Abstract:
The subject of this thesis is the design of a cellular computer specially organized to efficiently implement matrix algorithms, and capable of fault-tolerant operation.

The contents of the thesis are summarized as follows: First, a review of pertinent literature in the areas of cellular logic design, array-structured computers, fault-tolerant computing systems, and large scale integration of digital circuits is presented. Second, an assembly language instruction set and an array-structured, computer simulation program (PAL-1) are described including as an example a Kalman filter algorithm implementation. Third, the computer system organization and logical design of an array-structured computer are described including a discussion of the LSI implementation of a processing cell. Fourth, an adaptation of the array-structured computer design to fault-tolerant operation is presented. Three fault-tolerant computer organizations are described which use a special instruction set allowing fault detection, fault correction, and program rollback operations. Fifth, recommendations are made for further research in the area of array-structured computers capable of fault-tolerant operation.

THE DESIGN OF A MATRIX-ORIENTED CELLULAR COMPUTER CAPABLE OF FAULT-TOLERANT OPERATION

by

MICHAEL CARL MULDER

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

[illegible]
Head, Major Department

Donald A. Rudberg
Chairman, Examining Committee

K Goering
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

August, 1970

ACKNOWLEDGMENT

The author wishes to extend his gratitude for the guidance and encouragement rendered to him by Dr. Donald A. Rudberg during the course of this graduate work and research effort. Concurrently, special appreciation is given to Dr. Robert C. Minnick for his timely suggestions and efforts as a consultant for this research endeavor.

The financial support provided by the Department of Electrical Engineering at Montana State University and the Office of Naval Research under Contract No. N00014-67-C-0477 is gratefully acknowledged.

To a family of cheerful fortitude and endurance--my wife Carol, my daughter Jennifer Leigh, and my son Jonathan Michael, I dedicate this thesis.

TABLE OF CONTENTS

		Page
Chapter 1	INTRODUCTION AND REVIEW OF PERTINENT LITERATURE	1
1.1	Introduction	2
1.2	Survey of Microcellular Work	4
1.3	Survey of Array-Organized Computers	8
	1.3.1 The Unger Spatially-Oriented Computer	8
	1.3.2 The Holland Computer	10
	1.3.3 The Comfort Computer	12
	1.3.4 The SOLOMON Computer	13
	1.3.5 The Gonzalez Iterative Circuit Computer	15
	1.3.6 The ILLIAC IV Computer	16
	1.3.7 The MCB Computer	19
	1.3.8 The Berkeley Array Processor	20
1.4	Survey of the Cannon Cellular Computer	22
1.5	Current Trends in Fault-Tolerant Computers	26
1.6	Current Trends in Large Scale Integrated Circuits	28
1.7	Description of Research Effort	30
1.8	Organization of Remaining Chapters	32

Page

Chapter 2 DESIGN OF A PARALLEL ASSEMBLY LANGUAGE PROGRAM TO SIMULATE AN ARRAY-STRUCTURED COMPUTER 34

2.1 Introduction 35

2.2 Assembly Language Instruction Set 36

2.3 Simulation Program Structure 42

2.4 Memory Storage and Retrieval Subroutines . . . 53

2.5 Arithmetic Subroutines 53

2.5.1 Addition-Subtraction Subroutine . . . 57

2.5.2 Multiplication Subroutine 57

2.5.3 Division Subroutine 71

2.6 Normalization Subroutine 78

2.7 Timekeeping Subroutine 81

2.8 Output Data Structure 82

2.8.1 Assembly Language Instructions 82

2.8.2 Special Instructions 85

2.9 Implementation of the Kalman Filter Algorithm in PAL-1 86

2.10 Results Derived from the Simulation Program . . 90

Chapter 3 DESIGN OF AN ARRAY-STRUCTURED COMPUTER 95

3.1 Introduction 96

3.2 Computer System Functional Organization . . . 97

3.2.1 Introduction 97

Page

3.2.2 Global Control Unit Functions 97

3.2.3 Exterior Unit Functions 99

3.2.4 X and Y Command Block Functions 103

3.2.5 Processing Array Functions and Organizations 103

3.3 Data Representation and Logic Convention 106

3.3.1 Data Representation 106

3.3.2 Data Normalization 108

3.3.3 Logic Convention 109

3.4 Global Control Unit 109

3.5 Exterior Processing Units 115

3.5.1 Input-Output Unit 115

3.5.2 Data Transfer Microprocessing Unit 117

3.5.3 Dynamic Microprocessing Unit 122

3.6 X and Y Command Blocks 126

3.7 Mathematical Operation Block 131

3.7.1 Introduction 131

3.7.2 MOB Command Generation Module 137

3.7.2.1 Master Control and Sequential Router Section 139

3.7.2.2 Microinstruction Generation Sections .142

3.7.2.2.1 Data Source Selector and Router Generation Section . .142

Page

3.7.2.2.2 Parallel Adder-Subtractor Generation Section 144

3.7.2.2.3 Parallel Carry-Save Multiplier Generation Section 147

3.7.2.2.4 Normalizing and Data Link Generation Section . . 149

3.7.3 Memory Module 152

3.7.4 Data Source Selector and Router Module . . 155

3.7.5 Parallel Adder-Subtractor Module 157

3.7.6 Parallel Carry-Save Multiplier Module . . 161

3.7.7 Normalizing and Data Link Module 166

3.8 Data Transfer Block 170

3.8.1 Introduction 170

3.8.2 Data Transfer Block Organization 172

3.8.3 Data Transfer Operation Bussing Structures 177

3.9 Adaptation of the Array-Structured Computer to Fault-Tolerant Operation 184

3.9.1 Introduction 184

3.9.2 Fault-Tolerant Operation Hardware . . . 185

3.9.2.1 Introduction 185

3.9.2.2 Fault Detection Hardware 186

3.9.2.3 Fault Correction Hardware 188

Page

3.9.3 Fault-Tolerant Operation Instruction Set 191

3.9.4 Fault-Tolerant Computer Structures . . 192

3.9.4.1 Introduction 192

3.9.4.2 Software-Oriented Structure . . . 193

3.9.4.3 Triple-Redundant Structure . . . 195

3.9.4.4 Multiple-Processing Array Structure 197

3.10 Implementation of a Single Processing Array Cell 200

3.10.1 Introduction 200

3.10.2 Implementation of a Single Processing Cell 201

Chapter 4 SUMMARY AND RECOMMENDATIONS 208

4.1 Introduction 209

4.2 Summary 209

4.2.1 Parallel Assembly Language Program . . 209

4.2.2 Array-Structured Cellular Computer . . . 211

4.2.3 Fault-Tolerant Computer Organizations . . 214

4.3 Recommendations for Future Research 215

Appendix A The Kalman Filter Algorithm 218

Appendix B Pertinent Matrix Algorithms 227

Appendix C Listing of the Parallel Assembly Language Simulation Program 239

Appendix D Listing of the Linearized Kalman Filter Algorithm in PAL-1 303

Bibliography 308

LIST OF FIGURES

Figure		Page
1.1	The Unger Spatially-Oriented Computer	8
1.2	The SOLOMON Computer	14
1.3	The ILLIAC IV Computer	17
1.4	The Cannon Cellular Computer	23
2.1	Memory Doubleword Format	43
2.2	Generalized Flow Chart for PAL-1	44
2.3	Instruction Set Format	45
2.4	Flow Chart for Instruction Decomposition	46
2.5	Flow Chart for Storage of Doubleword in MEMORY . .	54
2.6	Flow Chart for Generating Element Address	55
2.7	Flow Chart for Retrieving Doubleword from MEMORY	56
2.8	Flow Chart for Subroutine ALNADD	58
2.9	Flow Chart for Subroutine MULTIPLY	64
2.10	Flow Chart for Subroutine RECIP	72
2.11	Flow Chart for Subroutine NORMALIZE	79
2.12	Flow Chart for Subroutine NORMTIME	83
3.1	Array Structured Computer Organization	98
3.2	Global Control and Exterior Units	100
3.3	Data Buss Structure for a 4 x 4 Processing Array	105

Figure Page

3.4 Flow Chart for the Data Transfer Cycle 107

3.5 Floating Point Data Format 108

3.6 Modular Organization of the GCU 113

3.7 Data Transfer Microprocessing Unit 118

3.8 Data Transfer Macroinstruction Format 119

3.9 Dynamic Microprocessing Unit 123

3.10 Mathematical Operation Macroinstruction Format . . 124

3.11 Control Lines for a 4 x 4 Processing Array 127

3.12 X Command Block for a 4 x 4 Processing Array . . . 128

3.13 Switching Module for the X Command Block 129

3.14 Y Command Block for a 4 x 4 Processing Array . . . 132

3.15 Mathematical Operation Block 134

3.16 MOB Macroinstruction Format 135

3.17 MOB Command Generation Module 138

3.18 Master Control and Sequential Router Sections . . . 140

3.19 Microinstruction Generation Section for the Data Source Selector and Router Module 143

3.20 Microinstruction Generation Section for the Parallel Adder-Subtractor Module 145

3.21 Microinstruction Generation Section for the Parallel Carry-Save Multiplier Module 148

3.22 Microinstruction Generation Section for the Normalizing and Data Link Module 150

Figure		Page
3.23	Memory Storage Cell	153
3.24	Memory Module	154
3.25	Data Source Selector and Router Module	156
3.26	Parallel Floating Point Adder-Subtractor Module	158
3.27	Parallel Floating Point Carry-Save Multiplier Module	162
3.28	Binary Multiplication Process	164
3.29	Parallel Carry-Save Multiplier	165
3.30	Normalizing and Data Link Module	167
3.31	Normalizing Section	169
3.32	Data Transfer Block	173
3.33	32 Bit Parallel Shift Register	175
3.34	32 Bit Parallel Comparator	177
3.35	Data Bussing Structure for a Rotate Right Operation	178
3.36	Data Bussing Structure for a Rotate Up Operation	180
3.37	Data Bussing Structure for the Transpose Operation	182
3.38	Flow Chart for the Array Comparison Cycle	187
3.39	Restructuring of a 4 x 4 Processing Array	189
3.40	Data Transfer Macroinstruction Modification Process	190
3.41	Software Oriented, Fault-Tolerant Computer Organization	194

Figure		Page
3.42	Triple-Redundant, Fault-Tolerant Computer Organization	196
3.43	Multiple-Processing Array, Fault-Tolerant Computer Organization	199
A-1	Kalman Filter Problem Block Diagram	222

LIST OF TABLES

Table		Page
2.1	Parallel Assembly Language Instruction Set . . .	38
2.2	PAL-1 Output Format	84
2.3	Special Instruction Output Format	85
2.4	Matrix Allocation for the Kalman Filter Example .	86
2.5	State Vector Estimate Comparison	89
3.1	Selected Set of Cellular Logic Units	111
3.2	X Command Block Data Source Selection Code . . .	130
3.3	MOB Macroinstruction Bit Designation	136
3.4	Sequence of Microinstruction Generation	141
3.5	Microinstructions for the Parallel Adder-Subtractor Module	146
3.6	Microinstructions for the Normalizing and Data Link Module	149
3.7	Truth Table for a Memory Storage Cell	152
3.8	Truth Table for the Data Source Selector and Router	155
3.9	Normalizer and Data Source Selection Code . . .	168
3.10	DTB Microinstruction Bit Designation	174
3.11	DTB Microinstructions for a Rotate Right Operation .	179
3.12	DTB Microinstructions for a Rotate Up Operation .	179
3.13	DTB Microinstructions for the Transpose Operation	183

Table		Page
3.14	Special Instruction Set for the Fault-Tolerant Operation	191
3.15	Special Instruction Set Example	192
3.16	Gates Required for the Selected Cellular Logic Units	203
3.17	Gates Required in the MOB	205
3.18	Gates Required in the DTB	206

ABSTRACT

The subject of this thesis is the design of a cellular computer specially organized to efficiently implement matrix algorithms, and capable of fault-tolerant operation.

The contents of the thesis are summarized as follows: First, a review of pertinent literature in the areas of cellular logic design, array-structured computers, fault-tolerant computing systems, and large scale integration of digital circuits is presented. Second, an assembly language instruction set and an array-structured computer simulation program (PAL-1) are described including as an example a Kalman filter algorithm implementation. Third, the computer system organization and logical design of an array-structured computer are described including a discussion of the LSI implementation of a processing cell. Fourth, an adaptation of the array-structured computer design to fault-tolerant operation is presented. Three fault-tolerant computer organizations are described which use a special instruction set allowing fault detection, fault correction, and program rollback operations. Fifth, recommendations are made for further research in the area of array-structured computers capable of fault-tolerant operation.

CHAPTER 1

INTRODUCTION AND REVIEW OF PERTINENT LITERATURE

1.1 Introduction

The evolution of computer systems has resulted in significant changes in engineering methods and applications. This evolution began with the development of large general-purpose machines which were capable of many diverse operations. These machines tended to be physically large, expensive, relatively slow, and difficult to program. As computer design technology evolved, these machines became faster, more efficient, easier to program, but remained large and costly. It became apparent that definite applications existed for smaller computing units capable of fast operation. Hence, the special purpose computer was developed, which has a small physical size, low cost, fast execution time, and a small instruction repertoire. The disadvantage of this type of machine was the large initial cost and limited application of a given machine design. To implement another set of algorithms might require an entire system redesign. A classic example of such a limited application machine is the digital differential analyzer (DDA).

Neither of the above approaches appears to be suitable for combinations of on-line applications such as filtering, vehicle navigation, and signal processing. To effectively meet the computational requirements of these and similar algorithms, a third computer design philosophy is evolving which utilizes the inherent structure of a given class of operations to produce a computing system design.

Hence, with a computer so structured, classes of operations can be efficiently processed as in a special-purpose machine, and yet the general-purpose nature can be retained. The specially-organized computer resulting from this research effort is a product of this third computer design philosophy.

An important class of problems whose inherent structure can be used to form the design of a specially-organized computing system is that class which deals with matrix and vector operations. This type of problem suggests a machine structure which is array oriented, and capable of parallel operation. Such an organization would strongly suggest a processing array composed of identical processing cells with appropriate interconnection structure. Examples of this class of problems are Kalman filtering (1), vehicle navigation, phased-array radar control computations, and sonar receiving array data processing.

The design and hardware implementation of a specially-organized computing system will be strongly influenced by the recent advances in solid state technology. The advent of large scale integrated circuit technology has made possible the fabrication of complex digital processors on a single monolithic substrate. Hence, consideration must be given to the impact of this and related technologies on the design of computing systems. In the next section a survey of microcellular research as it pertains to the organization and physical embodiment of a specially-organized computer is presented.

1.2 Survey of Microcellular Work

As the trend in the field of digital circuit implementation shifted to the use of integrated circuit technology, it became apparent that new bodies of digital switching theory must be developed. One such new body of theory undertook design of complex digital circuits in the form of identical cells placed together and connected through a regular, repetitive interconnective structure to form what is called a cellular array. Each of these identical cells consisted of an integrated circuit capable of some specified logical operation. Since logical operations appear with various levels of complexity, it is clear that the corresponding cellular arrays can be divided into a number of categories according to their cell complexity. The category of cellular array to be surveyed in this section will be the low complexity set referred to as microcellular arrays. A microcellular array consists of rather simple cells, with each cell containing no more than a few gates.

In the study of cellular arrays, it is apparent that numerous methods exist for imbedding logical functions into microcellular arrays. Approaches found in reviewed literature are of two general types: fixed cell function units, and variable cell function units. The fixed cell function approach considers cell parameters used only in the modification of the cell interconnection structure, whereas the variable cell function approach considers cell parameters of both the function and interconnection structure.

One of the earliest efforts in the fixed cell function units was the eight neighbor array proposed in 1961 by Brooking (2) at the AFCRL to allow implementation of any arbitrary switching function. Each cell in this array is an eight input NOR gate; the NOR operation alone forming a complete set of logical primitives. Associated with each cell are eight parameters which provide an electrical disconnect for any subset of the eight nearest neighbor inputs. Several studies were made at AFCRL to implement variations of the eight neighbor arrays.

Combinational logic design methods for eight neighbor fixed cell function (NAND) cellular arrays were developed in 1963 by Spandorfer and Murphy (3) at the UNIVAC Engineering Center. Since the NAND operation forms a complete set, this type of cellular array will also allow implementation of any arbitrary switching function. This initial NAND cellular array consisted of imposing horizontal and vertical interconnection structures called wiggle busses on an eight neighbor array. This work evolved into a diamond-structured cellular array, resulting in more efficient array utilization. Approaches were also developed at UNIVAC (3) whereby faulty cells could be avoided by the introduction of spare rows and columns into the rectangular cellular arrays.

Another type of fixed cell function array that has been considered by several workers utilizes as each cell function a majority or minority element, usually of three inputs. Such majority gates have been considered by Minnick and Short (4), Canaday (5), and by Amarel, Cooke and Winder (6). In 1964, Miyata developed several methods for synthesizing arbitrary combinational functions in terms of arrays of three input minority or majority cells (7). Canaday has also developed a class of canonical synthesis methods that are based on various ways for decomposing switching functions to accomplish the same result.

The use of variable cell function cells will result in a more flexible cellular array. An early effort to utilize variable function cells was proposed by Gluckman at AFCRL (8), in which each cell was selected to be an OR gate, NOR gate, or a crossover.

In 1962, Maitra (9) considered one-dimensional arrays in which each cell can be any one of the 16 functions of two variables. This cellular array is known as the Maitra cascade, general function cascade, or as a tributary network. Several cellular arrays have been based on the Maitra cascade, the first of which appears to be the cutpoint array (10) developed by Minnick in 1964. The array consists of a number of vertical Maitra cascades, interconnected by common horizontal busses. Each cell in the cutpoint array can produce any of eight combinational switching functions or an R-S flip flop. Continuing work by Minnick (11) resulted in the cobweb array in 1965. In this cellular array, cell parameters are used both for choosing among the cell functions, and for changing the interconnection structures. Logical design and fault avoidance algorithms were also developed for the cutpoint and cobweb cellular arrays.

The detailed development of microcellular work is presented by Minnick (12).

As concluding remarks, the survey of microcellular work has yielded two important design considerations. The first is that in designing a cellular processing array, as much of the structure of the problem as possible should be designed into the array interconnection structure. The second consideration is that to implement a processing block in cellular form, a select set of functional cells should be used. Hence, any logical function must be realizable with this canonical set. These concepts are applied to the logical design of the specially-organized computer. In the next section a survey of array-structured computer designs is presented with emphasis placed on features pertinent to the processing of the selected class of problems.

1.3 Survey of Array-Organized Computers

Review of available literature on specially-structured computing systems reveals a number of array-organized computers. Several pertinent computer organizations are discussed in the following sections.

1.3.1 The Unger Spatially-Oriented Computer

One of the first examples of a specially-organized computer was proposed by Unger (13) in 1958. This machine is a stored program computer which can handle spatial problems by directly operating on information in planar form without scanning or format conversion. Hence, this machine is specially organized to efficiently handle pattern detection problems, such as detection of a lower left corner of a binary pattern.

The structure of the proposed machine is shown in Figure 1.1. It consists of a master control unit and a rectangular array of modules. Each module communicates with its four nearest neighbors and receives command signals from the master control. The master control consists of a random access memory for storing instructions, a clock, and decoding circuits, and issues commands in parallel to all modules; it cannot address modules individually. The machine is programmed with a set of 14 assembly language instructions.

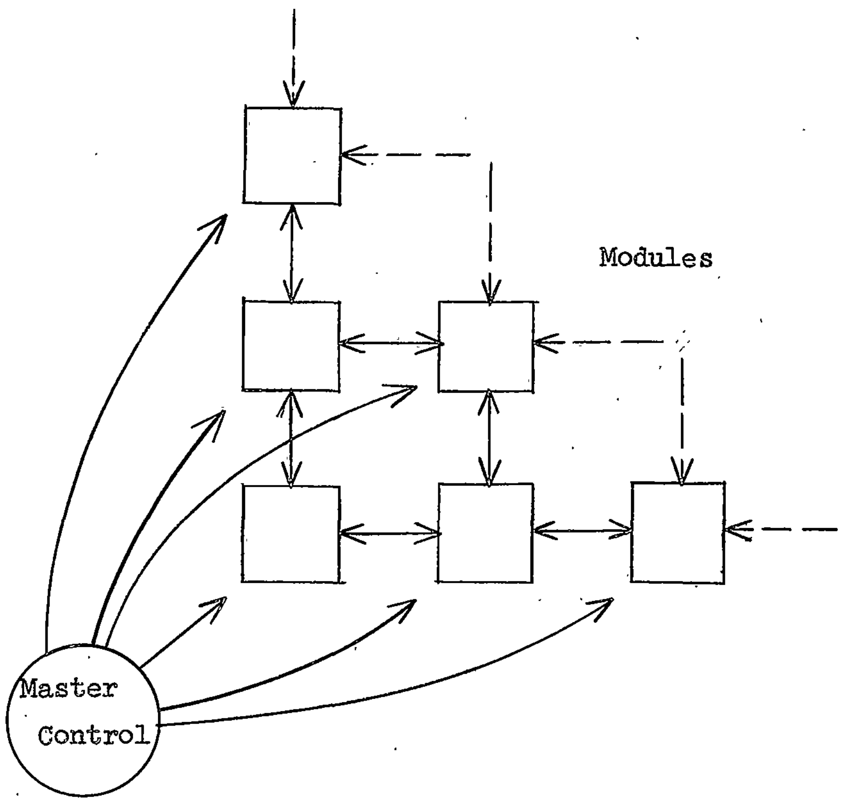

Figure 1.1. The Unger Spatially-Oriented Computer

Each module in the processing array consists of a one bit accumulator, a small amount of random access memory (six bits in one bit words), and some associated logic. Inputs to each module come from the master control and from the accumulators of the nearest neighbor modules.

A logical adder (OR gate) with an input from each module accumulator tells the master control if all of the accumulators contain zero, making possible a transfer on zero order. This instruction, analogous to the conditional transfer orders used in conventional computers, indicates to the master control to skip to the instruction indicated by the transfer zero order. This is the only decision dependent command present in the machine.

It is in this machine organization that one of the first examples of a processing array controlled in parallel by a central control unit is found.

1.3.2 The Holland Computer

A computer organization was described by Holland (14), establishing system control at the local level in the processing array. This organization is in direct contrast to the central control concept suggested by Unger.

The purpose of this computer organization was to provide a basis for investigation into computability and the theory of automata. It consists of a two-dimensional array of identical modules, each module containing a storage register, routing logic, and auxiliary registers. At any given time a module will be active or inactive. If the module is active, it treats the contents of its storage register as an instruction and proceeds to execute the instruction. After a module has executed its instruction it passes its active status on to its successor, which may be any of its four nearest neighbors in the array. With this computer structure, sequences of instructions are arranged spatially throughout the array of modules, with an arbitrary number of sequences being executed at any given time.

The operation cycle of this computer consists of three phases. In the first phase, module storage registers may be set to values provided by an external source. In the second phase, active modules determine the locations of their operands by causing paths to be logically enabled. The third phase causes the instructions in the storage registers of all active modules to be executed.

This computer appears to be difficult to program efficiently so that more than a small number of array modules are active at any given time. Also, a large amount of hardware is required to accomplish reasonable computing tasks.

One of the important results of this work is the development of an array of locally controlled identical processing modules. However, the utilization of such a processing array results in large hardware expense and a computer that is difficult to program.

1.3.3 The Comfort Computer

The concept of an array-structured computer utilizing local system control was further considered by Comfort (15). The result was a modified Holland computer structure having the form of a fixed size rectangular array of modules. Each module consists of two relatively independent sections; the memory and control section, and the communications section. At one side of the array of modules is a set of arithmetic units. These units do all the mathematical and logic computations. Since the machine contains no central control, each module executes its own instruction when it is enabled. The execution of a sequence of instructions causes the enabling and disabling of successive modules.

This computer was organized to have some important improvements to the Holland computer, which are noted below:

1. Programmability has been improved by several orders of magnitude.
2. Machine size has been reduced by a factor of five.
3. Hardware utilization has been improved by a factor of three.

A possible degradation in system performance may occur, since only one program sequence per arithmetic unit can be executed at one time.

1.3.4 The SOLOMON Computer

The SOLOMON (Simultaneous Operation Linked Ordinal Modular Network) computer is a parallel processing computer introduced by Slotnick, Borck, and McReynolds in 1962 and later revised in 1966 (16). This computer organization was the first attempt to satisfy a class of problems limited by current computing system capabilities at that time. This computer was specially organized to efficiently implement matrix or mesh equations. This class of problems is typified by linear system analysis, matrix calculations, and solutions to systems of ordinary and partial differential equations.

The computer, as shown in Figure 1.2, is composed of three major units. The first is the network control unit (NCU) which is a central control for the machine. The NCU can consist of a minimum of one arithmetic and control subunit, and can be expanded to multiple subunits. The second is the processing array composed of 32 x 32 processing elements (PE). The PE array is designed so that modules of 256 PE's including the associated PE memories can be added or removed from the basic system without design change. The third is the input-output unit (IOU) which consists of five modules of 32 data channels each (a channel being an input-output data link).