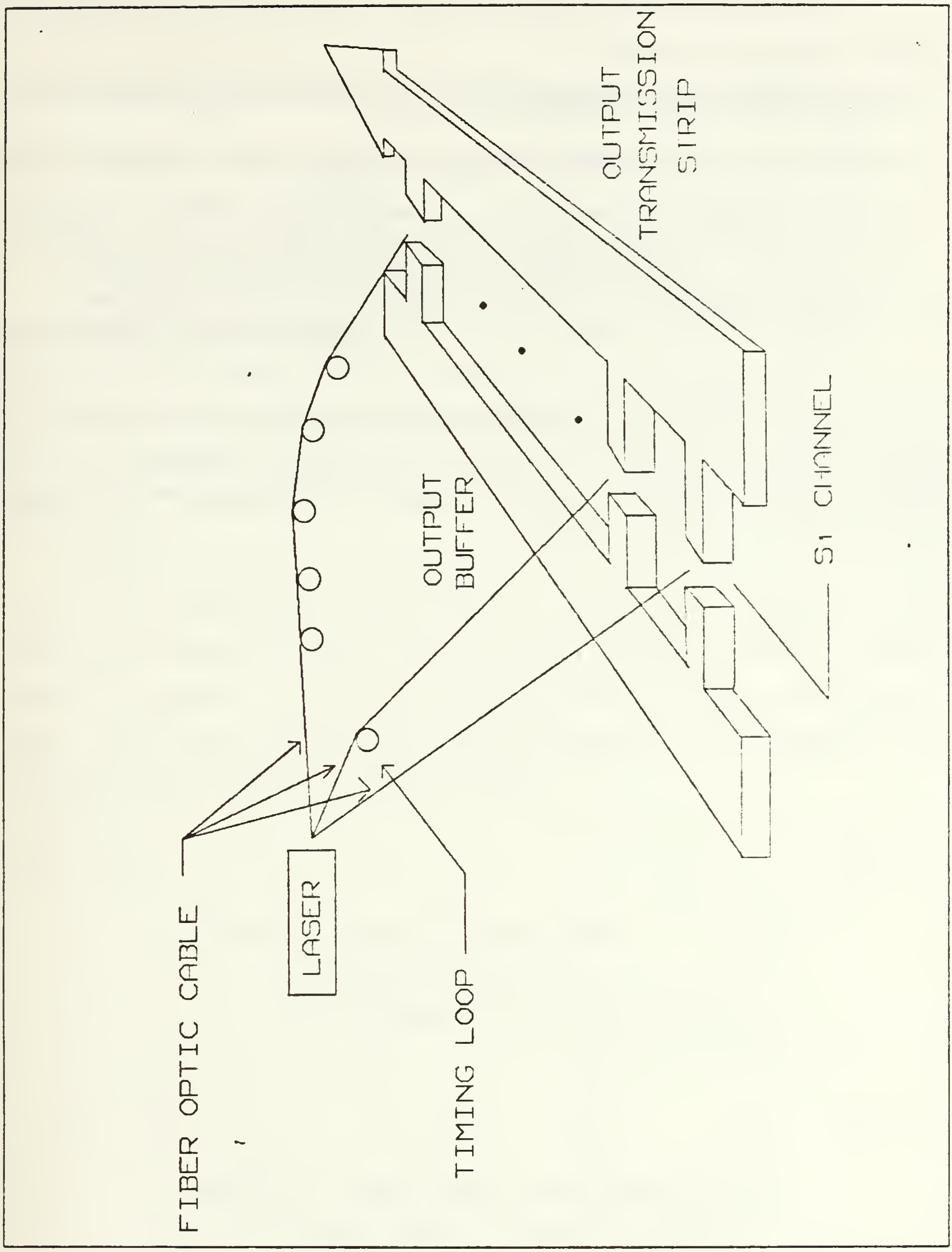
FIBER OPTIC CABLE
LASER
TIMING LOOP
OUTPUT BUFFER
OUTPUT TRANSMISSION STRIP
S1 CHANNEL

Figure 1.5 Optoelectronic Multiplexer Block Diagram [Ref. 11].

Four conditions would have to be met in order for a single-chip OM-based parallel processor to be feasible:

- IC manufacturing technology should be able to fabricate enough transistors on a single chip to create a multi-processor chip.
- A large chip partitioned into many processors would produce higher throughput than the same chip fabricated as a large uniprocessor.
- Chip throughput (measured in bits per second) would exceed the capacity of conventional multiplexers, justifying the use of the OM.
- The package of such a multiple processor chip would require so many pins that package size would be excessive and a multiplexer would be used instead.

The first condition is easily dealt with by a specific example. The Intel 8080 microprocessor contained about 4500 transistors [Ref. 12], while Motorola's MC68020 contains about 200000 [Ref. 13]. Using the technology of the Motorola MC68020, one could produce a chip with over 40 Intel 8080s. Clearly, manufacturers can *already* fabricate a multiple-processor chip. The remaining points require further discussion and are covered in Chapters II and III.

II. OPTIMUM ARCHITECTURE OF LARGE INTEGRATED CIRCUITS

Chapter I's demonstration that a multiple-processor chip could be fabricated prompts the following questions:

- Is a multiprocessor chip the best use of IC fabrication technology, or should all available transistors be assembled into a single processor?
- How large (in terms of transistor count, heat dissipation, and number of processors) would a chip have to be in order to justify the use of the Optoelectronic Multiplexer?

A. PARTIONING SILICON FOR MAXIMUM THOUGHPUT

Should designers divide the available silicon among a few large and capable processors or among many, less capable processors? Which mix yields the highest throughput?

Consider a system of N processors, each executing the same program and producing the same number of output data words each second. Applications of such architectures abound in the field of real time signal processing, which uses regularly structured algorithms. As N increases, processors share the load, so each may run more slowly without changing the speed of the *system.* If we imagine a system throughput goal of R bits per second (bps), then:

$$R = NS \qquad \text{(eqn 2.1)}$$

where R = System throughput (bps)
N = Number of processors
S = Throughput of each processor (bps).

$$S_{req'd} = RN^{-1} \qquad \text{(eqn 2.2)}$$

where $S_{req'd}$ = Speed required of each processor
in order to meet the system goal of R bps.

These equations describe what is *required* of a processor--but how does a processor's *actual* performance vary with N? At issue is the apportionment of the entire chip's allotment of transistors and heat dissipation ability among N processors.

1. Transistor Constraints

Assuming we can put only so many devices on a chip, then:

$$t = TN^{-1} \qquad \text{(eqn 2.3)}$$

where t = complexity of any processor, measured in transistors
N = number of processors
T = Total number of transistors on chip

Generally, a complex processor will be able to perform a given calculation faster than a simple processor. For example, a microprocessor with an on-board floating-point unit can handle a multiplication in a few clock cycles, while a smaller processor has to do tedious successive additions, requiring much more time. But what is the exact relationship between processor complexity and speed? To answer this we shall examine the specifications of some existing processors, as listed in Table I and graphed in Figure 2.1.

TABLE I

SPECIFICATIONS OF SOME ACTUAL PROCESSORS

Group	Reference	Data Word (Bits)	Time Required for Multiplication (10^{-6} sec)	Bit Rate (10^{6} sec^{-1})	Transistor Count (thousand)
CPU's 1981-82	Ref. 14	32	8.30	3.86	60.
	Ref. 15	16	6.25	2.56	40.
	Ref. 16	32	1.80	17.8	450.
CPU's 1982-85	Ref. 17	32	5.50	5.82	25.
	Ref. 18	32	4.50	7.11	24.
	Ref. 19	32	2.70	11.9	32.1
	Ref. 20	32	0.75	42.7	65.
	Ref. 21	32	0.32	100.	120.
	Ref. 22	80	0.18	444.	150.
FPU's	Ref. 23	16	.090	178.	6.5
	Ref. 24	16	.080	200.	7.5
	Ref. 25	16	.060	267.	10.3
	Ref. 26	16	.045	356.	10.35
	Ref. 27	16	.027	593.	11.5
	Ref. 28	16	.079	405.	23.0

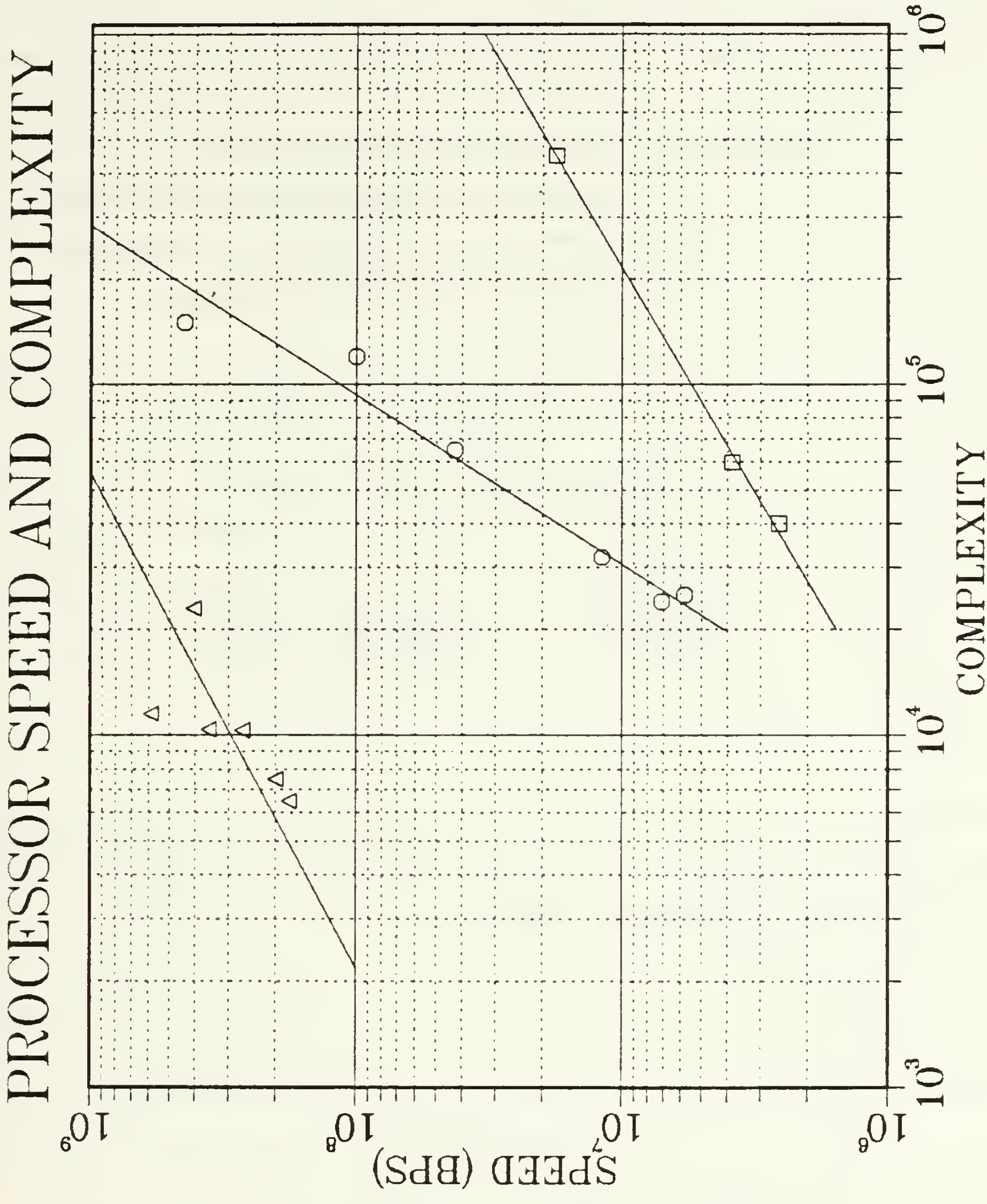

Figure 2.1 Processor Speed and Complexity
(Experimental).

From the experimental relationships between processor speed and complexity shown in Figure 2.1, we can see that the data in each group are approximated by the equation:

$$S_{proc} = At^{a} \qquad \text{(eqn 2.4)}$$

where S_{proc} = processor speed (in bps throughput)
t = processor complexity (in number of transistors)
A = empirical constant of proportionality given in Table II
a = empirical constant given in Table II.

TABLE II
EXPERIMENTAL CONSTANTS

Group	A	a
CPU 81-82	6.69×10^{3}	0.783
CPU 82-85	5.16×10^{-3}	2.07
FPU	4.22×10^{5}	0.711

Equation 2.4 describes how, in some typical one-processor systems, processor speed is related to complexity. To apply these findings to a N-processor system of T transistors, we combine equations 2.3 and 2.4:

$$S_{proc} = A(TN^{-1})^{a} \qquad \text{(eqn 2.5)}$$

$$S_{proc} = AT^{a}N^{-a}$$

$$S_{proc} = K_{1}N^{-a}$$

where S_{proc} = processor speed (in bps throughput)
t = processor complexity (in number of transistors)
N = number of processors
A and a are constants given in Table II.

A family of "processor curves" may be used to describe the tradeoff between individual processor speed and the number of processors, constrained by a constant number of transistors. The tradeoffs are shown in Figure 2.2. For example, consider the curve labeled "CPU 82-85," which is based on a constant 10^6 transistors per chip. If these transistors are divided into 10 processors of 10^5 transistors each, Equation 2.4 predicts that each will produce about 116×10^6 bps of output. But if the chip is divided into more (for example 25) processors of 4×10^4 transistors each, then these less complex processors will be capable of only about 17.3×10^6 bps each.

When we superimpose these processor curves (Figure 2.2) with a family of "system" curves, generated by choosing several values of "R" in Equation 2.2, the result (Figures 2.3 and 2.4) yields a strategy for choosing N. Where the processor curve (describing what the processor *can* do) intersects the system curve (describing what each processor *must* do) determines the number of processors (N) into which the chip should be divided to yield that particular level of system throughput. For example, to achieve a system throughput of 10^9 bps, Figure 2.3 shows the chip should be divided into about 12 processors (point A). Yet choosing to partition the silicon into fewer, larger processors (point B) yields a higher system throughput of 2×10^9 bps.

In general, when processor speed is a *strong* function of complexity, that is when:

$$S_{proc} = At^a \qquad \text{with } a > 1 \qquad \text{(eqn 2.6)}$$

then S_{proc} is proportional to N^{-a} ($a > 1$) while $S_{req'd}$ is proportional to N^{-1}. Thus, S_{proc} falls *faster* than S_{reqd} as N increases. In this case, the highest performance will always result from choosing the lowest N possible, in other words $N = 1$. This strategy may be constrained for very large values of T--there may not be a processor design which can effectively use 10^7 transistors, for example. Also, the optimistic relationship of Equation 2.6 may not hold for large values of t.

On the other hand, when a *weak* relationship exists between speed and complexity, as shown in Figure 2.4, the best strategy is to select N as *large* as possible. As before, however, there are limits to this rule. It may be impractical to divide the computational task beyond a certain point. For example, a 256-point FFT probably

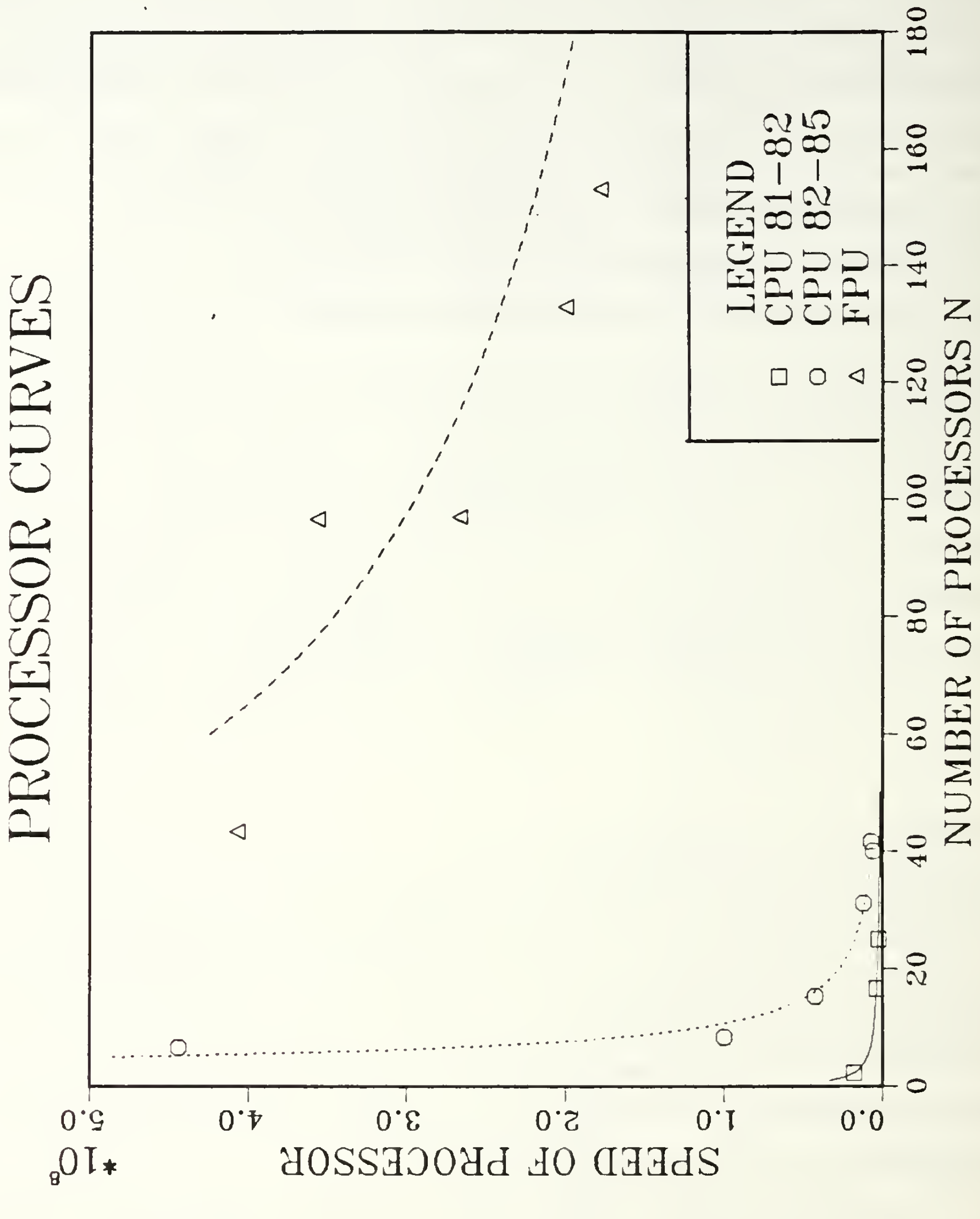

Figure 2.2 Processor Speed and Number of Processors Based on a Constant Number of Transistors.

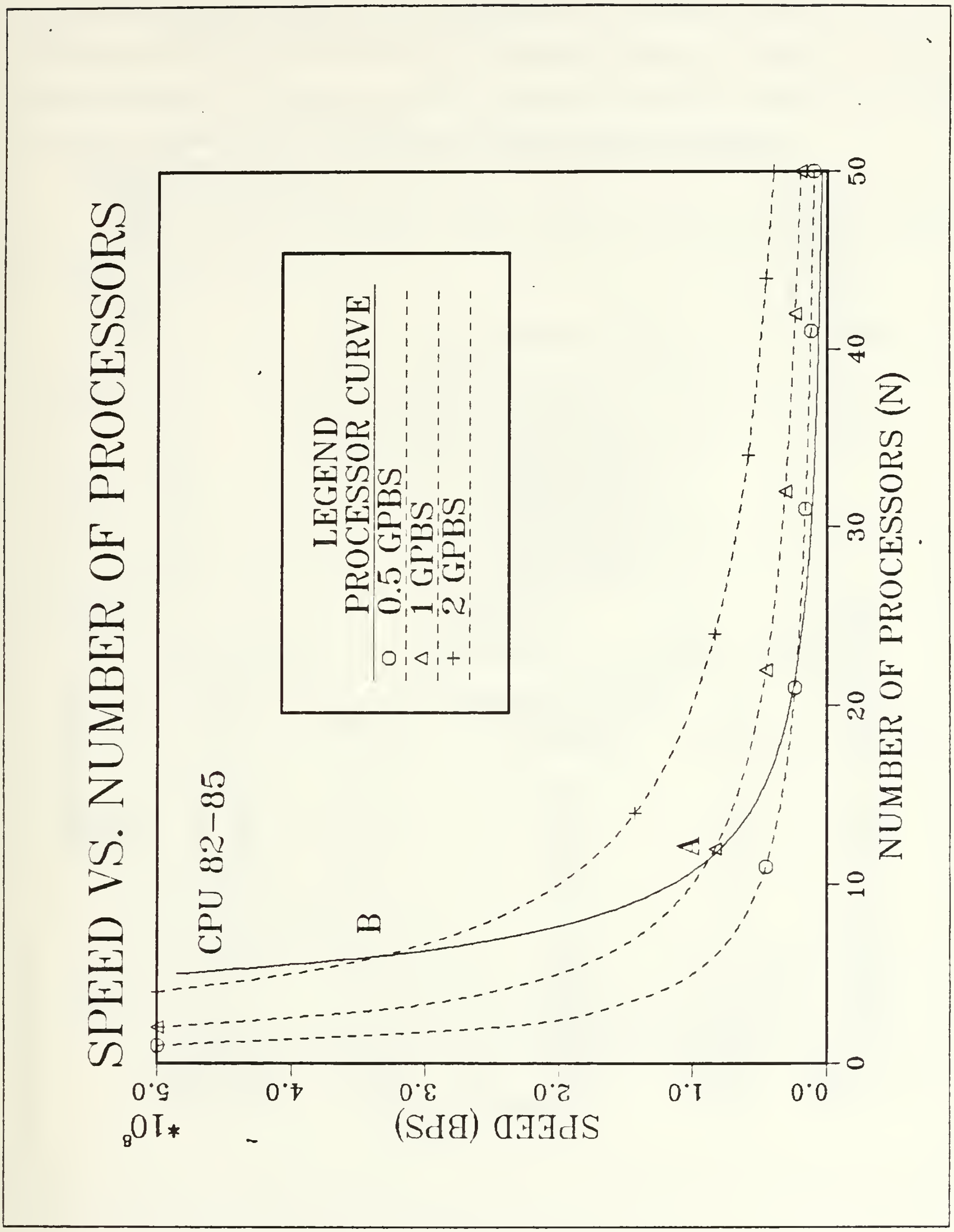

Figure 2.3 Relationship Between Processor Capability And System Requirements (Speed a Strong Function of Complexity).

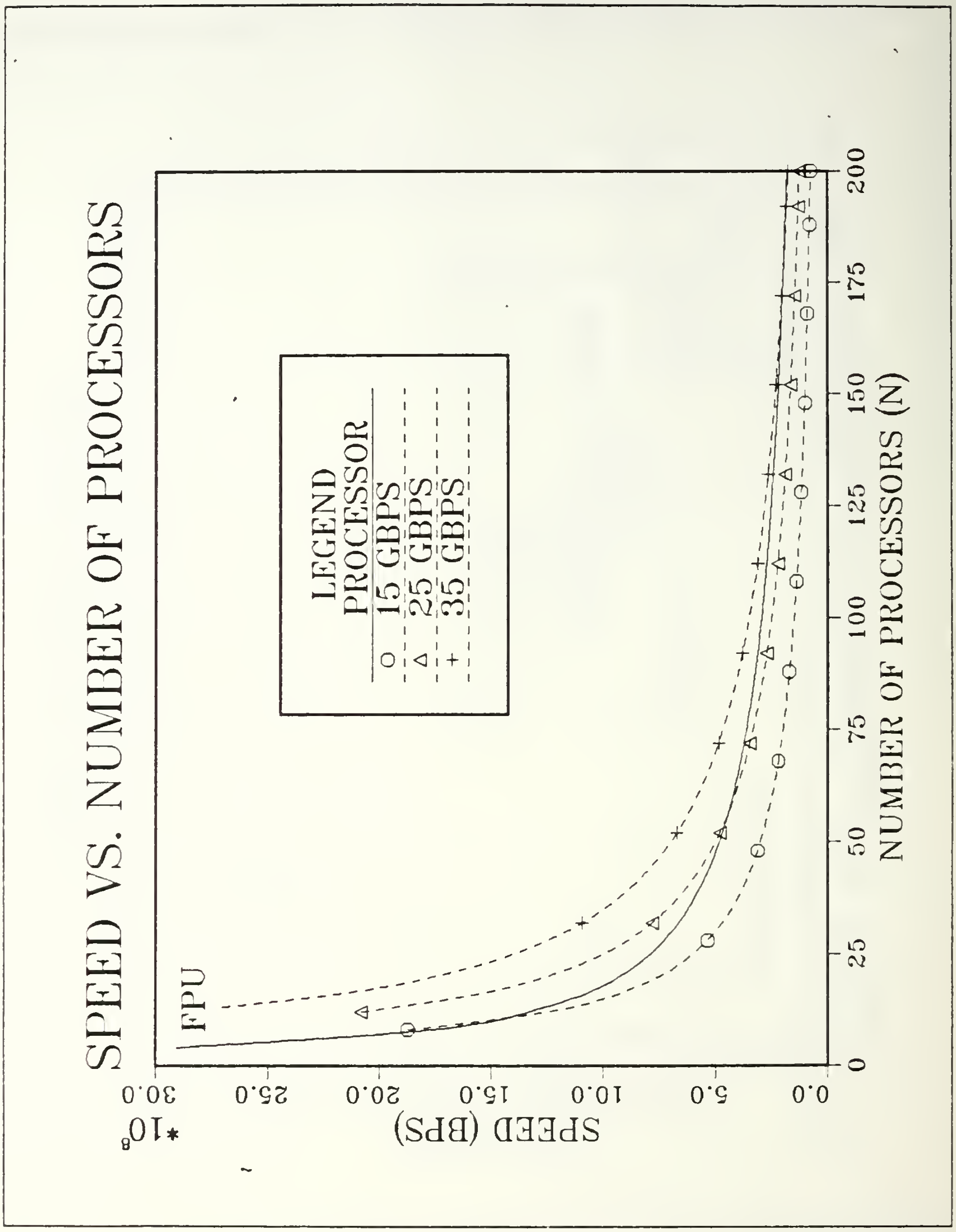

Figure 2.4 Relationship Between Processor Capability And System Requirements (Speed a Weak Function of Complexity).

can not be efficiently shared by more than 128 x 8 = 1024 processors.[1] Also, as N increases and t decreases, processors will eventually become too simple to function as microprocessors. For example, excessive reduction in processor complexity could yield a circuit unable to retain a data word or perform a basic calculation.

2. Power Constraints

Each chip can only dissipate a given amount of heat. The power available to any individual processor is:

$$p = PN^{-1} \tag{eqn 2.7}$$

where p = Power available to any one processor
N = number of processors
P = Total power available to the chip

TABLE III

SPECIFICATIONS OF SOME ACTUAL PROCESSORS

Group	Reference	Data Word (Bits)	Time Required for Multiplication (10^{-6} sec)	Bit Rate (10^{6} sec^{-1})	Power (watts)
NMOS CPU's	Ref. 14	32	8.30	3.86	1.50
	Ref. 15	16	6.25	2.56	1.40
	Ref. 16	32	1.80	17.8	7.00
	Ref. 17	32	5.50	5.82	0.75
	Ref. 18	32	4.50	7.11	2.00
	Ref. 19	32	2.70	11.85	2.10
CMOS FPU's	Ref. 23	16	.090	178.	0.125
	Ref. 25	16	.060	267.	0.065
	Ref. 26	16	.045	356.	0.100
	Ref. 27	16	.027	593.	0.15
	Ref. 28	16	.079	405.	0.195
	Ref. 29	32	.100	320.	0.40
	Ref. 30	16	.065	246.	0.200
	Ref. 31	32	.100	320.	0.500
	Ref. 32	16	.065	246.	0.10
	Ref. 33	16	.130	123.	0.275

---

[1] There are $256 \div 2 = 128$ processors per stage and $\log_2(256) = 8$ stages.

Examining the relationship between processor speed and power in the light of data from actual processors, (Table III and Figure 2.5) there in no clear trend evident in Figure 2.5. In particular, there is a great deal of scatter in the CMOS multiplier chip data. This may be due to differences in the way researchers report power dissipation data; for example, some may report only the power consumption of the computational segment, while others report the power used by the entire chip, including bus drivers. In spite of these limitations, one interpretation of the power/throughput data is:

$$S_{proc} = Bp^{b} \qquad \text{(eqn 2.8)}$$

where S_{proc} = processor speed (in bps throughput)
p = processor power (in watts)
B and b are empirical constants given in Table IV.

Therefore, combining equations 2.5 and 2.8 as before:

$$S_{proc} = B(PN^{-1})^{b} \qquad \text{(eqn 2.9)}$$

$$S_{proc} = BP^{b}N^{-b}$$

$$S_{proc} = K_{2}N^{-b}$$

where S_{proc} = processor speed (in bps throughput)
p = processor power (in watts)
N = number of processors
B and b are empirical constants given in Table IV.

Figure 2.6 shows the relationship described in equation 2.9, namely, the tradeoff of individual processor speed against the number of processors, constrained this time by a constant power level, as required by equation 2.7. Since, for the group of actual processors examined,

$$S_{proc} = Bp^{b} \qquad \text{with } b < 1 \qquad \text{(eqn 2.10)}$$

Figure 2.7 shows that the best strategy is to select N as *large* as possible.

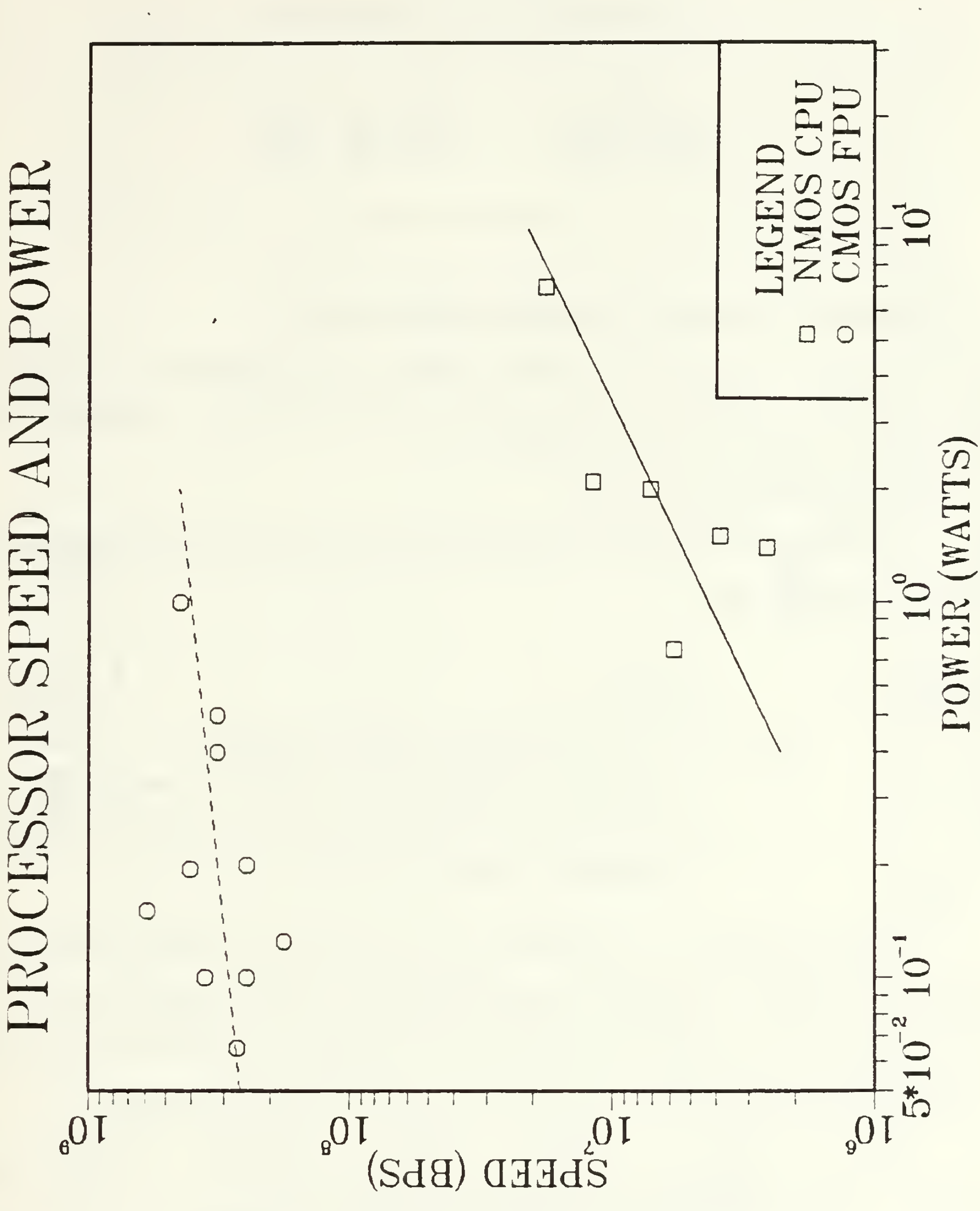

Figure 2.5 Processor Speed and Power (Experimental).

TABLE IV

EXPERIMENTAL CONSTANTS

Group	B	b
NMOS cpu's	4.27×10^6	0.693
CMOS fpu's	3.43×10^8	0.099

B. MINIMUM CHIP SIZE FOR OM APPLICATION

How large (in terms of transistor count, heat dissipation, and number of processors) would a chip have to be in order to produce sufficient throughput to justify the use of the Optoelectronic Multiplexer?

1. Minimum Transistor Count

Assuming the individual processors are of low complexity (like the FPU group of Figure 2.1) implies that:

$$S_{proc} = At^a \quad \text{with } a < 1 \qquad \text{(eqn 2.11)}$$

where S_{proc} = processor speed (in bps throughput)

t = processor complexity (in number of transistors)

A = empirical constant of proportionality given in Table II

a = empirical constant , here < 1.

For this group, the discussion in the previous section shows that the maximum throughput is achieved by partioning the available silicon into the largest number of processors possible, limited by the minimum complexity of the simplest processor design.[2] Therefore:

$$N_{max} = Tt_{min}^{-1} \qquad \text{(eqn 2.12)}$$

where T = total number of transistors on chip

t_{min} = complexity of the simplest processor design, measured in transistor

N_{max} = number of simple processors possible on chip of T transistors

---

[2] While the components of systolic arrays are less complex than the assumed simplest processor, this research did not study the performance of such ICs--accordingly they are not considered here.

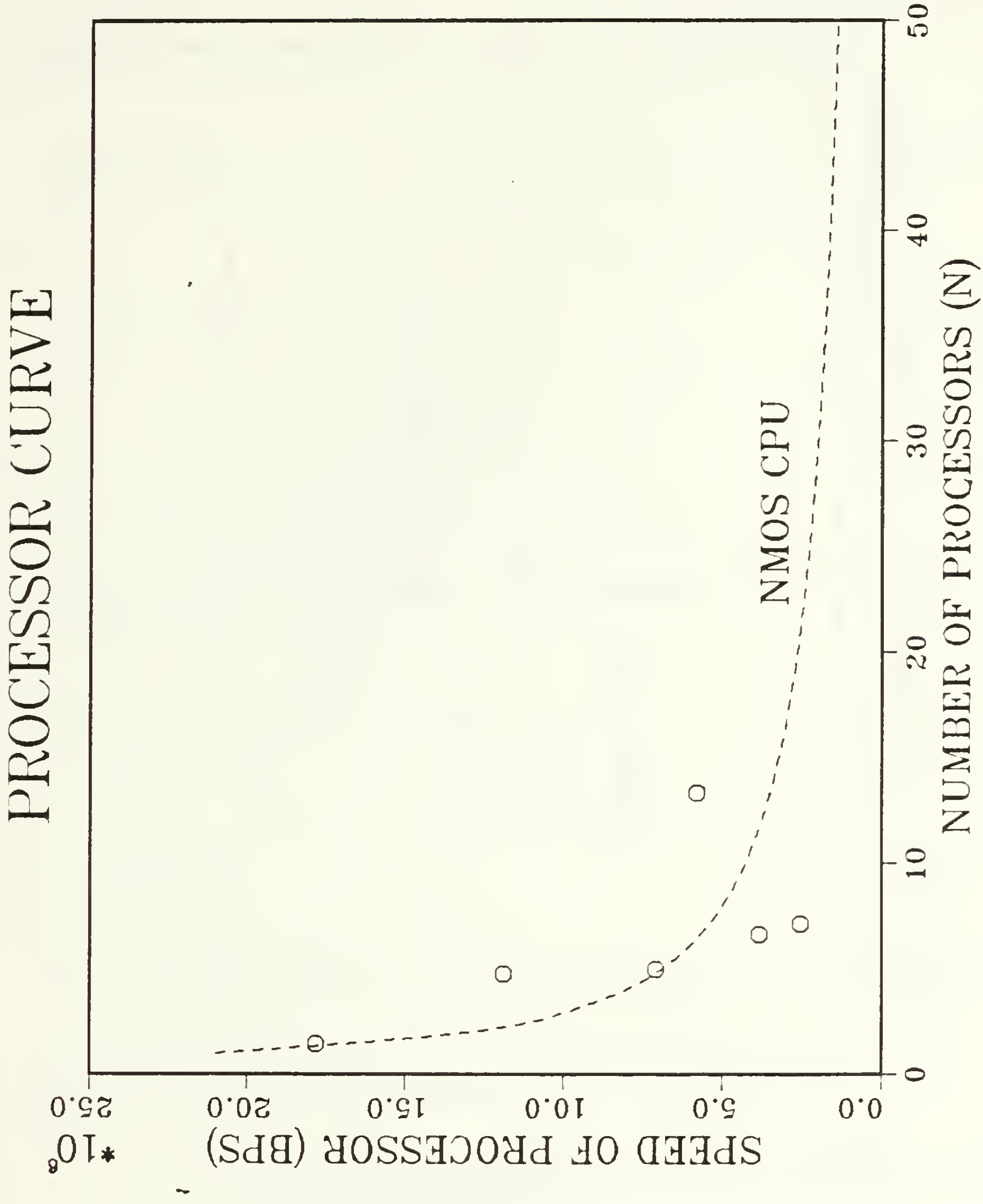

Figure 2.6 Processor Speed and Number of Processors Based on a Constant Chip Power Level.

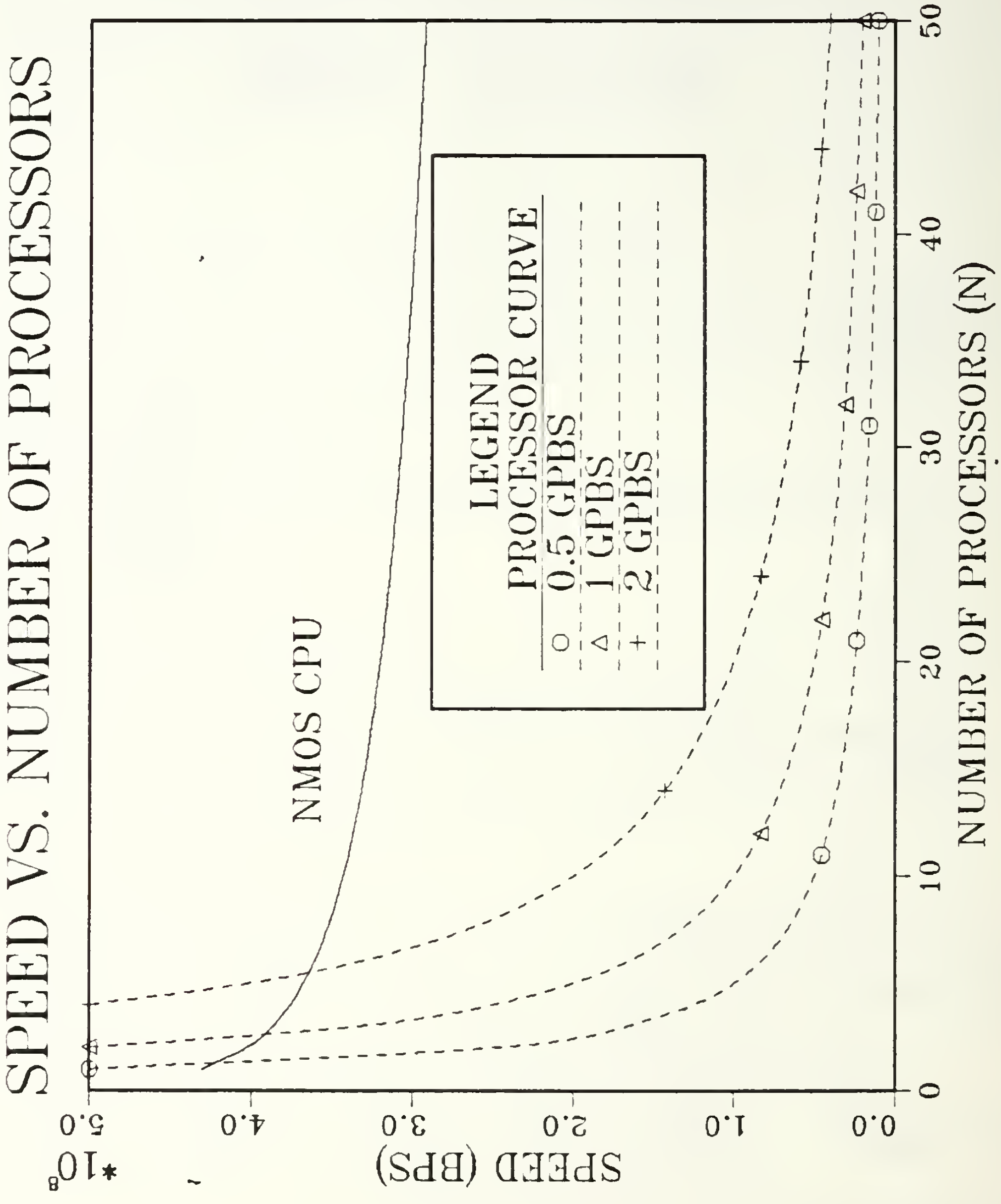

Figure 2.7 Relationship Between Processor Capability And System Requirements (Speed a Weak Function of Power).

Since each processor produces an output of S_{proc} bits per second and there are N_{max} processors, the system throughput is:

$$S_{sysmax} = N_{max}S_{proc} \qquad \text{(eqn 2.13)}$$

$$= N_{max}At_{min}^{a} \qquad \text{(eqn 2.14)}$$

$$= [Tt_{min}^{-1}]At_{min}^{a} \qquad \text{(eqn 2.15)}$$

$$= TAt_{min}^{a-1} \qquad \text{(eqn 2.16)}$$

Defining S_{om} to be the minimum system throughput for which use of the OM is justified leads to:

$$S_{om} = TAt_{min}^{a-1} \qquad \text{(eqn 2.17)}$$

$$T_{min} = S_{om}[t_{min}^{1-a}]A^{-1} \qquad \text{(eqn 2.18)}$$

where t_{min} = minimum number of transistors on chip for OM usage to be justified

To estimate the value of T_{min} assume:

$t_{min} = 4 \times 10^3$ transistors (lower end of FPU group in Table I)
$A = 4.22 \times 10^5$ (Table II)
$a = 0.711$ (Table II)
$S_{om} = 3 \times 10^9$ bps (Curently the upper range of conventional multiplexers.) [Refs. 34,35,36,37]

Therefore:

$$T_{min} = 92000 \text{ transistors}$$
$$N_{max} = 13 \text{ processors}$$

Thus, since processors with transistor counts > T_{min} are already in existence [Ref. 21], it seems that an OM-based single chip multiple processor is feasible with respect to the number of transistors required.

2. Minimum Power Dissipation

What is the minimum heat dissipation of a multi-processor chip which would yield throughput in the OM range?

$$N_{max} = Pp_{min}^{-1} \qquad \text{(eqn 2.19)}$$

where P = Total power dissipation of the chip (watts)

p_{min} = power used by the simplest processor design, measured in watts

N_{max} = number of simple processors possible on chip of P watts

$$S_{sysmax} = N_{max}S_{proc} \qquad \text{(eqn 2.20)}$$

Substituting from Equation 2.8,

$$S_{sysmax} = N_{max}Bp_{min}^{b} \qquad \text{(eqn 2.21)}$$

And, substituting for N_{max} from Equation 2.19,

$$S_{sysmax} = [Pp_{min}^{-1}]Bp_{min}^{b} \qquad \text{(eqn 2.22)}$$

$$S_{sysmax} = PBp_{min}^{b-1} \qquad \text{(eqn 2.23)}$$

Defining S_{om} to be the minimum system throughput for which use of the OM is justified leads to:

$$S_{om} = PBp_{min}^{b-1} \qquad \text{(eqn 2.24)}$$

$$P_{min} = S_{om}[p_{min}^{1-b}]B^{-1} \qquad \text{(eqn 2.25)}$$

where P_{min} = minimum power dissipation of the chip for OM usage to be justified

To estimate the value of P_{min} assume:

$p_{min} = 0.10$ watts (lower end of CMOS FPU group in Table III)
$B = 3.43 \times 10^8$ (Table IV)
$b = 0.099$ (Table IV)
$S_{om} = 3 \times 10^9$ bps

Therefore:

$P_{min} = 1.10$ watts
$N_{max} = 11$ processors

This power level is quite reasonable, and it would seem that from the standpoint of heat dissipation an OM-based multiple processor chip is feasible.

III. THE NEED FOR A HIGH-SPEED MULTIPLEXER

Chapter II demonstrated that current technology could produce a chip whose throughput would exceed the capacity of conventional multiplexer technology. But why consider serial communications and multiplexers at all? Why not exchange data with the chip in parallel via pins or leads?

A. PROCESSOR POWER LIMITED BY COMMUNICATION PATH

We have seen that future high-density IC's may be optimally structured as a bank of many processors, each of moderate capability. However, even if manufacturers can achieve sufficient circuit density to fabricate a multi-processor chip, such a device might not be practical due to the large number of leads needed to communicate with each processor from off-chip. For example, imagine an N-processor IC designed to compute a 2N-point Fast Fourier Transform (FFT). During the computation, the IC must read in, then write out, 2N complex output words, or 4N real words. Assuming a 40 bit word size, and using the same pins for input and output, we can see this IC would need:

$$\left[\frac{\text{40 leads}}{\text{word}}\right] \times [\text{ 4N words}] = \text{160N leads} \qquad \text{(eqn 3.1)}$$

How large a package will we need to handle all these leads? Using a Pin-Grid Array (PGA) package with pins spaced every 0.1 inch, the area of the package is:

$$\text{Area} = \frac{[\text{ 160N leads}]}{\left[\frac{\text{10 leads}}{\text{25.4 mm}}\right]^2} = \text{1032N mm}^2 \qquad \text{(eqn 3.2)}$$

For illustrative purposes we can estimate the area of the silicon chip in this package by assuming the chip size of the processor is approximately the same as that of the processor recently reported by the Matsushita Corporation of Osaka. [Ref. 28] Their processor performs a 32 bit floating point multiplication in about 75 nsec and is 32.6 mm^2 in area. A chip containing N of these processors would occupy about 32.6N mm^2 of silicon. Thus, the ratio of silicon area to package area in our hypothetical IC is:

$$\text{Ratio of } \frac{\text{Silicon Area}}{\text{Package Area}} = \frac{[\ 32.6N]}{[\ 1032N]} = 3.2\ \% \qquad \text{(eqn 3.3)}$$

As IC fabrication technology improves, this waste of space gets even worse. A new production technique enabling manufacturers to produce circuits in half the silicon area previously required would permit us to double "N" without increasing the silicon area. Yet package area *would* double, due to increased pinout requirements. Once some maximum package size is reached, further improvements in circuit density do us no good--we simply can not communicate with more processors. As one researcher stated, "the technology has become increasingly constrained by packaging limitations" [Ref. 38].

Increasing lead density will produce some relief from this communication limit, but can not be pursued beyond some maximum without excessive fabrication cost. We are faced, then, with some maximum package size and maximum lead density, implying an eventual limit on the number of leads a single IC can have.

Given this eventual limit on the number of simultaneous off-chip communication paths, Rent's Rule [Ref. 12:p. 235]

$$P = 4G^{0.6} \qquad \text{(eqn 3.4)}$$

where P = Number of chip pads or leads
G = Number of gates on the chip

would seem to imply that if the number of paths (P) is limited, then so is the number of gates (G) and, therefore microprocessor complexity and computational power.

This ultimate limit on non-multiplexed designs is not precisely defined. Neither maximum package size nor maximum lead density have yet been reached, and industry experts are wary of predicting when they might be. In addition, the switch to multiplexed designs will probably occur over a range of processor densities and complexities, influenced by market factors (there will be few customers for very large packages) and manufacturing realities (specialized chip sizes mean more expensive chip handling equipment) as well as the theoretical factors described above.

For all these reasons, large ICs composed of multiple processors will require too many pins to use a conventional parallel-transfer scheme with pins or leads. Instead a serial communications link must be considered, and as shown in Chapter II, the speeds required will exceed the capacity of conventional multiplexers.

IV. SYSTEM ARCHITECTURE BASED ON SERIAL COMMUNICATION

Chapters II and III demonstrate that, in the next generation of ICs, a microprocessor may very well be organized as a bank of smaller processors, all sharing a relatively few pins through a high-speed multiplexer. But:

- What on-chip data flow architecture should be employed among these processors?
- How can a serial data stream be distributed among N processors?
- What are the detailed structures of the elements which make up an OM-based architecture?

A. ON-CHIP DATA FLOW ARCHITECTURE

How should a N-processor chip be organized? The ideal structure will vary with the application; this discussion considers one specific application--computing FFT's. The number of processors required to compute a given size FFT will depend on whether processors are "reused," that is whether a processor bank's outputs are shuffled and returned to the same processors (reused) or directed to the next bank of processors (pipelined). Reusing processors allows a given FFT to be computed with fewer processors, but takes more time. The architectures asssociated with both reuse and pipeline strategies are discussed in the following sections.

1. Pipeline Architecture

Assuming that the throughput of the system is to be maximized, there will be no "reuse" of processors. That is:

- each processor performs only a two point FFT "butterfly"
- a new bank of processors performs each stage of the computation in a pipeline strategy.

The most straightforward architecture for N processors is a N × 1 column. How would this grouping affect data flow among the processors? As an example, when the task is a 16 point FFT, the processors must exchange data as shown in Figures 4.1 and 4.2. Dividing up the 32 processors shown in Figure 4.2 into 4 × 1 chips forces 80 data words to cross chip boundaries during the computation, as shown in Figure 4.3.

Re-organizing the four processors on each chip into a 2 × 2 matrix (Figure 4.4) results in only 48 words crossing chip boundaries, thereby improving the system's throughput, since off-chip communication delay is lessened, and reducing the demands on the communications network.

The 2 × 2 structure is more efficient because it is the structure of a four point FFT. In a sense, the 2 × 2 structure performs all the computations possible on the four points it receives, while the 4 × 1 array, receiving eight points, must hand off its data only partially "chewed."

There are many such matrices, each corresponding to a particular FFT. For example, Figure 4.2 suggests that a 32 point processor IC designed for FFT computation would best be configured as a 8 × 4 matrix. In general, the matrix dimensions are:

$$2^{n-1} \times n \qquad \text{where } n = 1,2,3,\ldots \qquad \text{(eqn 4.1)}$$

2. Reuse Architecture

The number of processors required by a pipeline architecture to compute a P-point FFT is:

$$P/2 \times \log_2 P \qquad \text{(eqn 4.2)}$$

This number of processors may prove to be impractical or simply too expensive, or we may not need the ultimate throughput achievable by the pipeline architecture, yet still need more throughput than that provided by a uniprocessor. Also, it may be desirable to adapt an existing pipeline system to compute larger FFTs--without adding processors. In each of these cases, reusing processors in the computation enables the designer to tradeoff system throughput for design complexity and cost. How are data exchanged among processors in a reuse architecture?

The computations shown in Figure 4.2 still must be performed, but now instead of each block representing an actual processor, it represents a job that some processor will have to perform. For example, consider a 16-point FFT performed with two 4-processor ICs. Figure 4.5 shows the data exchange for this example if the four-processor ICs are organized as 4 × 1 vectors.

As shown in Table V, even though a chip processes eight points every frame, it only transmits four points per frame--keeping half its data onboard for further processing with the half it will receive from the other IC. This assumes that some on-chip communication path exists to enable processors to exchange data.

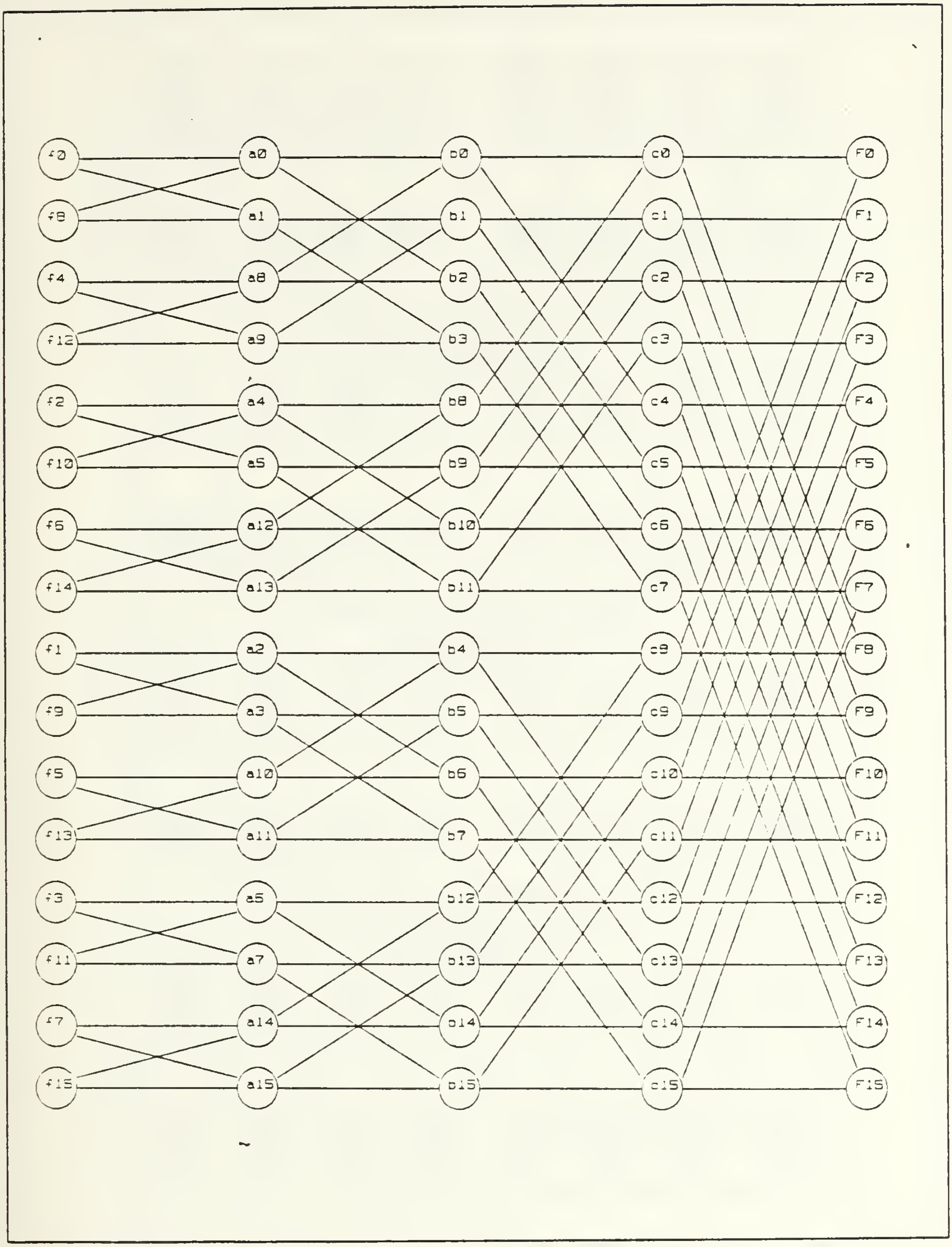
f0
f8
f4
f12
f2
f10
f6
f14
f1
f9
f5
f13
f3
f11
f7
f15
a0
a1
a8
a9
a4
a5
a12
a13
a2
a3
a10
a11
a5
a7
a14
a15
b0
b1
b2
b3
b8
b9
b10
b11
b4
b5
b6
b7
b12
b13
b14
b15
c0
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12
c13
c14
c15
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15

Figure 4.1 Sixteen Point Fast Fourier Transform [Ref. 3 pp. 2019-22].

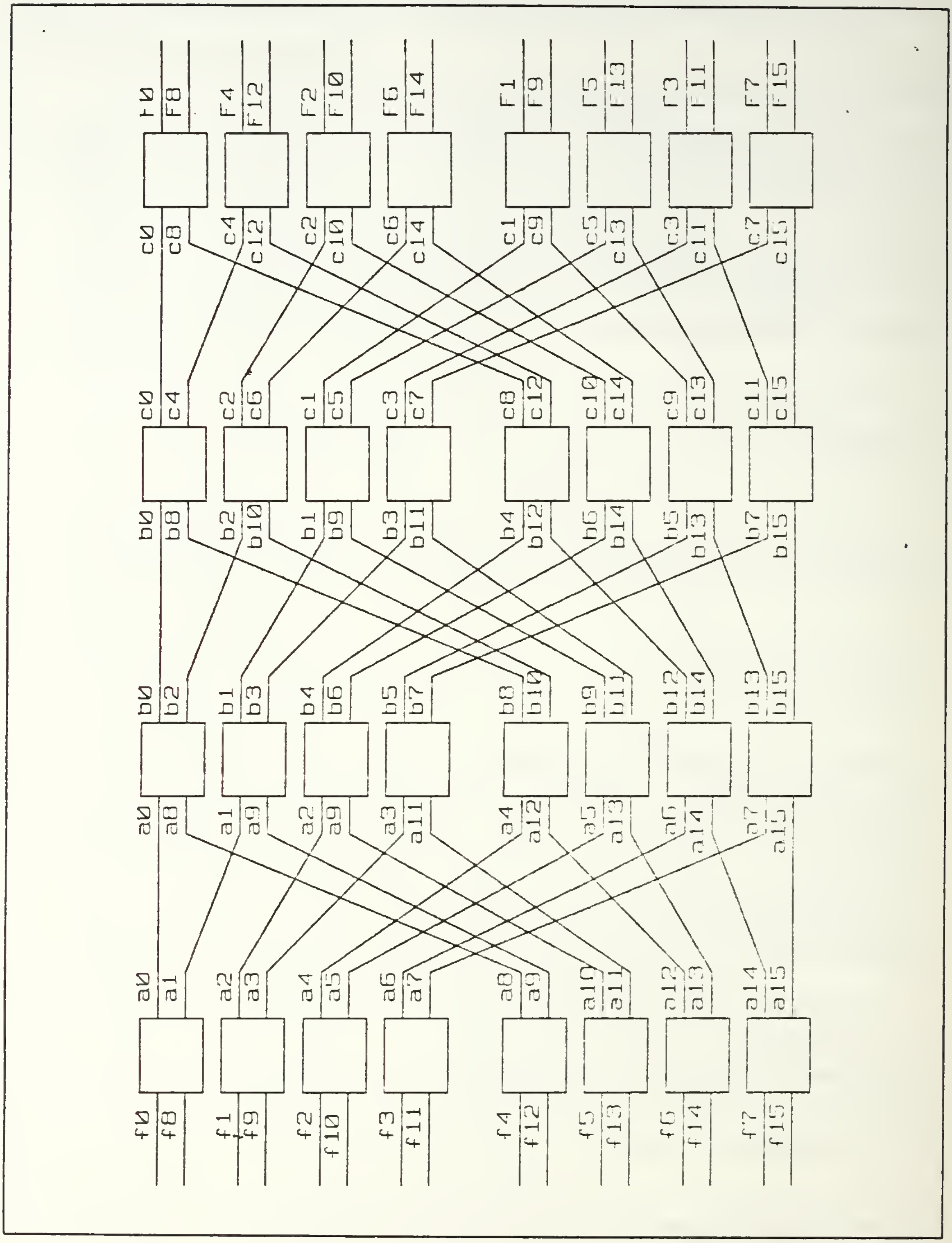
f0
f8
f1
f9
f2
f10
f3
f11
f4
f12
f5
f13
f6
f14
f7
f15
a0
a1
a2
a3
a4
a5
a6
a7
a8
a9
a10
a11
a12
a13
a14
a15
a0
a8
a1
a9
a2
a9
a3
a11
a4
a12
a5
a13
a6
a14
a7
a15
b0
b2
b1
b3
b4
b6
b5
b7
b8
b10
b9
b11
b12
b14
b13
b15
b0
b8
b2
b10
b1
b9
b3
b11
b4
b12
b6
b14
b5
b13
b7
b15
c0
c4
c2
c6
c1
c5
c3
c7
c8
c12
c10
c14
c9
c13
c11
c15
c0
c8
c4
c12
c2
c10
c6
c14
c1
c9
c5
c13
c3
c11
c7
c15
F0
F8
F4
F12
F2
F10
F6
F14
F1
F9
F5
F13
F3
F11
F7
F15

Figure 4.2 Sixteen Point Fast Fourier Transform Pipline Implementation.

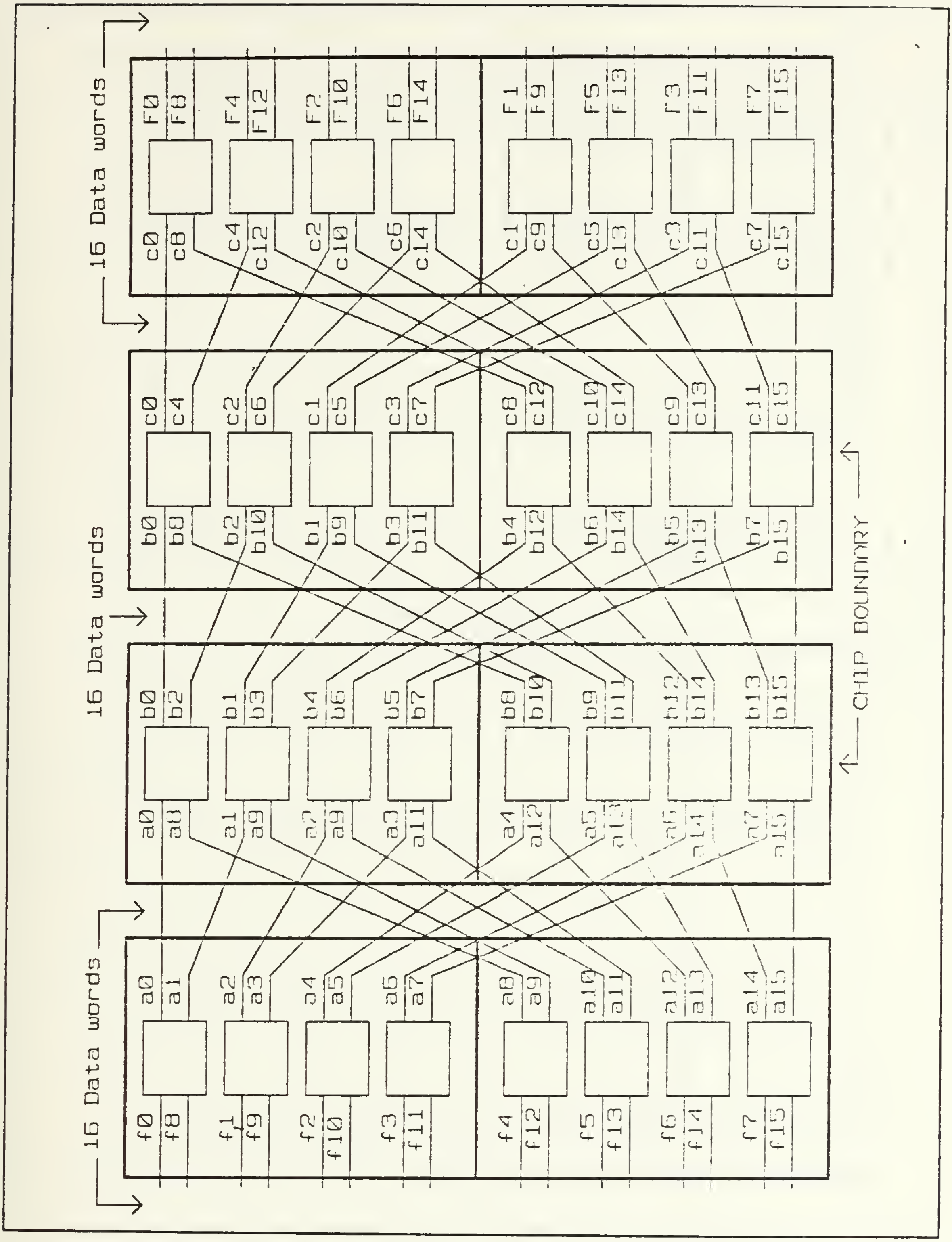
16 Data words
16 Data words
16 Data words
16 Data words
CHIP BOUNDARY
f0 f8 f1 f9 f2 f10 f3 f11 f4 f12 f5 f13 f6 f14 f7 f15
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
a0 a8 a1 a9 a2 a10 a3 a11 a4 a12 a5 a13 a6 a14 a7 a15
b0 b2 b1 b3 b4 b6 b5 b7 b8 b10 b9 b11 b12 b14 b13 b15
b0 b8 b2 b10 b1 b9 b3 b11 b4 b12 b6 b14 b5 b13 b7 b15
c0 c4 c2 c6 c1 c5 c3 c7 c8 c12 c10 c14 c9 c13 c11 c15
c0 c8 c4 c12 c2 c10 c6 c14 c1 c9 c5 c13 c3 c11 c7 c15
F0 F8 F4 F12 F2 F10 F6 F14 F1 F9 F5 F13 F3 F11 F7 F15

Figure 4.3 Sixteen Point Fast Fourier Transform Performed by 4 × 1 Chips.

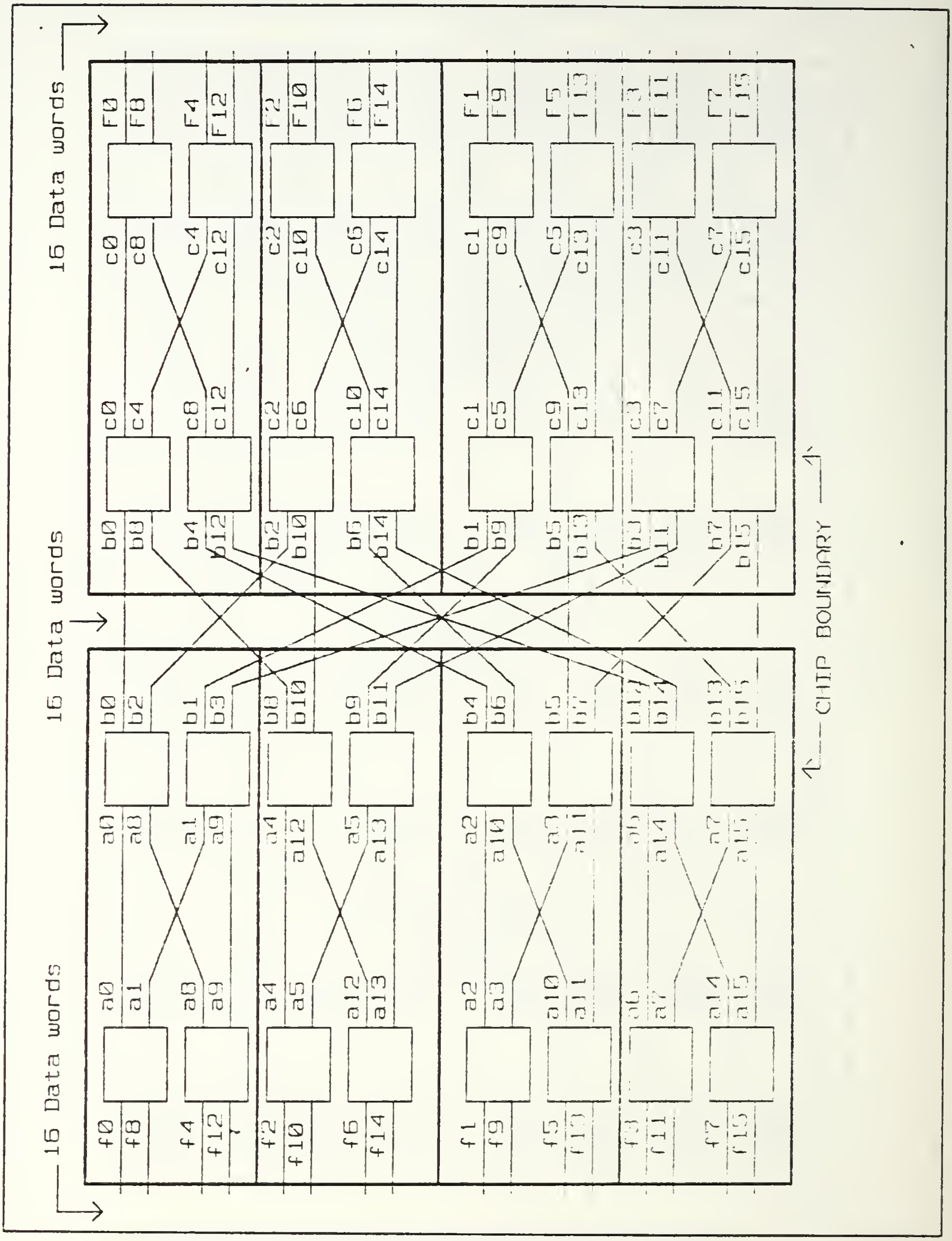
16 Data words
16 Data words
16 Data words
CHIP BOUNDARY
f0
f8
f4
f12
f2
f10
f6
f14
f1
f9
f5
f13
f3
f11
f7
f15
a0
a1
a8
a9
a4
a5
a12
a13
a2
a3
a10
a11
a6
a7
a14
a15
b0
b2
b1
b3
b8
b10
b9
b11
b4
b6
b5
b7
b12
b14
b13
b15
c0
c4
c8
c12
c2
c6
c10
c14
c1
c5
c9
c13
c3
c7
c11
c15
F0
F8
F4
F12
F2
F10
F6
F14
F1
F9
F5
F13
F3
F11
F7
F15

Figure 4.4 Sixteen Point Fast Fourier Transform Performed by 2 × 2 Chips.

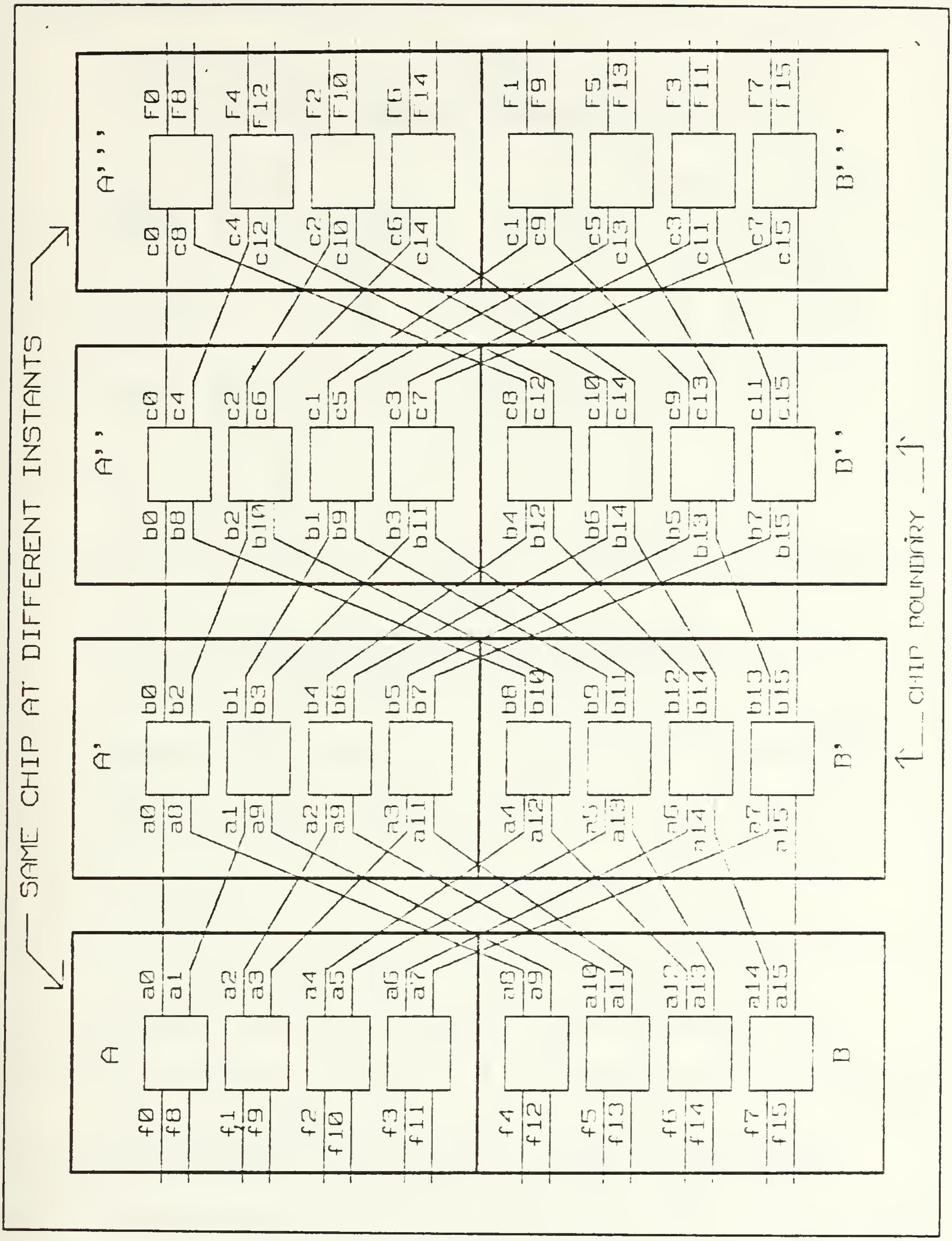
SAME CHIP AT DIFFERENT INSTANTS
CHIP BOUNDARY
A
B
A'
B'
A''
B''
A'''
B'''
f0 f8 f1 f9 f2 f10 f3 f11 f4 f12 f5 f13 f6 f14 f7 f15
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
a0 a8 a1 a9 a2 a9 a3 a11 a4 a12 a5 a13 a6 a14 a7 a15
b0 b2 b1 b3 b4 b6 b5 b7 b8 b10 b9 b11 b12 b14 b13 b15
b0 b8 b2 b10 b1 b9 b3 b11 b4 b12 b6 b14 b5 b13 b7 b15
c0 c4 c2 c6 c1 c5 c3 c7 c8 c12 c10 c14 c9 c13 c11 c15
c0 c8 c4 c12 c2 c10 c6 c14 c1 c9 c5 c13 c3 c11 c7 c15
F0 F8 F4 F12 F2 F10 F6 F14 F1 F9 F5 F13 F3 F11 F7 F15

Figure 4.5 Sixteen Point Fast Fourier Transform
4 × 1 Reuse Architecture.

TABLE V

INTER-PROCESSOR COMMUNICATIONS
4 X 1 REUSE ARCHITECTURE

Stage #	Chip A								Chip B							
	Receives				Transmits				Receives				Transmits			
1	f0 f1	f8 f9	f2 f3	f10 f11	a1	a5	a6	a7	f4 f6	f12 f14	f5 f7	f13 f15	a8	a9	a10	a11
2	a8	a9	a10	a11	b4	b5	b6	b7	a4	a5	a6	a7	b8	b9	b10	b11
3	b8	b9	b10	b11	c1	c3	c5	c7	b4	b5	b6	b7	c8	c10	c12	c14
4	c8	c10	c12	c14	F0 F2	F8 F10	F4 F6	F12 F14	c1	c3	c5	c7	F1 F5	F9 F13	F3 F7	F11 F15

TABLE VI

INTER-PROCESSOR COMMUNICATIONS
2 X 2 REUSE ARCHITECTURE

Stage #	Chip A								Chip B							
	Receives				Transmits				Receives				Transmits			
1	f0	f8	f4	f12		---			f1	f9	f5	f13		---		
2	f2	f10	f6	f14	b1	b3	b9	b11	f3	f11	f7	f15	b4	b12	b6	b14
3	b4	b6	b12	b14	F0 F2	F8 F10	F4 F6	F12 F14	b1	b3	b9	b11	F1 F5	F9 F13	F3 F7	F11 F15

As an alternative, consider the same 16-point FFT computed by two 4-processor ICs, this time organized as 2 × 2 matrices, as shown in Figure 4.6 and Table VI.

Because each IC is only two processors "wide," a single IC can only accept four data points at a time. This creates an awkward data flow--the source delivers only half the input vector, waits, then delivers the other half. Each chip must store the output of its first computation while processing the second half of the input vector. However, the number of data points exchanged between chips is sharply reduced from 24 for the 4 × 1 case to 8 for the 2 × 2 case.

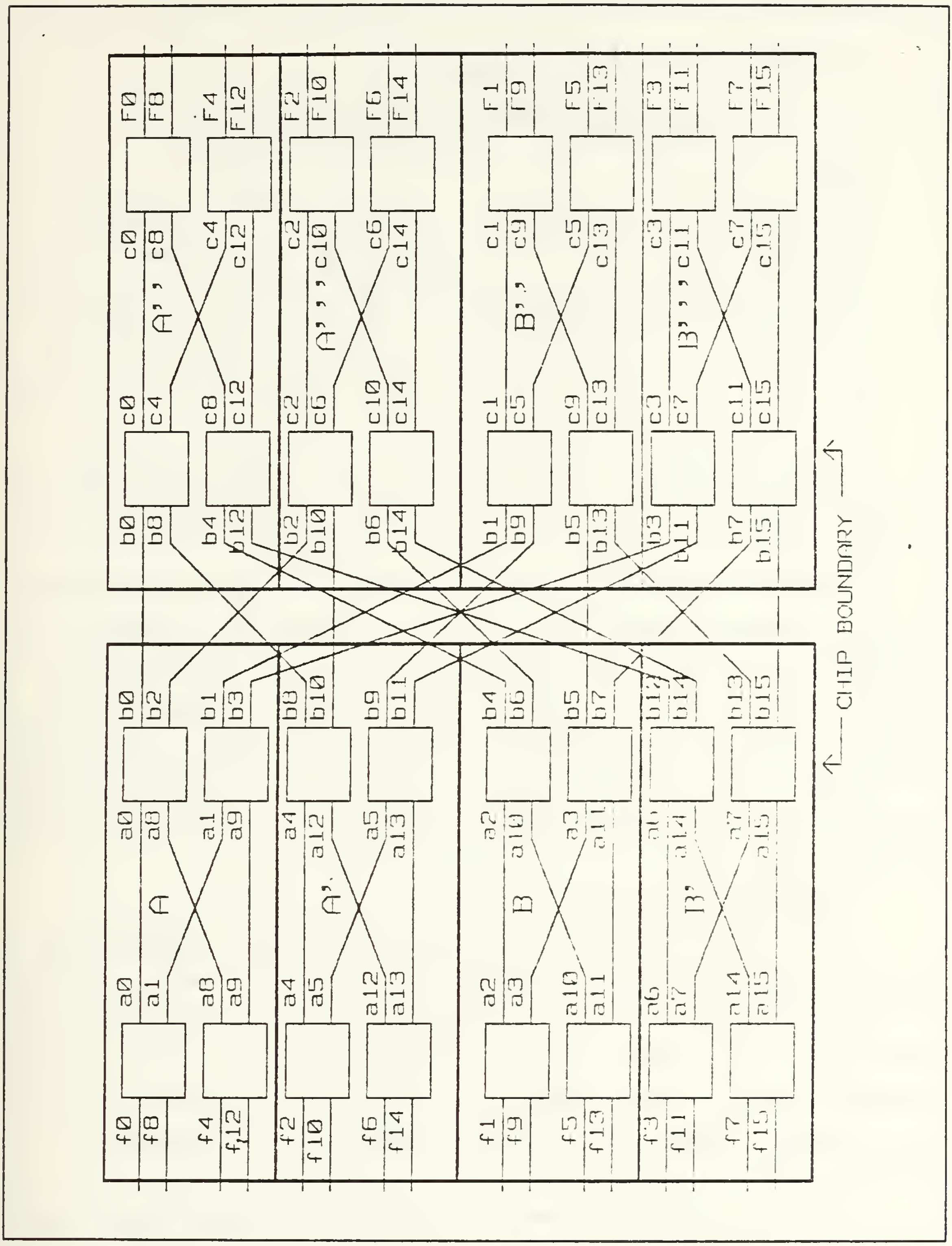
f0
f8
f4
f12
f2
f10
f6
f14
f1
f9
f5
f13
f3
f11
f7
f15
a0
a1
a8
a9
a4
a5
a12
a13
a2
a3
a10
a11
a6
a7
a14
a15
A
A'
B
B'
a0
a8
a1
a9
a4
a12
a5
a13
a2
a10
a3
a11
a6
a14
a7
a15
b0
b2
b1
b3
b8
b10
b9
b11
b4
b6
b5
b7
b12
b14
b13
b15
CHIP BOUNDARY
b0
b8
b4
b12
b2
b10
b6
b14
b1
b9
b5
b13
b3
b11
b7
b15
c0
c4
c8
c12
c2
c6
c10
c14
c1
c5
c9
c13
c3
c7
c11
c15
A''
A'''
B''
B'''
c0
c8
c4
c12
c2
c10
c6
c14
c1
c9
c5
c13
c3
c11
c7
c15
F0
F8
F4
F12
F2
F10
F6
F14
F1
F9
F5
F13
F3
F11
F7
F15

Figure 4.6 Sixteen Point Fast Fourier Transform 2 × 2 Reuse Architecture.

TABLE VII

INTER-PROCESSOR COMMUNICATIONS
MODIFIED 4 X 1 REUSE ARCHITECTURE

Stage	Chip A								Chip B							
#	Receives				Transmits				Receives				Transmits			
1	f0 f4	f8 f12	f2 f6	f10 f14	b1	b3	b9	b11	f1 f5	f9 f13	f3 f7	f11 f15	b4	b12	b6	b14
2	b4	b6	b12	b14	F0 F2	F8 F10	F4 F6	F12 F14	b1	b3	b9	b11	F1 F5	F9 F13	F3 F7	F11 F15

A third organization of these four processors permits transmission of the entire data vector (as in the 4 × 1 chip) and minimizes the data exchange (as in the 2 × 2 chip). Its structure is shown in Figure 4.7 and Table VII.

This structure, possible only if processors are reused, maximizes the "width" of the chip while preserving the communication advantages of a "deep" chip. As discussed in the previous section, these advantages stem from performing all the calculations possible on a given data set before releasing is to another chip. By not allowing "partially chewed" data off the chip, the number of data to be exchanged between chips at each stage is minimized. In general, an N-processor chip with this reuse architecture can perform a 2N-point FFT if organized as an N × 1 vector which performs $1+\log_2 N$ stages.

3. Interleaving Data Sets

The efficiency and throughput of any of these reuse architectures can be improved through interleaving data sets--that is, delivering new data to the processors to work on while they wait for the communications link to recycle their intermediate outputs back to their inputs. Consider the progress of a 16-point FFT calculation performed by eight processors organized as in Figure 4.7. The processor wait time is cleary evident in Figure 4.8, in which the data sets are not interleaved. In this example, the throughput is one FFT per ($4T_{CALC} + T_{XFR}$).

In Figure 4.9, however, a new data set is delivered to the processors while they wait for the results of the first phase of the calculation to be recirculated. In this interleaved case, processors are never allowed to be idle. For this example, throughput

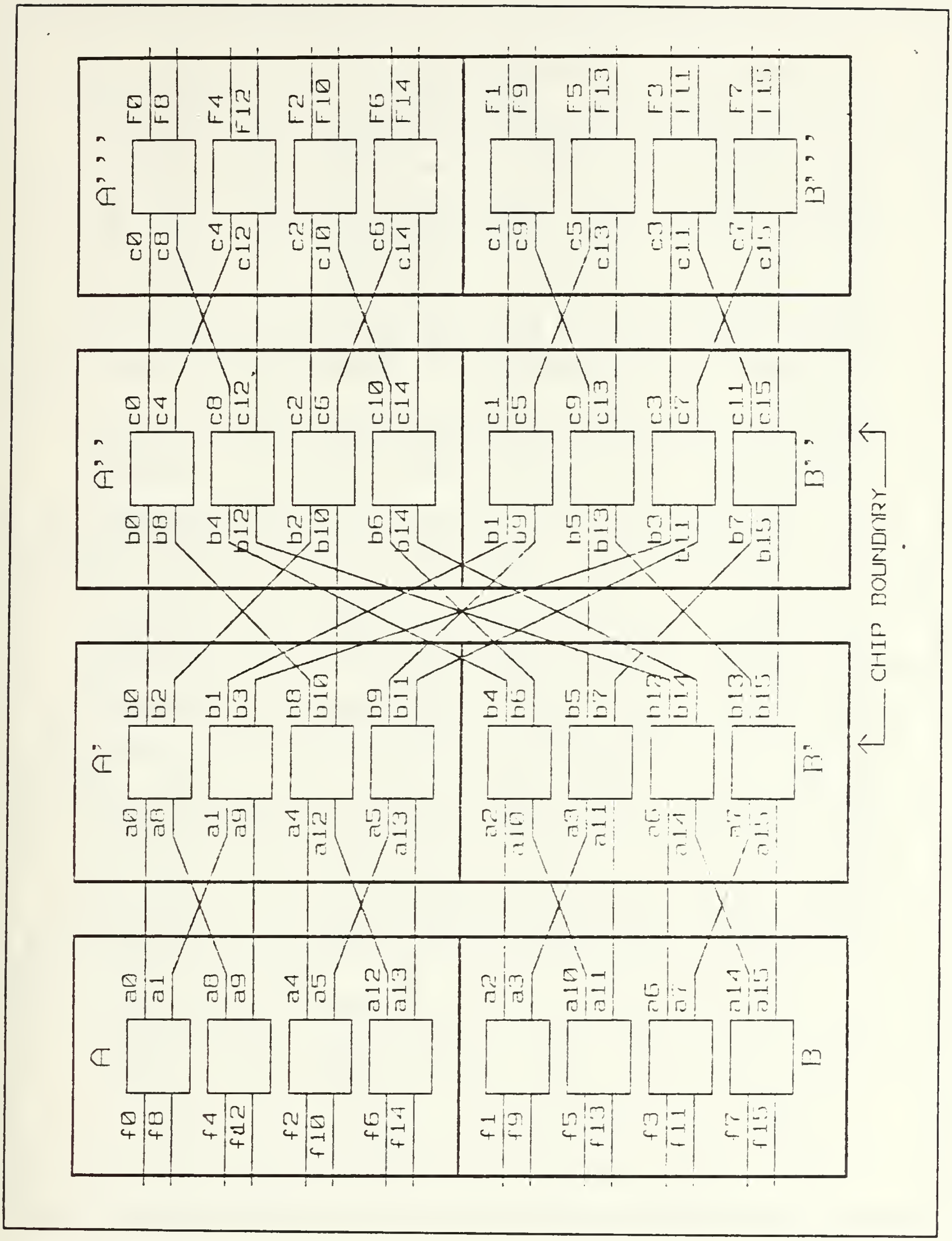
A
B
A'
B'
A''
B''
A'''
B'''
CHIP BOUNDARY
f0 f8
f4 f12
f2 f10
f6 f14
f1 f9
f5 f13
f3 f11
f7 f15
a0 a1
a8 a9
a4 a5
a12 a13
a2 a3
a10 a11
a6 a7
a14 a15
a0 a8
a1 a9
a4 a12
a5 a13
a2 a10
a3 a11
a6 a14
a7 a15
b0 b2
b1 b3
b8 b10
b9 b11
b4 b6
b5 b7
b12 b14
b13 b15
b0 b8
b4 b12
b2 b10
b6 b14
b1 b9
b5 b13
b3 b11
b7 b15
c0 c4
c8 c12
c2 c6
c10 c14
c1 c5
c9 c13
c3 c7
c11 c15
c0 c8
c4 c12
c2 c10
c6 c14
c1 c9
c5 c13
c3 c11
c7 c15
F0 F8
F4 F12
F2 F10
F6 F14
F1 F9
F5 F13
F3 F11
F7 F15

Figure 4.7 Sixteen Point Fast Fourier Transform
Modified 4 × 1 Reuse Architecture.

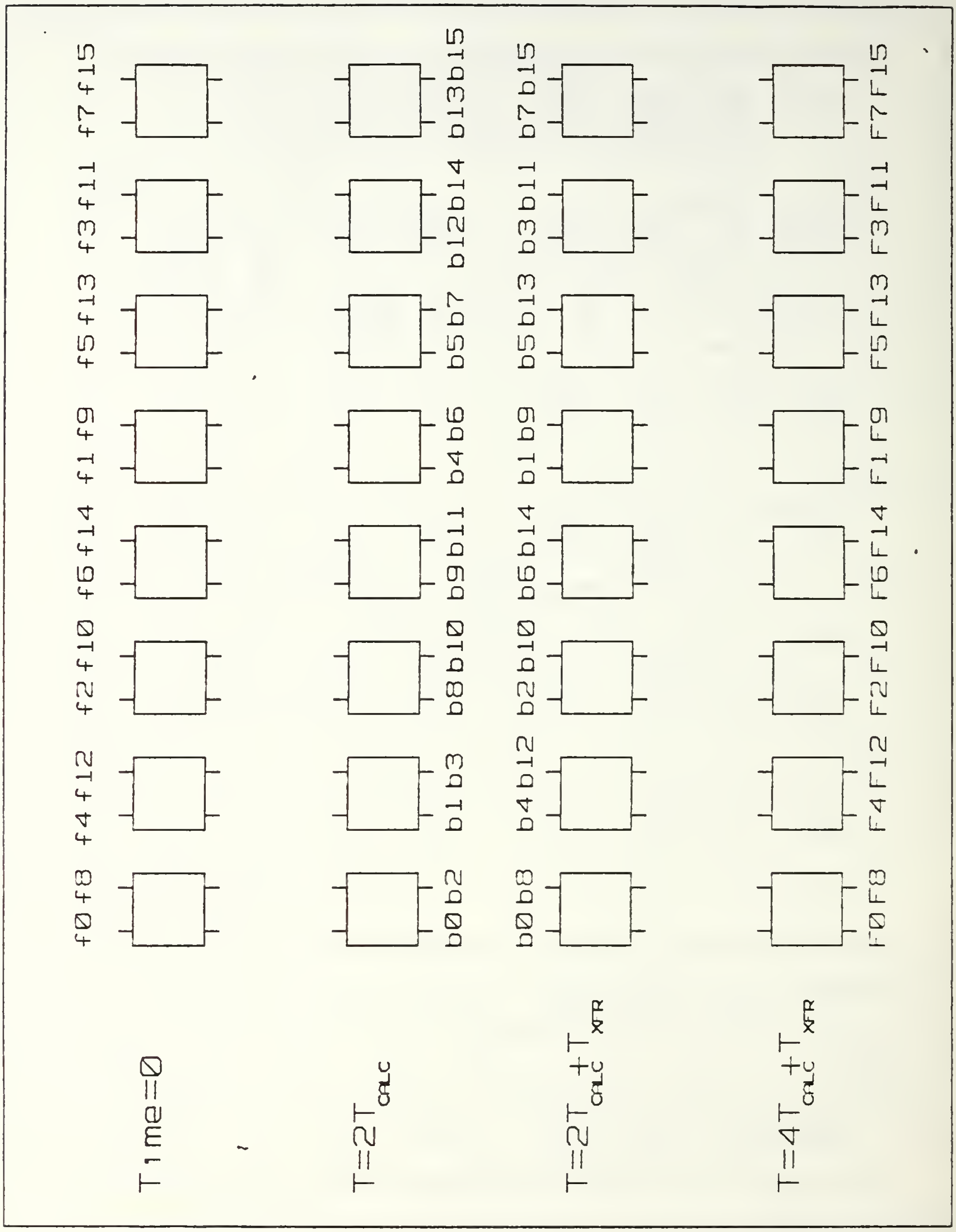
Time=0
f0 f8
f4 f12
f2 f10
f6 f14
f1 f9
f5 f13
f3 f11
f7 f15
T=2T_CALC
b0 b2
b1 b3
b8 b10
b9 b11
b4 b6
b5 b7
b12 b14
b13 b15
T=2T_CALC+T_XFR
b0 b8
b4 b12
b2 b10
b6 b14
b1 b9
b5 b13
b3 b11
b7 b15
T=4T_CALC+T_XFR
F0 F8
F4 F12
F2 F10
F6 F14
F1 F9
F5 F13
F3 F11
F7 F15

Figure 4.8 Sixteen Point Fast Fourier Transform
Modified 4 × 1 Reuse Architecture
Non-Interleaved.

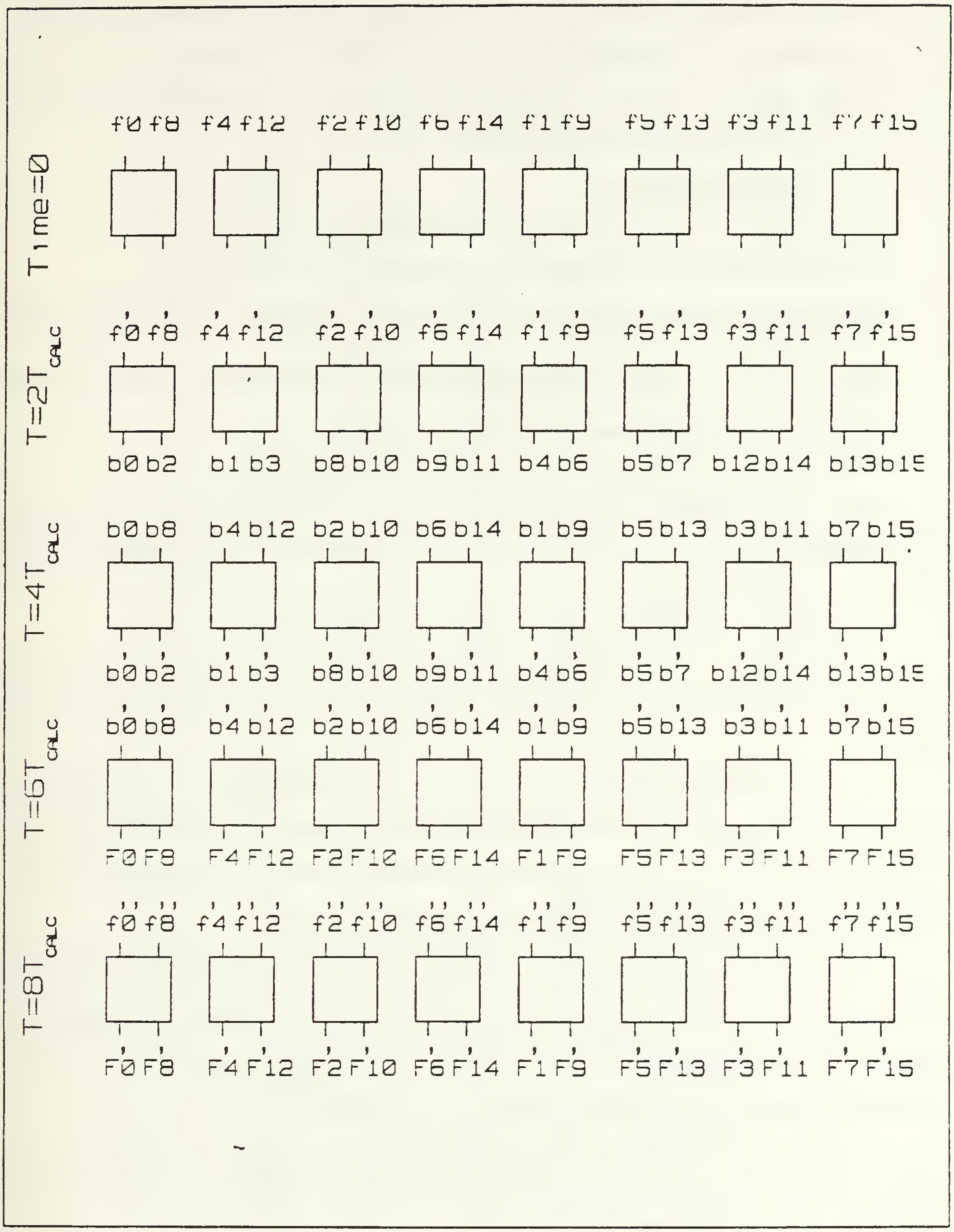
Time=0
f0 f8 f4 f12 f2 f10 f6 f14 f1 f9 f5 f13 f3 f11 f7 f15
$T=2T_{CALC}$
f0' f8' f4' f12' f2' f10' f6' f14' f1' f9' f5' f13' f3' f11' f7' f15'
b0 b2 b1 b3 b8 b10 b9 b11 b4 b6 b5 b7 b12 b14 b13 b15
$T=4T_{CALC}$
b0 b8 b4 b12 b2 b10 b6 b14 b1 b9 b5 b13 b3 b11 b7 b15
b0' b2' b1' b3' b8' b10' b9' b11' b4' b6' b5' b7' b12' b14' b13' b15'
$T=6T_{CALC}$
b0' b8' b4' b12' b2' b10' b6' b14' b1' b9' b5' b13' b3' b11' b7' b15'
F0 F8 F4 F12 F2 F10 F6 F14 F1 F9 F5 F13 F3 F11 F7 F15
$T=8T_{CALC}$
f0'' f8'' f4'' f12'' f2'' f10'' f6'' f14'' f1'' f9'' f5'' f13'' f3'' f11'' f7'' f15''
F0' F8' F4' F12' F2' F10' F6' F14' F1' F9' F5' F13' F3' F11' F7' F15'

Figure 4.9 Sixteen Point Fast Fourier Transform
Modified 4 × 1 Reuse Architecture
Interleaved.

is 2 FFTs per $8T_{\mathrm{CALC}}$, or 1 per $4T_{\mathrm{CALC}}$--slightly higher than in the non-interleaved case. This improvement in throughput was achieved without an increase in bus speed; alternatively one could reduce bus speed requirements without lowering throughput by incorporating an interleaved reuse architecture.

B. DATA DISTRIBUTION

Data delivery to the processors can be accomplished several ways:

- processing elements all receive the same data in broadcast fashion
- all processors "know" when it's their turn to receive data and they query the RBIU for it
- data words are "tagged" with their destination--RBIU reads the tag and delivers data words to their intended processor
- processors contend for bus access with each other
- RBIU delivers data to processors in a preset schedule.

Only this last scheme (using a preset schedule) promises to have sufficient speed to be acceptable for use with the OM. But is it possible to use an *a priori* schedule, and what would it look like?

1. Pipeline Architecture

Returning to the example of a FFT computer built of N-processor ICs, Figure 4.4 shows the data exchanges required by a sixteen point FFT if a pipeline architecture is used.

The sequence of data on the bus is essentially arbitrary. In choosing the sequence, it is reasonable to avoid sequences which deliver several data words to the same BIU one right after the other, in order to minimize the speed required of the BIU. Figure 4.10 shows one suitable choice.

Due to the regular structure of the FFT, there is a simple algorithm to calculate the address of any data word's destination, based only on its position in the data stream, as shown in Table VIII. Because of this, the RBIU's data distribution logic can be implemented with little more than a binary counter. The transmission algorithm is equally uncomplicated, as shown in Table IX.

The fact that inter-stage data exchange patterns in the FFT computation are regular and easily implemented in hardware lends further support to the use of preset schedules to control BIU data distribution.

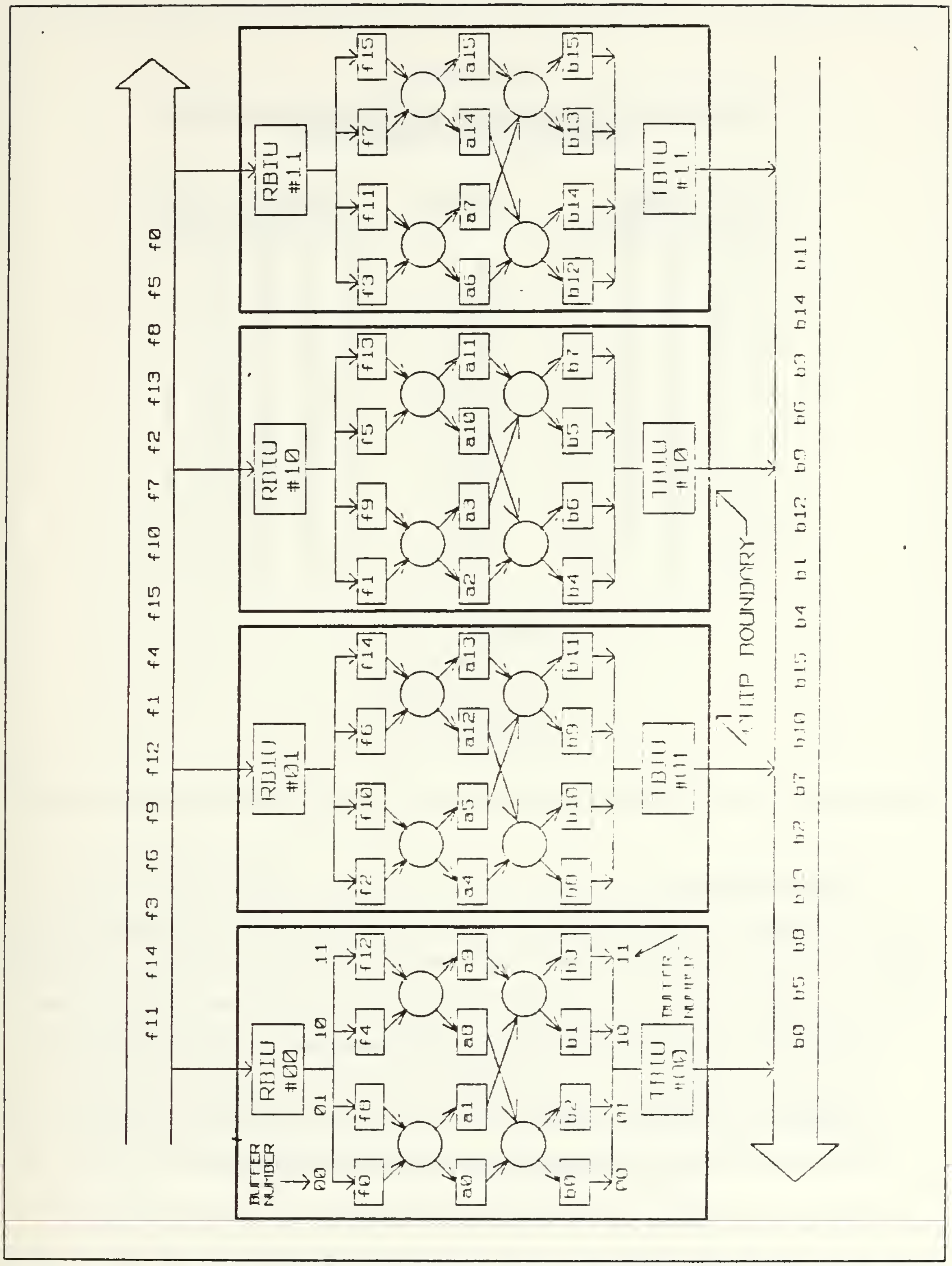
f11 f14 f3 f6 f9 f12 f1 f4 f15 f10 f7 f2 f13 f8 f5 f0
RBIU #00
RBIU #01
RBIU #10
RBIU #11
BUFFER NUMBER
00
01
10
11
f0 f8 f4 f12
a0 a1 a8 a9
b0 b2 b1 b3
f2 f10 f6 f14
a4 a5 a12 a13
b8 b10 b9 b11
f1 f9 f5 f13
a2 a3 a10 a11
b4 b6 b5 b7
f3 f11 f7 f15
a6 a7 a14 a15
b12 b14 b13 b15
TBIU #00
TBIU #01
TBIU #10
TBIU #11
CHIP BOUNDARY
b0 b5 b8 b13 b2 b7 b10 b15 b4 b1 b12 b9 b6 b3 b14 b11

Figure 4.10 Sixteen Point Fast Fourier Transform Distributed Among Four Multi-Processor Chips.

TABLE VIII

PRESET SCHEDULE FOR DATA DISTRIBUTION PIPELINE ARCHITECTURE

Word Sequence				Data Word	Destination Address			
					RBIU		Buffer	
W_3	W_2	W_1	W_0		D_3	D_2	D_1	D_0
0	0	0	0	f0	0	0	0	0
0	0	0	1	f5	1	0	1	0
0	0	1	0	f8	0	0	0	1
0	0	1	1	f13	1	0	1	1
0	1	0	0	f2	0	1	0	0
0	1	0	1	f7	1	1	1	0
0	1	1	0	f10	0	1	0	1
0	1	1	1	f15	1	1	1	1
1	0	0	0	f4	0	0	1	0
1	0	0	1	f1	1	0	0	0
1	0	1	0	f12	0	0	1	1
1	0	1	1	f9	1	0	0	1
1	1	0	0	f6	0	1	1	0
1	1	0	1	f3	1	1	0	0
1	1	1	0	f14	0	1	1	1
1	1	1	1	f11	1	1	0	1

$$D_3 = W_0$$

$$D_2 = W_2$$

$$D_1 = W_0 \oplus W_3$$

$$D_0 = W_1$$

2. Reuse Architecture

Tables X and XI and Figure 4.11 show the data flow structure required to compute a 16-point FFT with a reuse architecture. Although the task is accomplished with fewer processors than in Figure 4.10, there are three additional complications:

- additional buffers directly connect processors which must exchange data in intermediate stages of the calculation
- an internal path exists between TBIU and RBIU to allow processors which are not directly connected to exchange data
- BIUs must coordinate the use of internal and external paths.

TABLE IX

PRESET SCHEDULE FOR DATA DISTRIBUTION PIPELINE ARCHITECTURE

Word Sequence				Data Word	Source Address			
					TBIU		Buffer	
W_3	W_2	W_1	W_0		S_3	S_2	S_1	S_0
0	0	0	0	b0	0	0	0	0
0	0	0	1	b5	1	0	0	1
0	0	1	0	b8	0	1	0	0
0	0	1	1	b13	1	1	0	1
0	1	0	0	b2	0	0	1	0
0	1	0	1	b7	1	0	1	1
0	1	1	0	b10	0	1	1	0
0	1	1	1	b15	1	1	1	1
1	0	0	0	b4	1	0	0	0
1	0	0	1	b1	0	0	0	1
1	0	1	0	b12	1	1	0	0
1	0	1	1	b9	0	1	0	1
1	1	0	0	b6	1	0	1	0
1	1	0	1	b3	0	0	1	1
1	1	1	0	b14	1	1	1	0
1	1	1	1	b11	0	1	1	1

$$S_3 = W_0 \oplus W_3$$

$$S_2 = W_1$$

$$S_1 = W_2$$

$$S_0 = W_0$$

C. RECEIVER TASKS

We can view the data distribution circuitry as being separated into a Receiving Bus Interface Unit (RBIU) and a Transmitting Bus Interface Unit (TBIU). The RBIU must:

- capture data from the high-speed bus
- convert data from serial to parallel format
- perform error detection/correction
- deliver the data word to its destination processor.

Figure 4.12 shows the architecture developed in this project to accomplish these tasks. It may be noted that this architecture uses a separately distributed clock signal. This scheme was used to simplfy the construction and testing of a system prototype,

TABLE X

PRESET SCHEDULE FOR DATA DISTRIBUTION REUSE ARCHITECTURE

Word Sequence				Data Word	Destination Address			
					RBIU		Buffer	
W_3	W_2	W_1	W_0		D_3	D_2	D_1	D_0
0	0	0	0	f0	0	0	0	0
0	0	0	1	f1	1	0	0	0
0	0	1	0	f8	0	0	0	1
0	0	1	1	f9	1	0	0	1
0	1	0	0	f4	0	0	1	0
0	1	0	1	f5	1	0	1	0
0	1	1	0	f12	0	0	1	1
0	1	1	1	f13	1	0	1	1
1	0	0	0	f2	0	1	0	0
1	0	0	1	f3	1	1	0	0
1	0	1	0	f10	0	1	0	1
1	0	1	1	f11	1	1	0	1
1	1	0	0	f6	0	1	1	0
1	1	0	1	f7	1	1	1	0
1	1	1	0	f14	0	1	1	1
1	1	1	1	f15	1	1	1	1

$$D_3 = W_0$$

$$D_2 = W_3$$

$$D_1 = W_2$$

$$D_0 = W_1$$

but once past this phase the clock could be embedded in the data stream itself (as in Manchester coding), eliminating the need for a separate clock line. Alternatively, if a fiber optic data link were used, the clock could be sent on the same fiber as the data, but at a different carrier frequency (color), allowing clock recovery independently of data reception.

The control signals shown in Figure 4.12 also deserve some discussion. The RBIU circuitry develops these signals as a function of the bit count, then distributes the signals depending on which word is currently being received. These signals control the First-In-First-Out (FIFO) stacks which buffer data between the RBIU and the

TABLE XI

PRESET SCHEDULE FOR DATA DISTRIBUTION REUSE ARCHITECTURE

Word Sequence				Data Word	Source Address			
					TBIU		Buffer	
W_3	W_2	W_1	W_0		S_3	S_2	S_1	S_0
0	0	0	0	b0	0	0	0	0
0	0	0	1	b1	0	0	1	0
0	0	1	0	b8	0	1	0	0
0	0	1	1	b9	0	1	1	0
0	1	0	0	b4	1	0	0	0
0	1	0	1	b5	1	0	1	0
0	1	1	0	b12	1	1	0	0
0	1	1	1	b13	1	1	1	0
1	0	0	0	b2	0	0	0	1
1	0	0	1	b3	0	0	1	1
1	0	1	0	b10	0	1	0	1
1	0	1	1	b11	0	1	1	1
1	1	0	0	b6	1	0	0	1
1	1	0	1	b7	1	0	1	1
1	1	1	0	b14	1	1	0	1
1	1	1	1	b15	1	1	1	1

$$S_3 = W_2$$

$$S_2 = W_1$$

$$S_1 = W_0$$

$$S_0 = W_3$$

Processing Elements (P/E), as well as between the P/Es and the TBIU. These FIFOs require signals to cause them to:

- load a new data word (from the RBIU)
- output the next word (to the TBIU)
- advance the stack to bring up the next output word (now that the TBIU has the current word)

D. TRANSMITTER TASKS

The transmission part of the data distribution circuitry must:

- take the data word from its source processor.
- convert data from parallel to serial format
- add error detection bits
- insert data onto the high-speed bus

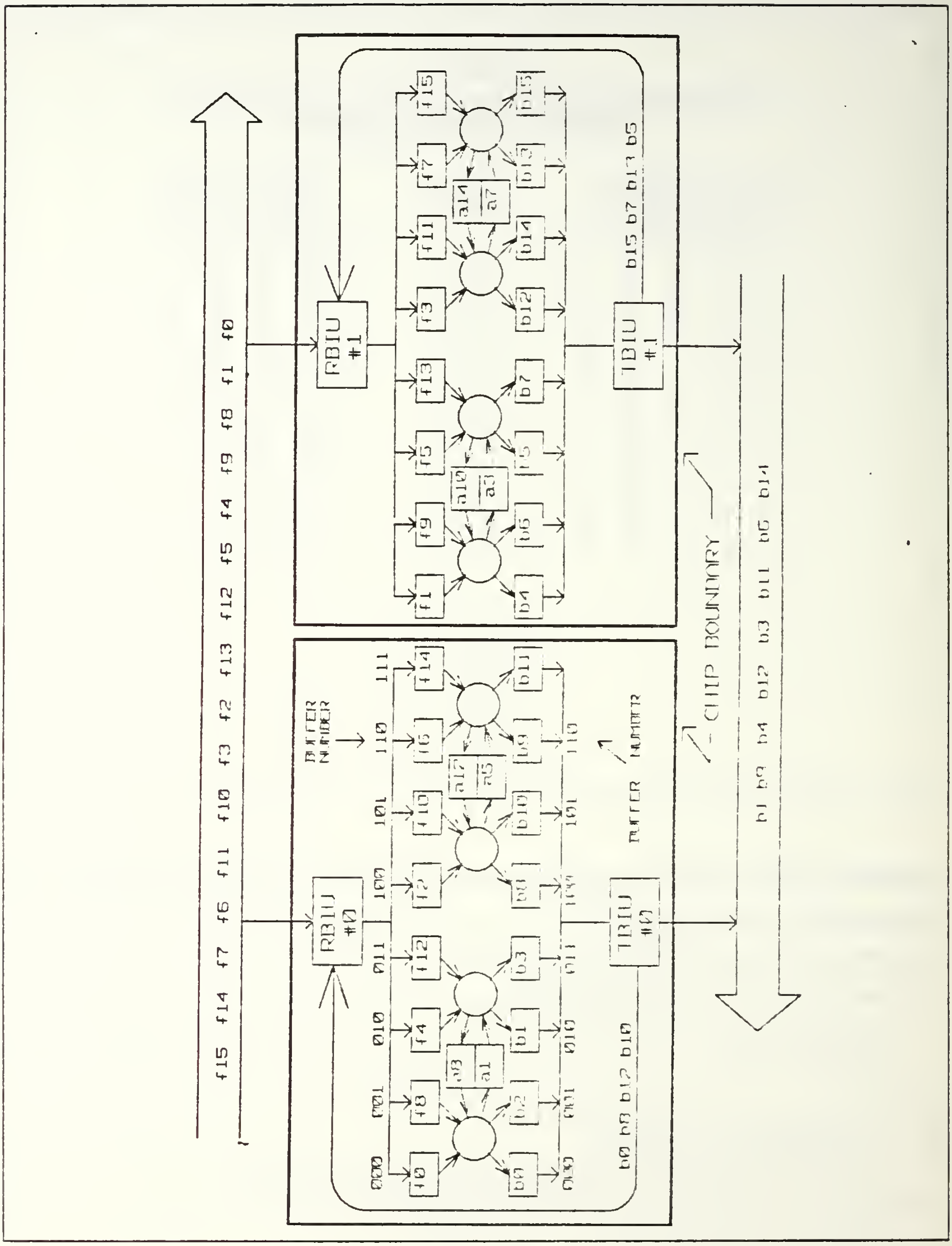
f15 f14 f7 f6 f11 f10 f3 f2 f13 f12 f5 f4 f9 f8 f1 f0
RBIU #0
RBIU #1
TBIU #0
TBIU #1
BUFFER NUMBER
BUFFER NUMBER
CHIP BOUNDARY
000
001
010
011
100
101
110
111
f0
f8
f4
f12
f2
f10
f6
f14
f1
f9
f5
f13
f3
f11
f7
f15
a0
a1
a4
a5
a12
a10
a3
a14
a7
b0
b2
b1
b3
b8
b10
b9
b11
b4
b6
b5
b7
b12
b14
b13
b15
b0 b8 b12 b10
b15 b7 b13 b5
b1 b9 b4 b12 b3 b11 b6 b14

Figure 4.11 Sixteen Point Fast Fourier Transform Distributed Between Two Chips Using a Reuse Architecture.

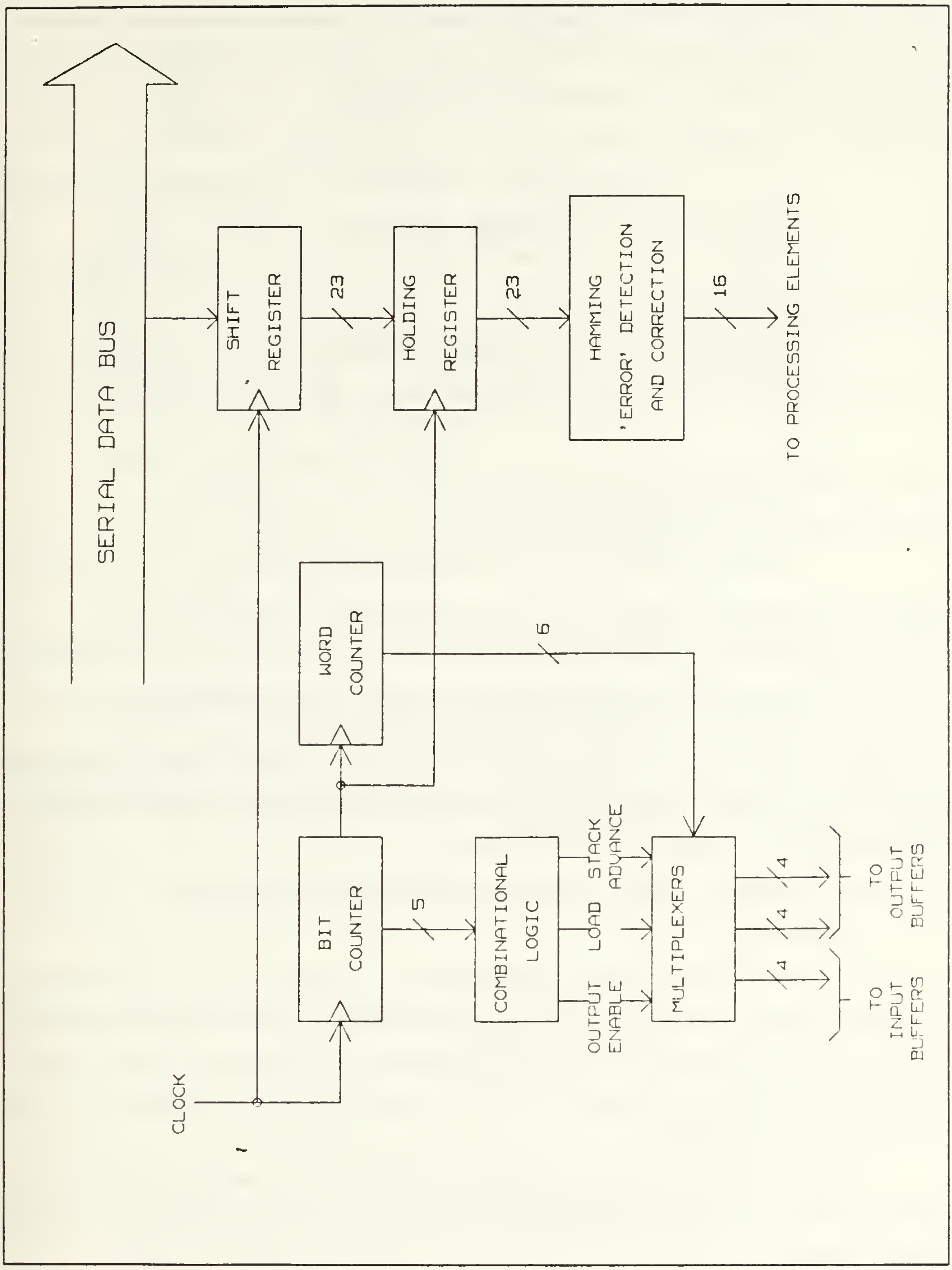
SERIAL DATA BUS
SHIFT REGISTER
23
HOLDING REGISTER
23
HAMMING 'ERROR' DETECTION AND CORRECTION
16
TO PROCESSING ELEMENTS
WORD COUNTER
6
BIT COUNTER
5
COMBINATIONAL LOGIC
STACK ADVANCE
LOAD
OUTPUT ENABLE
MULTIPLEXERS
4
4
4
TO OUTPUT BUFFERS
TO INPUT BUFFERS
CLOCK

Figure 4.12 Receiving Bus Interface Unit Architecture.

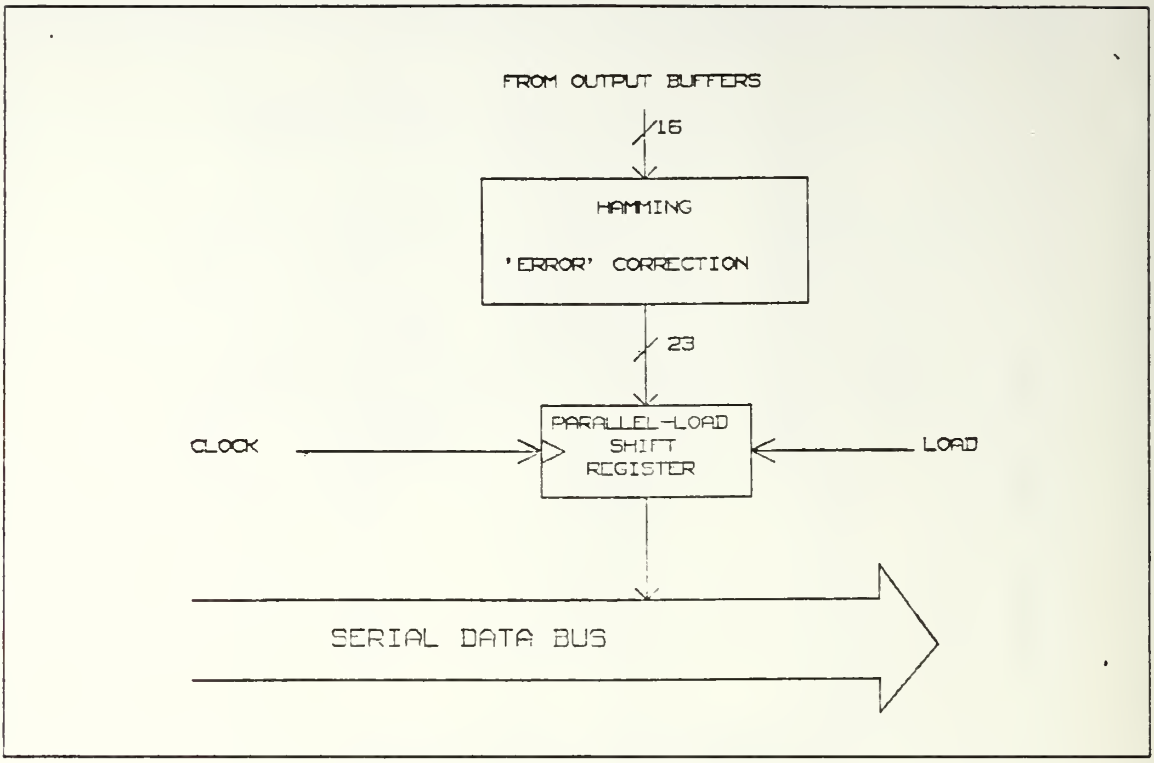

Figure 4.13 Transmitting Bus Interface Unit Architecture.

Figure 4.13 shows the architecture developed in this project to accomplish these tasks. The control and timing circuitry needed to interface the output FIFOs with the TBIU is included as part of the RBIU diagram.

E. CONCLUSIONS AND LIMITATIONS OF THIS RESEARCH

1. Conclusions

The high speed serial communication provided by the Optoelectonic Multiplexer makes possible a shared-bus parallel processing architecture for problems like the FFT where the data distribution schedule can be determined *a priori*. The data distribution algorithms for the FFT are quite simple and can be realized with little more than a binary counter.

For the FFT, processors groupings on chip should correspond to the $2^{n-1} \times n$ matrices inherent in the FFT calculation in order to minimize the amount of inter-chip communications.

Trends in actual processor data suggest that the throughput of the processor in most cases is proportional to the [size of the processor]$^{\lambda}$, where $\lambda < 1$ and "size" refers to both transistor count and power dissipation. This implies that, for a given chip size, dividing the chip into increasing numbers of smaller processors raises the number of processors faster than it lowers the throughput of an individual processor. Thus, for most types of processors studied, the greatest throughput is achieved by organizing a large chip as a bank of many simple processors.

Finally, a single-chip OM-based parallel processor is feasible since:

- Manufacturers can fabricate sufficient transsitors on a single chip to construct many simple processors.
- A chip composed of only about 12 simple processors, easily achieved with current fabrication technology, could produce enough throughput in a highly structured problem (like the FFT) to justify the use of the OM's high capacity.
- Constructing such a chip in a conventional package using one pin or lead per bit would require an excessive package size.

2. Limitations and Recommendations

The architecture described in this report was designed with only the FFT in mind. It may not be adaptable to less structured calculations or to systems which must perform a wide variety of calculations.

Multiple-processor chip performance was predicted based on a limited sampling of current processor data. Further research, using a comprehensive study of actual processor performance, is needed to augment the simple model developed here.

The comparison of conventional leaded packages and serially multiplexed packages considered only the extremes of one pin per bit and one pin per chip. Additional study of alternatives between these endpoints is needed to determine at what point the cost (in terms of dollars, chip area, and heat) of the Optoelectronic Multiplexer is justified by its higher performance.

LIST OF REFERENCES

1. Hopper, G., Speech to Naval Postgraduate School faculty and students, 10 July 1985.

2. Bernhard, R., "Computing at the Speed Limit," *IEEE Spectrum*, v. 19, p. 26, July 1982.

3. Rigas, H., "Parallel Processing of Class of Scientific Problem," *Proceeding of the Tenth Annual Pittsburg Conference on Modeling and Simulation*, p. 2017, Instrument Society of America, 1979.

4. Anderson, G. and Jensen, E., "Computer Interconnection Structures: Taxonomy, Characteristics and Examples," *Computing Surveys*, v. 7, pp. 197-213, December 1975.

5. Reedy, R. and Albares, D., *Optoelectronic Integrated Circuit*, U.S. Patent Application number 746704, 20 June 1985.

6. Filip, A., "A Distributed Signal Processing Architecture," *Proceedings of the Third International Conference on Distributed Computing Systems*, pp. 49-54, IEEE Computer Society Press, 1982.

7. Siegel, H., "The Theory Underlying the Partioning of Permutation Networks," *IEEE Transactions on Computers*, v. C-29, pp. 791-800, September 1980.

8. Horowitz, E. and Zorat, A., "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI," *IEEE Transactions on Computers*, v. C-30, pp. 247-253, April 1981.

9. Franklin, M., "VLSI Performance Comparisons of Banyan and Crossbar Communications Networks," *IEEE Transactions on Computers*, v. C-30, pp. 283-290, April 1981.

10. Lenfant, J., "Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations," *IEEE Transactions on Computers*, v. C-27, pp. 637-647, July 1978.

11. Rigas, H. and Abbott, L., *Final Report on Optoelectronic Multiplexer Project*, Report to Naval Ocean Systems Center, San Diego, California, p. 4, August 1986.

12. Muroga, S., *VLSI System Design*, p. 418, Wiley Press, 1982.

13. Motorola Corporation, MC68020 Product Data Card, August 1985.

14. Kaminker, A., "A 32-Bit Microprocessor with Virtual Memory Support," *IEEE Journal of Solid-State Circuits*, v. SC-16, p. 556, October 1981.

15. Kadota, H., and others, "A New Register File Structure for the High-Speed Microprocessor," *IEEE Journal of Solid-State Circuits*, v. SC-17, pp. 892-897, October 1982.

16. Beyers, J.W., and others, "A 32-Bit VLSI CPU Chip," *IEEE Journal of Solid-State Circuits*, v. SC-16, p. 537, October 1981.

17. Pomper, M., and others, "A 32-Bit Execution Unit in an Advanced NMOS Technology," *IEEE Journal of Solid-State Circuits*, v. SC-17, p. 533, June 1982.

18. Rowen, C., and others, "A Pipelined 32-b NMOS Microprocessor," *International Solid-State Circuits Conference*, p. 180, IEEE, 1984.

19. Simcoe, B., and others, "A Floating Point Unit for a 32b Microprocessor System," *Custom Integrated Circuits Conference*, p. 478, IEEE, 1984.

20. Ware, F. A., and others, "64-Bit Monolithic Floating Point Processors," *IEEE Journal of Solid-State Circuits*, v. SC-17, pp. 898-907, October 1982.

21. Bursky, D., "Triple-Mode ALU Lets DSP Chip Zip Through Multiplications," *Electronic Design*, v. 33, pp. 79-80, 31 October 1985.

22. Takeda, K., "A Single Chip 80b Floating Point Processor," *International Solid-State Circuits Conference*, pp. 16-17, IEEE, 1985.

23. Anderson, J. M., Troutman, B. L., and Allen, R. A., "A CMOS LSI 16 x 16 Multiplier/Multiplier-Accumulator," *International Solid-State Circuits Conference*, pp. 124-125, IEEE, 1982.

24. Wittmer, N. C., and others, "A NMOS LSI 16 x 16 Multiplier," *International Solid-State Circuits Conference*, p. 32, IEEE, 1983.

25. Iwamura, J., and others, "A 16-bit CMOS/SOS Multiplier-Accumulator," *IEEE International Conference on Circuits and Computers*, pp. 151-154, IEEE Computer Society, 1982.

26. Kaji, Y., and others, "A 45 ns 16 x 16 CMOS Multiplier," *International Solid-State Circuits Conference*, pp. 84-85, IEEE, 1984.

27. Iwamura, J., and others, "A CMOS/SOS Multiplier," *International Solid-State Circuits Conference*, pp. 92-93, IEEE, 1984.

28. Uya, M., Kaneko, K., and Yasui, J., "A CMOS Floating Point Multiplier," *International Solid-State Circuits Conference*, pp. 90-91, IEEE, 1984.

29. Windsor, B. and Wilson, J., "Arithmetic Duo Excells in Computing Floating -Point Products," *Electronic Design*, v. 32, pp. 144-145, 17 May 1984.

30. Lee, F., Chiu, C. P., and Toth, F., "16-by-16-Bit Multipliers Fabricated in CMOS Rival the Speed of Bipolars," *Electronic Design*, v. 32, p. 311, 14 June 1984.

31. Tran, T., and others, "A 1.0 Micron CMOS 32-Bit IEEE Format Floating Point Chip Set for Digital Signal Processing," *Custom Integrated Circuits Conference*, pp. 280-282, IEEE, 1985.

32. Bursky, D., "CMOS Multipliers Operate in as Little as 45 ns," *Electronic Design*, v. 32, p. 250, 12 July 1984.

33. Tago, H., and others, "A 6k-Gate CMOS Gate Array," *IEEE Journal of Solid-State Circuits*, v. SC-17, pp. 907-912, October 1982.

34. Rein, H., and others, "A Time-Division Multiplexer IC for Bit Rates Up to About 2 Gbits/s," *IEEE Journal of Solid-State Circuits*, v. SC-17, pp. 306-309, June 1982.

35. Hughes, J., and others, "A Versatile ECL Multiplexer IC for the Gbit Range," *IEEE Journal of Solid-State Circuits*, v. SC-14, pp. 812-817, October 1979.

36. Langmann, U., "Injection Laser Modulation at 2 Gbit/s by Monolithic Silicon Multiplexer," *IEEE Transactions on Microwave Theory and Techniques*, v. MTT-32, pp. 1675-1677, December 1984.

37. Nakayama, Y., and others, "A GaAs Data Switching IC for a Gigabits per Second Communication System," *IEEE Journal of Solid-State Circuits*, v. SC-21, pp. 157-160, February 1986.

38. Swartzlander, E., "VLSI Architecture," *VLSI Fundamentals and Applications*, p. 183, Springer-Verlag, 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Commander (Code 553) Naval Ocean Systems Center Attn: Mr. Don Albares San Diego, California 92152-5000	1
4.	Commander (Code 553) Naval Ocean Systems Center Attn: Mr. Ron Reedy San Diego, California 92152-5000	1
5.	Superintendent (Code 62) Naval Postgraduate School Attn: Prof. H.B. Rigas Monterey, California 93943-5000	5
6.	Superintendent (Code 62) Naval Postgraduate School Attn: Prof. L.W. Abbott Monterey, California 93943-5000	5
7.	Superintendent (Code 62) Naval Postgraduate School Attn: ECE Department Office Monterey, California 93943-5000	2

100
17663/7

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-50

220158

Thesis
D2635
c.1

Delaney
A serial bus architecture for parallel processing systems.

220158

Thesis
D2635
c.1

Delaney
A serial bus architecture for parallel processing systems.

thesD2635
A serial bus architecture for parallel p
3 2768 000 68144 9
DUDLEY KNOX LIBRARY