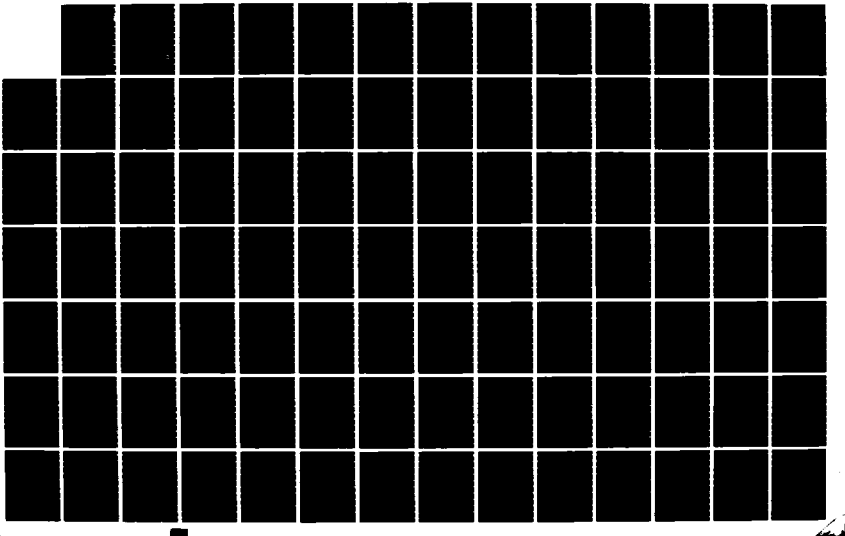
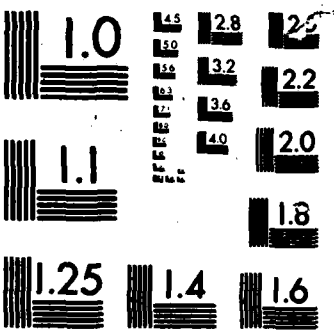
AD-A167 336 DISTRIBUTED COMPUTING FOR SIGNAL PROCESSING: 1/3
TOPOLOGICAL PROPERTIES OF IN..(U) PURDUE UNIV LAFAYETTE
IN R R SEBAN DEC 85 ARO-18790.17-EL-APP-E
UNCLASSIFIED DAAG29-82-K-0101 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.2
2.0
1.8

TOPOLOGICAL PROPERTIES OF INTERCONNECTION NETWORKS FOR PARALLEL PROCESSORS

Ph.D. Thesis by:
Robert R. Seban

Faculty Advisor:
Howard Jay Siegel

Appendix E for

Distributed Computing for Signal Processing: Modeling of Asynchronous Parallel Computation Final Report

U.S. Army Research Office
Contract No. DAAG29-82-K-0101*

*Chapters 1 through 8 supported by this contract.

TOPOLOGICAL PROPERTIES OF INTERCONNECTION NETWORKS FOR PARALLEL PROCESSORS - A UNIFIED APPROACH

A Thesis

Submitted to the Faculty

of

Purdue University

by

Robert R. Seban

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 1985

ACKNOWLEDGMENTS

I would like to thank my major advisor Professor H. J. Siegel and my committee members Professor E. J. Coyle, Professor S. E. Hambrusch, and Professor D. G. Meyer for their time and effort.

Chapters 4 and 5 were supported in parts by the U.S. Army Research Office, Department of the Army, under contract number DAAG29-82-K-0101. Chapters 6, 7, and 8 were supported in parts by a David Ross Grant 1984-1985, under contract number 0857-56-12855, and by the U.S. Army Research Office, Department of the Army, under contract number DAAG29-82-K-0101. Chapter 9 was supported by IBM Federal Systems Division, under contract number 289662B-YD. Chapter 10 was supported in parts by National Science Foundation Grant under contract number ECS-8120896.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Seban, Robert R. Ph.D., Purdue University. December 1985. TOPOLOGICAL PROPERTIES OF INTERCONNECTION NETWORKS FOR PARALLEL PROCESSORS - A UNIFIED APPROACH. Major Professor: Howard Jay Siegel.

Two methods are used to speed up the execution of a computational task. One is new technology development and the other is the exploitation of parallelism in the computation. To take an advantage of the parallelism in a task requires the utilization of parallel computer architectures. At a certain high level of abstraction a parallel computer system is represented as a graph where the nodes represent processors, memories, or other devices, and the edges represent the communication links.

In this research the following problems of parallel processing are studied. First is a theoretical study of topological properties of interconnection networks. Second is a case study of a network design for a real-time system. Lastly, the use of SIMD networks for performing "shuffles."

A general model that can be used to describe networks and systems with arbitrary topologies is developed. Based upon the of morphism of groups, the concept of morphism of systems is developed. The morphism of systems is called quasimorphism and allows a method of comparison between topologically arbitrary parallel computer systems. The quasimorphism is used to study the emulation of one system by another.

The composition, decomposition, and partitionability of single stage networks are studied. Informally, the partitionability property means that the

network can be divided into several parts each with a degree of independence. The synthesis of single stage partitionable interconnection networks is examined. The applications of the model to multistage networks is discussed.

A case study of the design of a network for a real-time signal processing system is performed. A network and network interfaces are designed for a distributed digital signal processing system subject to high throughput, extendibility, fault tolerance, and other constraints.

The data permuting ability of single stage SIMD networks are studied. Specifically, algorithms for the PM2I and Illiac networks to perform the "shuffle" data permutations are developed.

1 INTRODUCTION

Two methods are used to speed up the execution of a computational task. One is new technology development and the other is the exploitation of parallelism in the computation. To take advantage of the parallelism in the task requires the utilization of parallel computer architecture [KuL78, ThW75]. There are two major classes of parallel computer system architectures, loosely coupled, where the information transfer is infrequent, and tightly coupled, where information transfer is frequent, perhaps every operation cycle. In this research the primary concern is the class of tightly coupled parallel computer systems.

At a certain high level of abstraction a parallel computer system is represented as a graph where the nodes represent processors, memories, or other devices and the edges represent the communication links. This representation is frequently used by researchers and is based upon the belief that one of the salient features of a parallel computer system is the topology of the interconnection network and the way the processors and other devices are connected to it. Although the graph depiction of the system contains large amount of information, it does not convey the dynamic structure of a reconfigurable network. Our model developed in this research embodies that information.

Much research has been devoted to study several topologically regular interconnection networks. Amongst the best known networks are Illiac [BoD72], Shuffle [LaS76], Omega [Law75], multistage Cube [AdS82b], STARAN [Bat76], ADM [McS82], k-connected mesh [NaS80], and PM2I [SeS84b]. The researcher usually proceeded as follows: he devised a model for the network of interest and derived analytical results based on that model. This approach has

the drawback that the results are network specific since the model is network specific and sometimes implementation dependent.

Our research differs from the past work in several aspects. First, a unified approach to the analysis of interconnection networks that is valid for large classes of interconnection networks was developed. Second, several algorithms that allow systematic analysis and design of networks with the desired property of partitionability will be developed. In more detail, the following related topics of topological properties of parallel computer systems will be studied.

In Chapter 3, the background of parallel computer architecture is presented. Numerous parallel computer systems have been discussed in the literature and proposed, and several have been built. Parallel systems are divided into two major classes, tightly coupled and loosely coupled. The subject of analysis here is the tightly coupled parallel systems group which can be divided into several categories.

It is shown that each type of parallel computer architecture requires one or more interconnection networks. Some systems use networks dedicated to the communication between particular subsystems, some other systems use a single network multiplexed for communication among different parts of the system. In an ensemble parallel system the network is used by the control unit to broadcast instructions and data [ThW75]. In a pipelined system the interconnection network is used to provide data communication among the computational units (segments) of the pipeline [Bae80]. In vector and array parallel system one network is used for interprocessor communication and a usually separate network is used by the control unit to broadcast data, instructions, and control information to the processors [BaB68]. In a systolic system the network is used to propagate the wave of partial results from a set

of processors to the next set of processors [KuL78]. In an associative system the control unit uses the network to broadcast selected data fields to the processors for comparison, and in some cases another network is used for interprocessor communications [Bat74]. Reconfigurable systems have a network that allows the system to be statically or dynamically restructured into multiple machines of different sizes [SiS84]. A data flow system consisting of multiple rings needs a communication network to move data among rings [WaG82].

In Chapter 4 a general model of single stage interconnection networks is developed [SeS84a]. This model is sufficiently general so that it can be used to model networks with an arbitrary topology, including both regular and irregular topologies. The model is independent of the method of implementation of the network. This is necessary because properties of networks such as similarity relationships, emulation, and partitionability of networks are implementation independent.

The model together with additional information is then used to construct a model for parallel computer systems. A system, informally, consists of a set of devices, an interconnection network, and a method for use of the network. Each device is assumed to have two logical ports, an input port and an output port, possibly implemented physically as the same set of I/O pins. Some examples of devices are processors, memories, or processor/memory pairs. Based upon the use of the network, three types of systems, recirculating, nonrecirculating, and partially recirculating, are defined. Relationships between systems such as equality and three types of subsystems are rigidly defined and their properties explored.

In Chapter 5 a technique is developed to measure the similarity of two systems. This generalizes the past work on similarity used by many researchers which classifies the relationships between two networks into two kinds only, (a) the networks are isomorphic or (b) the networks are not isomorphic. The measure has a number of uses and is applied in this chapter to the analysis of emulation. Our definition of emulation is a generalized case of the one described in [FiF82].

Previous work, related to our research developed here, can be found in the classification of groups in the field of abstract algebra and group theory [Han68, Her75]. The theory of group classification is based upon the concept of morphism. Morphism measures the similarity of behavior between group operations of two groups. This measure ignores the labeling of the elements of the groups and is concerned strictly with the structure which is determined by the group operation.

Based upon the idea of morphism of groups, the concept of morphism of systems is developed [SeS84a]. In the domain of parallel computer systems the structure of interest is the structure of the correspondences of the system's network in the graph theoretical sense. The morphism of systems is called quasimorphism and allows a method of comparison between topologically arbitrary parallel computer systems. The quasimorphism facilitates the analysis of following problems in parallel processing: system emulation, multiple mapping of a problem into a system for increased reliability, and partitioning of systems. The quasimorphism is analyzed with respect to properties similar to the properties of reflexivity, symmetry, and transitivity.

Also in this chapter the problem of emulation of one system by another is discussed. Three different types of emulation are considered. Several efficiency

measures of the emulation were defined and the three types of emulation were evaluated using these criteria.

In Chapter 6 the composition, decomposition, and partitionability of single stage networks are studied [SeS85]. Informally, the partitionability property means that the network can be divided into several parts each of which has certain degree of independence. The type of partitionability analyzed in this chapter has three subtypes.

The partitionability property of interconnection networks in the context of parallel computer systems has the following advantages, besides being interesting from the theoretical point of view.

(1) If the network is partitionable then the resource allocation of only a subset of the total resources is possible. This can be used as follows.

   (a) A user can utilize only a small part of the machine for program development phase.

   (b) In a multiple user environment the partitioning provides a natural protection among users.

   (c) In a multitasking environment the partitioning provides a protection among independent tasks.

(2) If the network is partitionable, the fault tolerance of the system increases as follows.

   (a) A method of graceful degradation is possible by separating the faulty section from the correctly operating ones.

   (b) If in addition to being a partitionable network, the sections are isomorphic, then an increase of reliability may be realized by multiple mappings of the same task onto the multiple sections and tandem

cross checking of partial results.

(c) It is possible to construct a link and switching element fault tolerant network using a partitionable network as a core.

(3) If the network is partitionable, then there is an efficient implementation in terms of hardware and control. The network can be implemented as a set of network components each with its own set of inputs and outputs. Consequently the data path layout and in some instances the control lines layout on VLSI chip or on a printed circuit board can be simplified.

An algorithm to classify partitionability of interconnection networks is developed which will output one of the following:

(1) The network is not partitionable.

(2) The network is partitionable into subnetworks with common control signals and the combination of the of the subnetworks will exactly generate all interconnection patterns of the original network.

(3) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will exactly generate all interconnection patterns of the original network.

(4) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will generate a superset of interconnection patterns of the original network.

The algorithm is general in the sense that it will accept as an input a topologically arbitrary interconnection network.

In Chapter 7 the synthesis of single stage interconnection networks with the partitionability property is studied. Several different techniques are developed, each of which can be used to construct a large class of single stage

partitionable networks. The algorithms are presented for a simplified case, but they can be easily generalized in a number of different ways.

In the first part, of this chapter an algorithm to generate a large class of partitionable networks is developed and proven correct. This algorithm is based upon the results of the analysis presented in Chapter 6.

The second part of this chapter discusses the problem of synthesis of a special case of partitionable networks. This special class of networks consists of those networks that are isomorphic to a direct product of groups [Han68, Her75]. Since these groups have been studied in the abstract algebra extensively, techniques are known to determine the possibility of decomposition of a given group into a direct product of groups.

In Chapter 8 the analysis of multistage networks will be addressed. This extends the work done in Chapter 6 into the domain of multistage interconnection networks.

First a method of composition of single stage networks is presented and its properties studied. Using the composition of single stage networks, the multistage model is defined. This approach has the advantage that some results of analysis of single stage networks can be applied to the study of the multistage networks. The model is very general since each stage consists of the general single stage model presented earlier. Several examples of an application of the multistage model are presented.

In Chapter 9 a case study of a communication system for a real-time, distributed digital signal processing system. Network and network interfaces are designed subject to number of system constraints such as very high throughput, system extendibility, and fault tolerance requirements. For this

application, and given the current and near future technology, a crossbar based interconnection network was selected for the task under consideration. Two different fault tolerant chip architectures are presented. Four network architectures are designed and their characteristics are discussed. Several fault detection and recovery techniques on the system level are developed.

In Chapter 10, a study of shuffle interconnection function emulation by PM2I and Illiac SIMD networks is performed. It was previously shown that a lower bound on the number of transfers needed for the PM2I network to perform the shuffle is $\log_2 N$. The algorithm described here is near optimal and requires only $(\log_2 N)+1$ transfers. Also, an algorithm for the case where there is a machine with a PM2I network and it is desired to emulate a shuffle that is of smaller size than the host network is presented. Using the PM2I algorithm as a basis, an algorithm for the Illiac to emulate the shuffle is given. It requires $2\sqrt{N}-1$ transfers, which is only three transfers more than lower bound of $2\sqrt{N}-4$ shown previously.

2 ORGANIZATION OF THE THESIS

In Chapter 3 an overview of several classes of tightly coupled parallel computer architectures will be given. First the defining features of each class will be presented, and then an example of the class will be discussed in detail. All the examples consist of existing systems or systems in research or design stages which have been described in the literature.

In Chapter 4 the network model is presented. The model together with additional information is then used to define the model of a parallel computer system. Three types of systems based upon the method of use of the network are defined and examples of each category given.

In Chapter 5 a measure of similarity of systems with arbitrary labeling and topology is introduced. The measure is called quasimorphism and is used in this chapter to analyze emulation of one system by another.

In Chapter 6 the horizontal composition and decomposition of interconnection networks are formally defined and analyzed. Using the compositions, three types of partitionable single stage networks are recognized. An algorithm is presented that accepts as an input a topologically arbitrary interconnection network and outputs one of following four outcomes: the network is not partitionable, or the network is partitionable in one of the three types.

In Chapter 7 the synthesis of single stage partitionable networks is studied. An algorithm is presented to synthesize a large class of partitionable networks. In addition, a special class of partitionable interconnection networks that are isomorphic to a direct product of groups is described.

In Chapter 8 the analysis of multistage networks is discussed. Basic definitions such as vertical composition of networks is presented and its properties analyzed. Using composition of single stage networks, the multistage network model is defined and some applications are shown.

In Chapter 9 a network and network interfaces are designed for a real-time, distributed digital signal processing system. The design is subject to number of system constraints such as very high throughput, system extendibility, and fault tolerance requirements. Several fault detection and recovery techniques on the system level are studied, since fault tolerance is a salient issue of this system.

In Chapter 10 the ability of the PM2I and Illiac type single stage SIMD machine interconnection networks to perform the shuffle interconnection was examined. Two algorithms were developed, one for the case of a PM2I of same size as the shuffle and one for the case of a PM2I of a larger size than the shuffle. Both algorithms are near optimal in the number of network transfers. In addition, using the PM2I algorithm as a basis, an algorithm for the Illiac to emulate the shuffle is developed.

3 PARALLEL COMPUTER ARCHITECTURES

3.1 Introduction

One method of speeding up the execution of computational tasks is to use parallel computer architectures which exploit the parallelism in the execution phase of the task. Numerous parallel computer systems have been discussed in the literature and proposed, and several have been built. Parallel systems are divided into two major classes, tightly coupled and loosely coupled. The subject of analysis here is the tightly coupled parallel systems group which can be divided into several categories.

As will be shown, each type of parallel computer architecture requires one or more interconnection networks. Some systems use networks dedicated to the communication between particular subsystems, some other systems use a single network multiplexed for communication among different parts of the system. In an ensemble parallel system the network is used by the control unit to broadcast instructions and data. In a pipelined system the interconnection network is used to provide data communication among the computational units (segments) of the pipeline. In vector and array parallel system one network is used for interprocessor communication and a usually separate network is used by the control unit to broadcast data, instructions, and control information to the processors. In a systolic system the network is used to propagate the wave of the partial results from a set of processors to the next set of processors. In an associative system the control unit uses the network to broadcast the selected data fields to the processors for comparison, and in some cases another network is used for the interprocessor communications. Reconfigurable systems use a network for interprocessor communication and perhaps a different

network for fetching/storing data in the memories. Data flow system consisting of multiple rings needs a communication network to move data among rings.

3.2 Overview

In this chapter an overview of different classes of tightly coupled parallel computer architectures will be given. Each class will be presented as follows. First the defining features of the class will be presented, and then a representative system of the class will be discussed. All the examples consist of existing systems or systems in research or design stages described in the literature. For a good survey of systems see [HaL82] and of interconnection networks see [Sie85].

3.3 Problem Statement

Several categories of parallel computer architectures will be defined. This will be followed by a detailed description of an example of architecture in each category. The description of the system will demonstrate that each category of parallel computer architecture described uses one or more interconnection

network as well as show the different ways the networks are used by the system.

3.4 Parallel Computer Architecture Classes

The Ensemble Processors achieve the speedup of execution of computational task by utilizing many processing elements each of which is operating on an independent data stream. The system does not use an interprocessor interconnection network, however, the control unit uses an interconnection network to transfer data and instructions to the processors.

A representative of this group is the Parallel Element Processing Ensemble (PEPE) [ThW75, ViC78], whose design can support up to 288 processors. PEPE was developed to handle the tracking of multiple targets and as such it must compute identical operations on large number of independent data streams. These data streams are radar signal returns of possibly multiple objects entering the radar's surveillance volume. PEPE also uses an associative operation to locate the file of a target given its new data coordinates. This operation is implemented by broadcasting of the new data from the control unit to the processors using the interconnection network. If a correlation is found between new data and a file in a processor then the new information is added to the file, otherwise an idle processor will be allocated for a new target.

The pipelined processors (MISD mode) achieve speedup of computation by (a) breaking the instruction into a sequence of smaller operations and (b) executing concurrently the smaller operations using several computational units. The flow of data is such that unit u_i executes its subtask and passes the data to unit u_{i+1}, hence the term pipeline. Some systems that fall into this category are TI ASC [Bae80, Sto80, The74], CRAY 1 [KoT80], and CYBER 205 [Bae80, KoT80].

The TI Advanced Scientific Computer (ASC) consists of an instruction unit and from one to four processing units. The instruction unit is constructed as a four stage pipeline and the stages are: instruction fetch, instruction decode, effective address calculation, and register operand fetch. All processing units are identical and each consists of eight stages, however, using a dynamic reconfiguration (via a network) a custom pipeline can be constructed from the basic eight elements. The stages of the processing unit are: input, exponent subtractor, prenormalizer, multiplier, adder, normalizer, accumulator, and output.

The vector and array processors (SIMD Mode) achieve speedup of computation by using a large number of computational elements. Examples of their applications include the image processing, such as filtering and convolution, and in matrix operations for the weather prediction or simulation. Some examples of these systems are Illiac IV [BaB68, BoD72], MPP [Bat80], Cartesian Moment Computer (CMC) [ReS82, Seb82], and BSP [KoT80]. Two examples will be discussed, the MPP and the BSP.

The Massive Parallel Processor (MPP) consists of $128 \times 128 = 16384$ simple processing elements. Each element processes data one bit wide (bit serial). Each processor communicates with other processors in the array using

the four nearest neighbor interconnection network. The processing array uses staging memories to reorder the data received from a satellite into a form where each processor receives all the bits of the grey value of one pixel in the image.

Burroughs Scientific Processor (BSP) consists of 16 arithmetic units, each capable of operating on 48 bit words. There is an input alignment network to move data from the 17 memory units to the 16 arithmetic units and an output alignment network to move the data from the 16 arithmetic units to the 17 memory units. The alignment network allows a 16×16 matrix to be stored in the 17 memories in such manner that row, column, diagonal, and many other substructures of the matrix can be fetched/stored without an accessing conflict [BuK71].

The systolic arrays or wavefront processors [KuL78, Kun82] receive the name from their mode of operation which can be described as follows. The systolic arrays are usually organized as one or two dimensional arrays of simple processors, each connected to its neighbors in some regular way (two, three, four, or six nearest neighbors). Each processor repeatedly executes the same operation on data as it is pipelined through the systolic array, creating partial results. Each partial result is passed to a neighboring processor which will use the partial result and additional (partial) results to create a more complete result until finally at the output edge of the array the final result is outputted.

The associative processors achieve the speedup by operating in parallel on a large number of records that are selected based on the value of a field in the record. Examples in this category are STARAN [Bat74, FeF74, RoP77], OMEN [Hig72], and ALAP [YaF77]. The ALAP will be described here.

The Associative Linear Array Processor (ALAP) consists of a linearly connected array of processors that receive common data and commands from the control unit. Their matching line outputs are "or"ed together to notify the control unit if there is a match to the input data. A VLSI system consisting of 13 processors was constructed and tested. A bus is used to input individual as well as common data into the processors, therefore it would become a bottleneck if a large number of processors were used.

The reconfigurable systems consist of a large number of processors which communicate through a reconfigurable interconnection network. Some examples of this category are PASM [SiS79, SiS81, SiS84], TRAC [KaP80, SeU80], and CHIP [Sny82]. The PASM system will be described here.

The partitionable SIMD/MIMD (PASM) system is currently under development in Purdue University, School of Electrical Engineering. The system includes $Q = 2^q$ Micro Controllers (MCs), and the Parallel Computation Unit (PCU) which is comprised of $N = 2^n$ processors, N memory modules, and an interconnection network.

The system's strength lies in its ability to allocate a subset of its N processors to a particular task. For details on the allocation strategies see [TuS83]. It is intended to be used in image processing and pattern recognition applications. The collection of resources consisting of RN/Q processors, ($R = 2^r$, $0 \leq r \leq q$) together with R Micro Controllers and RN/Q memories is called a virtual machine. The actual processors selected for a given virtual machine depend upon the type of partitionability of the system and the partition selected. The type of partitionability is a function of the inter-processor interconnection network. The virtual machines are independent of each other, consequently different machines can execute different jobs

concurrently. The current status of PASM is the logic design phase and building of a small prototype of 16 processors and four MC's using off shelf logic devices.

Data flow system achieves computational speedup by exploiting the parallelism at the instruction level [WaG82]. Conceptually, each instruction is translated into a template consisting of an operation and data slots. An instruction gets executed if its data are available and a processor is available.

Data flow computers are usually implemented as rings, each ring consisting of at least the following blocks: a token queue, a matching store, and a processing unit. The "token queue" saves results generated by the processing unit. The "matching store" tries to match incoming tokens from the token queue with the slots of templates currently residing in the matching store. The "processing unit" accepts the instruction template with all its fields resolved and executes the operation, passing the results to the token queue. Since multiple rings each consisting of a token queue, a matching store, and a processing unit are used for speedup of the execution, a token generated in one ring may be needed as a data in a template residing in the matching store of another ring. In order for the token to move from one ring to another an interconnection network must be used to connect the data paths of different rings.

3.5 Conclusions

In this chapter an overview of several major classes of tightly coupled parallel computer systems was presented. Each category of parallel computer system was described in sufficient details to show that an essential part of each system is one or more interconnection network. The usage of the interconnection network varies from system to system. Some systems use networks dedicated to the communication between particular subsystems, some other systems use a single network multiplexed for communication among different parts of the system. In an ensemble parallel system the network is used by the control unit to broadcast instructions and data. In a pipelined system the interconnection network is used to provide data communication among the computational units (segments) of the pipeline. In a vector and array parallel system one network is used for interprocessor communication and a usually separate network is used by the control unit to broadcast data, instructions, and control information to the processors. In a systolic system the network is used to propagate the wave of the partial results from a set of processors to the next set of processors. In an associative system the control unit uses the network to broadcast the selected data fields to the processors for comparison, and in some cases another network is used for interprocessor communications. Reconfigurable system uses a network for interprocessor communication and perhaps a different network for fetching/storing data in the memories. Data flow system consisting of multiple rings needs a communication network to move data among rings.

When a designer is facing the problem of selecting a parallel computer system for a particular task or a class of tasks, then several properties of the network becomes of interest. These properties are heavily dependent upon the topology of the network and therefore the study of the topological properties of networks is an important method of evaluation and classification of parallel computer systems.

4 MODELING OF NETWORKS AND SYSTEMS

4.1 Introduction

Most current analytical techniques for interconnection networks and modeling techniques of networks are concentrated on the analysis of topologically regular interconnection networks. Examples of such networks are Illiac [BoD72], Shuffle [LaS76, SeS84b], multistage Cube [AdS82b], STARAN [Bat74], ADM [McS82], k-connected mesh [NaS80], and PM2I [SeS84b]. The past research usually proceeded on the following lines. A network specific model is defined and then analytical results are derived using this model. The problem with this approach is that the results developed are problem specific, that is to say, the results are valid only for the small class of networks that the model represents. One way to generalize the results of the analysis is to develop a general model describing the topology of the network.

In this chapter, the following problems are discussed. A general problem of modeling networks with arbitrary topology is developed [SeS84a]. This model is sufficiently general so that it can be used to model networks with arbitrary including regular and irregular topology. The model is independent of the method of implementation of the network. This is necessary because properties of networks such as similarity measures, emulation and partitionability of networks are implementation independent. The similarity measures between two networks is classified into several classes. This is a refinment of the old system which classified the similarity measure between two networks into two classes only, isomorphic and nonisomorphic. The model of network together with additional information is then used to construct a model for parallel computer systems. A system, informally, consists of a set of

devices, interconnection network, and a method of use of the network. Each device is assumed to have two logical ports, an input port and an output port, possibly implemented as the physically same set of I/O pins. Some examples of devices are processors, memories, or processor/memory pairs. Based upon the use of the network, three types of systems, recirculating, nonrecirculating, and partially recirculating are defined. Relationships between systems such as equality and three types of subsystems are rigidly defined and their properties explored.

4.2 Overview

This chapter is organized as follows. In section 4.3 definitions of the problems addressed in this chapter are given. In section 4.4 the previous related work is briefly described. In section 4.5 the basic concepts are defined. In section 4.6 the network model is presented, several major relationships between networks described, their properties given and some examples of applications presented. In section 4.7 the concept of a parallel computer system is formally introduced. Three types of systems based upon the method of use of the network are defined and examples of each category given. Several similarity measures between two systems are defined and examples presented. In section 4.8 the conclusion and summary of the chapter is given.

4.3 Problem Statement

In this section, an informal description of the work presented in this chapter is given. The descriptions will be informal only, as the basic mathematical concepts have not been defined yet and will be introduced later in this chapter. The following problems of analysis of interconnection networks are addressed in this chapter. In order to analyze the topological properties of interconnection networks a model must be developed. In this chapter, a general model of topologically arbitrary interconnection networks is presented which will be used through most of this research [SeS84b]. This model is implementation independent, which is desired since the topological properties of networks are implementation independent, moreover if the model were implementation dependent that would reduce its scope of applicability to the class of networks having that implementation. Next, several important relationships between networks such as equality and two types of subnetworks are defined and their properties shown. In the next section the model for a parallel computer system is defined. A system, informally consists of a set of devices, an interconnection network, and a use of the network. Each device (processor, memory or processor/memory pair) is assumed to have two logical ports, one input port and one output port. Based upon the method of use of the network, three types of systems, recirculating, nonrecirculating, and partially recirculating are defined. Relationships between systems such as equality and three types of subsystems are rigidly defined and their properties explored.

4.4 Previous Work

In this section, the previous work is briefly described. Previous work on modeling of interconnection networks in [Gok76, GoL73, LiM82, Upp81] was used to describe the class of SW Banyans networks. The model is based on graph theory and is sufficiently general to describe the class of SW Banyans, however it is implementation dependent, which narrows down the the scope of its applicability. Some issues discussed using the model were mapping methods of simple regular interconnection networks such as ring or a tree onto the Banyan networks. Additional work on modeling of regular networks, such as mesh, shuffle, Cube and PM2I was done in [FiF82], and was network specific. In [FiF82] a specific class of networks called quotient networks was discussed. Informally, a quotient network is a network that is homomorphic to the same type of network of a smaller size in terms of processors. A class specific model was developed in [RaF83] for the evaluation of a class of linear array-processor systems for VLSI implementation. A general model was developed for the analysis of time space tradeoff of interconnection network in [MaM81b]. A good overview of interconnection networks can be found in [Sie85, WuF84].

4.5 Basic Concepts

In this section, basic definitions and notation needed as the background for the rest of the paper are introduced. Some of the definitions can be found in books on basic abstract algebra [Han68, Her75] and graph theory [BoM76, Har69], however are included here for completeness. The purpose of these definitions is to develop a formal notation that will be used to discuss more complex concepts such as networks and systems. To relate these definitions to the subject at hand, some examples are given in the end of this section.

Let the set of input labels of a graph/algebraic structure be denoted by V_I and the set of output labels of the structure be denoted by V_O. All graph/algebraic structures defined in this paper over $V_I \times V_O$ will assume that $V_I \cap V_O = \emptyset$, $V_I \neq \emptyset$, $V_O \neq \emptyset$, where $\emptyset$ is the *empty set* and $V_I \times V_O = \{<v_a, v_b> \mid v_a \in V_I, v_b \in V_O\}$.

The following notation will be used throughout this paper. The symbols are enclosed in a pair of double quotation marks.

"{","}" - delimiters for set. "(",")" - function application and grouping of operations. "<",">" - delimiters for n-tuple.

"[","]" - used as defined in context.

Definition 4.5.1:

Let A be a set, then $P[A] \triangleq \{B \mid B \subseteq A\}$ is the *power set of A*.

Definition 4.5.2:

Let $C_m \in P[V_I \times V_O]$, then C_m is an *I/O correspondence over* $V_I \times V_O$.

Definition 4.5.3:

Let $C_m \in P[V_I \times V_O]$ such that $\langle v_a, v_b \rangle, \langle v_c, v_d \rangle \in C_m \rightarrow v_b \neq v_d$, then the C_m is a *nondestructive I/O correspondence over* $V_I \times V_O$. (Physically, C_m represents one state of a reconfigurable network).

Definition 4.5.4:

Let $C[V_I \times V_O] \triangleq \{C_m \in P[V_I \times V_O] \mid C_m$ is nondestructive$\}$. Then $C[V_I \times V_O]$ is called the *C-set over* $V_I \times V_O$.

The definitions 4.5.5 to 4.5.8 discuss the connectivity or accessibility aspects of the I/O correspondences.

Definition 4.5.5:

Let $C_m \in C[V_I \times V_O]$, then $s(C_m) \triangleq \{v_a \mid \langle v_a, v_b \rangle \in C_m\}$ is the *source set of* C_m.

Definition 4.5.6:

Let $C_m \in C[V_I \times V_O]$, then $d(C_m) \triangleq \{v_b \mid \langle v_a, v_b \rangle \in C_m\}$ is the *destination set of* C_m.

Definition 4.5.7:

Let $C = \{C_m \mid m=1,2,...n\} \subseteq C[V_I \times V_O]$, then $s(C) \triangleq \bigcup_m s(C_m)$ is the *source set of C.*

Definition 4.5.8:

Let $C = \{C_m \mid m=1,2,...n\} \subseteq C[V_I \times V_O]$, then $d(C) \triangleq \bigcup_m d(C_m)$ is the *destination set of C.*

Example 4.5.9:

Let $V_I = \{v_0,v_1\}$, and set $V_0 = \{u_0,u_1,u_2,u_3\}$. Consider the set $A = \{<v_0,u_0>, <v_0,v_2>,<v_1,u_3>\}$. What type of correspondence it is.

Solution:

(a): Clearly $A \subseteq P[V_I \times V_O]$, therefore A is an I/O correspondence over $V_I \times V_O$.

(b): $<v_0,u_0>,<v_0,u_2> \in A$, and $u_0 \neq u_2$. $<v_0,u_2>,<v_1,u_3> \in A$, and $u_2 \neq u_3$. $<v_0,u_0>,<v_1,u_3> \in A$, and $u_0 \neq u_3$. Therefore A is a nondestructive I/O correspondence over $V_I \times V_O$.

(c): The source set of A is $\{v_0,v_1\} = V_I$.

(d): The destination set of A is $\{u_0,u_2,u_3\} \subset V_O$.

Example 4.5.10:

Let V_I and V_O be two sets, $V_I = \{v_0,v_1\}, V_0 = \{u_0,u_1,u_2,u_3\}$. Consider the set $B = \{<v_0,u_0>, <v_0,u_2>, <v_1,u_2>\}$. What type of correspondence it is.

Solution:

(a): Clearly $B \subseteq P[V_I \times V_O]$ therefore B is an I/O correspondence over $V_I \times V_O$.

(b): $\langle v_0,u_2\rangle,\langle v_1,u_2\rangle \in B$ and $u_2 = u_2$. Therefore B is not nondestructive I/O correspondence over $V_I \times V_O$, and B could not represent a state of a reconfigurable network.

4.6 Interconnection Network Model

In this section, a formal graph/algebraic model of an interconnection network is presented. Graph models for analyzing networks have been used by other researchers. For example, in [Gok76, GoL73, LiM82, Upp81] they are used to analyze regular SW Banyan networks, and in [FiF82] they are used to study the partitioning of regular networks. The model presented here differs from [Gok76, GoL73, LiM82, Upp81] and [FiF82] by being completely general so that it can be used to describe an arbitrary, topologically regular and irregular, interconnection network.

Certain relationships between networks that are of interest to the computer system designer are presented here in a rigid mathematical fashion. In particular, the relationships subnetworks and equality are defined and their properties described. In the end of the section, some examples of applications are presented and an example is generalized into a theorem.

Definition 4.6.1:

Let $K = \langle C \rangle$ be such that:

(1) $C \subseteq C[V_I \times V_O]$.

(2) $V_I = s(C)$.

(3) $V_O = d(C)$.

(3) $|C| \geq 2$.

Then $K = \langle C \rangle$ is an *I/O representation of a reconfigurable network over* $V_I \times V_O$.

Physical implications: $\langle v_a, v_b \rangle \in C_m$, $C_m \in C$ represents the network moving data from input v_a to output v_b when the state of the network is C_m. C represents the set of all possible states of the reconfigurable network. For an example of a topologically arbitrary interconnection network see Figure 4.1. The example has the following parameters:

$V_I = \{u_a, u_b, u_c\}$, $V_O = \{v_0, v_1\}$,

$C_0 = \{\langle u_a, v_0 \rangle \langle u_a, v_1 \rangle\}$,

$C_1 = \{\langle u_a, v_0 \rangle, \langle u_b, v_1 \rangle\}$,

$C_2 = \{\langle u_a, v_1 \rangle, \langle u_c, v_0 \rangle\}$,

$C = \{C_0, C_1, C_2\}$. $K = \langle \{C_0, C_1, C_2\} \rangle$.

Definition 4.6.2:

Let $K[V_I \times V_O] \triangleq \{K \mid K = \langle C \rangle$ is a network over $V_I \times V_O\}$. Then $K[V_I \times V_O]$ is called the *K-set over* $V_I \times V_O$.

The Definitions 4.6.3 to 4.6.5 are used to classify formally the measure of similarity between two networks. The classes are presented here in the order of increasing strictness. Note that these relationships provide a refined scale of

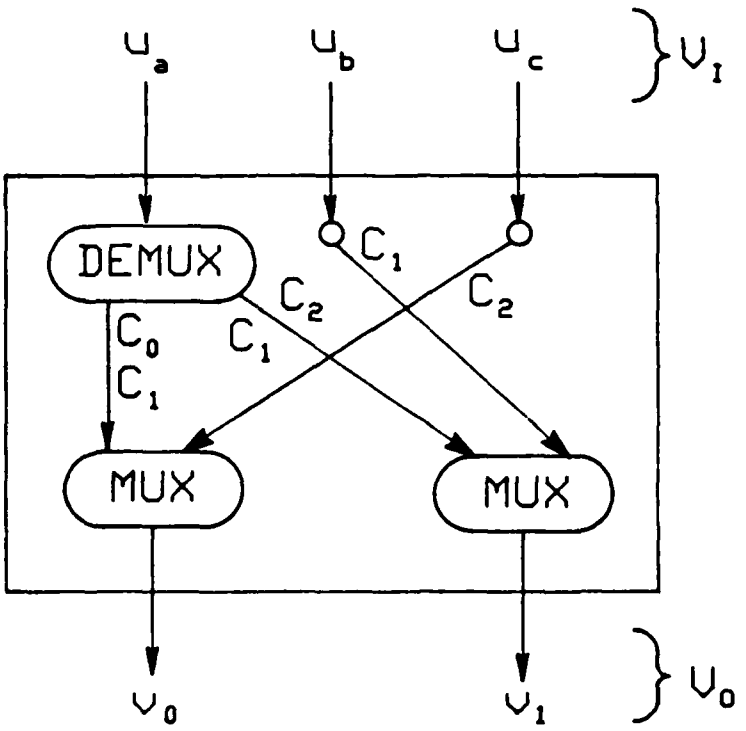


Figure 4.1:
An example of a topologically arbitrary network.

the measure of similarity between two networks compared to the more customary classification of isomorphic/nonisomorphic which was used in the past. Several examples illustrating the application of these measures are given after the definitions are presented. The examples are generalized into a theorem relating the PM2I and k-dimensional Illiac networks.

Definition 4.6.3:

Let $K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that:

(1) $V_I^1 \subseteq V_I^2, V_O^1 \subseteq V_O^2$.

(2) $\forall C_m^1 \in C^1 \; \exists \; C_n^2 \in C^2 \ni: C_m^1 \subseteq C_n^2$.

Then K^1 *is subnetwork of type b of* K^2. *Notation:* $K^1 \subseteq b \; K^2$.

Definition 4.6.4:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that:

(1) $V_I^1 \subseteq V_I^2, V_O^1 \subseteq V_O^2$.

(2) $\forall C_m^1 \in C^1 \;\; \exists \; C_n^2 \in C^2 \ni: C_m^1 = C_n^2$.

Then K^1 *is subnetwork of type c of* K^2. *Notation:* $K^1 \subseteq c \; K^2$.

Note: The reason for referring to these subnetworks as types b and c is to make this notation consistent with the definitions of subsystems in Section 4.7, where the three types of subsystems type a, b, and c are described.

Definition 4.6.5:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that:

(1) $V_I^1 = V_I^2$, $V_O^1 = V_O^2$.

(2) $C^1 = C^2$.

Then K^1 *is equal to* K^2. *Notation:* $K^1 = K^2$.

The Theorems 4.6.6 to 4.6.8 describe the sufficient conditions for the relationships of the different types to exist.

Theorem 4.6.6:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks. If $\forall C_m^1 \in C^1 \; \exists \; C_n^2 \in C^2 \; \ni: \; C_m^1 \subseteq C_n^2$, then $K^1 \subseteq_b K^2$.

Proof:

(1): Show $V_I^1 \subseteq V_I^2$.

$(\forall C_m^1 \in C^1), \; (\exists \; C_n^2 \in C^2) \; \ni: \; (C_m^1 \subseteq C_n^2)$

$\rightarrow (\forall C_m^1 \in C^1, \quad C_m^1 \subseteq C_{g(m)}^2)$

$\rightarrow (\forall C_m^1 \in C^1, \quad s(C_m^1) \subseteq s(C_{g(m)}^2))$

$\rightarrow (\bigcup_m s(C_m^1) \subseteq \bigcup_m s(C_{g(m)}^2))$

$\rightarrow (\bigcup_m s(C_m^1) \subseteq \bigcup_n s(C_n^2)) \rightarrow V_I^1 \subseteq V_I^2$.

(2): Show $V_O^1 \subseteq V_O^2$.

Similar to (1) except replace the s set by the d set.

□

Theorem 4.6.7:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks. If $\forall C_m^1 \in C^1 \; \exists \; C_n^2 \in C^2 \; \ni: \; C_m^1 = C_n^2$, then $K^1 \subseteq_c K^2$.

Proof:

Show $V_I^1 \subseteq V_I^2$ and $V_O^1 \subseteq V_O^2$.

The proofs are similar to proof of Theorem 4.6.6.

□

Theorem 4.6.8:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks. If $C^1 = C^2$ then $K^1 = K^2$.

Proof:

(1): Show $V_I^1 = V_I^2$.

$C^1 = C^2 \rightarrow s(C^1) = s(C^2) \rightarrow V_I^1 = V_I^2$.

(2): Show $V_O^1 = V_O^2$.

$C^1 = C^2 \rightarrow d(C^1) = d(C^2) \rightarrow V_O^1 = V_O^2$.

□

The following examples show an application of the similarity measure between two networks. Note the increasing similarity between the PM2I and k-dimensional Illiac as the dimension k increases. The examples are generalized into a theorem showing what happens at the limit of k as k increases to maximum.

Example 4.6.9:

Consider the Illiac network with N = 64 processors. The network can be modeled as follows.

$V_I^1 = \{u_j \mid j=0,1,...63\}$, $V_O^1 = \{v_k \mid k=0,1,...63\}$.

$C^1 = \{C_0^1, C_1^1, C_2^1, C_3^1\}$.

Let $\oplus$ denote addition modulo 64 and $\ominus$ subtraction modulo 64.

$C_0^1 = \{<u_j, v_{j\oplus 1}> \mid j = 0,1,...63\}$,

$C_1^1 = \{<u_j, v_{j\ominus 1}> \mid j = 0,1,...63\}$,

$C_2^1 = \{<u_j, v_{j\oplus 8}> \mid j = 0,1,...63\}$,

$C_3^1 = \{<u_j, v_{j\ominus 8}> \mid j = 0,1,...63\}$.

Then $K^1 = <C^1>$ describes the network.

Consider the single stage PM2I network with N = 64 processors. The network can be described as follows.

$V_I^2 = \{u_j \mid j=0,1,...63\}$, $V_O^2 = \{v_k \mid k=0,1,...63\}$.

$C^2 = \{C_0^2, C_1^2, ... C_{11}^2\}$,

$C^2_s = \{<u_j, v_{j\oplus 2^s}> \mid s = 0,1,...5;\ j = 0,1,...63\}$.

$C^2_{6+t} = \{<u_j, v_{j\ominus 2^t}> \mid t = 0,1,...5;\ j = 0,1,...63\}$.

Then $K^2 = <C^2>$ describes the network.

What is the relationship between the networks.

Solution:

(a): $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$,

(b): $\forall C_p^1 \in C^1 \quad \exists\, C_n^2 \in C^2 \ni: C_p^1 = C_n^2$. By Theorem 4.6.7 K^1 is a subnetwork of type c of K^2, denoted by $K^1 \subseteq_c K^2$. Since $C_1^2 \notin C^1$, therefore $K^1 \neq K^2$. In the special case of N = 4 the Illiac is equal to the PM2I.

Example 4.6.10:

Consider the generalized three dimensional Illiac system with 64 processors, arranged as a 4 × 4 × 4 matrix. This network can be modeled as follows.

$V_I^1 = \{u_j \mid j=0,1,...63\}$, $V_O^1 = \{v_k \mid k=0,1,...63\}$.

$C^1 = \{C^1_0, C^1_1, C^1_2, C^1_3, C^1_4, C^1_5\}$.

Let $\oplus$ denote addition modulo 64 and $\ominus$ subtraction modulo 64.

$C_a^1 = \{<u_j, v_{j \oplus 4^a}> \mid a = 0,1,2;\ j = 0,1,...63\}$,

$C^1_{3+b} = \{<u_j, v_{j \ominus 4^b}> \mid b = 0,1,2;\ j = 0,1,...63\}$.

Then $K^1 = <C^1>$ describes the network.

Let $K^2 = <C^2>$ be the PM2I network with N = 64 as in Example 4.6.9.

What is the relationship between the networks.

Solution:

(a): $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$,

(b): $\forall C_p^1 \in C^1 \quad \exists\ C_n^2 \in C^2 \ni$: $C_p^1 = C_n^2$. By Theorem 4.6.7 K^1 is a subnetwork of type c of K^2, denoted by $K^1 \subseteq_c K^2$. Since $C_1^2 \notin C^1$, therefore $K^1 \neq K^2$.

Example 4.6.11:

Consider the generalized six dimensional Illiac system with 64 processors, arranged as a 2 × 2 × 2 × 2 × 2 × 2 matrix. This network can be modeled as follows.

$V_I^1 = \{u_j \mid j=0,1,...63\}$, $V_O^1 = \{v_k \mid k=0,1,...63\}$.

$C_a^1 = \{C^1_0, C^1_1, ...C^1_{11}\}$.

Let $\oplus$ denote addition modulo 64 and $\ominus$ subtraction modulo 64.

$C_a^1 = \{<u_j, v_{j \oplus 2^a}> \mid a = 0,1,2,3,4,5;\ j = 0,1,...63\}$,

$C^1_{6+b} = \{<u_j, v_{j \ominus 2^b}> \mid b = 0,1,2,3,4,5;\ j = 0,1,...63\}$.

Then $K^1 = <C^1>$ describes the network.

Let $K^2 = <C^2>$ be the PM2I network with N = 64 as in Example 4.6.9.

What is the relationship between the networks.

Solution:

$C^1 = C^2$, and Theorem 4.6.8 imply $K^1 = K^2$.

Theorem 4.6.12:

Let there be $K^1 = <C^1>$ a PM2I network with $N = 2^m$ processors, then there exist $K^2 = <C^2>$ a generalized Illiac network in k dimensions, such that $K^1 \subseteq_c K^2$; moreover there exists $k = m$ dimension such that $K^1 = K^2$. Consequently PM2I can be viewed as a limiting case of a k-dimensional Illiac network.

Proof:

(1): Consider the generalized k-dimensional Illiac system with 64 processors, arranged as $2 \times 2 \times 2 \times 2 \times 2 \times 2$ matrix. This network can be modeled as follows.

$V_I^1 = \{u_j | j=0,1,...63\}$, $V_O^1 = \{v_k | k=0,1,...63\}$.

$C_a^1 = \{C^1_0, C^1_1, \cdots C^1_{11}\}$.

Let $\oplus$ denote addition modulo 64 and $\ominus$ subtraction modulo 64.

Let $d = N^{\frac{1}{k}}$,

$C_a^1 = \{<u_j, v_{j \oplus d^a}> |\ a = 0,1,...k-1;\ j = 0,1,...N-1\}$,

$C^1_{a+b} = \{<u_j, v_{j \ominus d^b}> |\ b = 0,1,...k = 1;\ j = 0,1,...N-1\}$.

Then $K^1 = <C^1>$ describes the network.

Consider the single stage PM2I network with N processors. The network can be described as follows.

$V_I^2 = \{u_j | j=0,1,...N-1\}$, $V_O^2 = \{v_k | k=0,1,...N-1\}$.

$C^2 = \{C^2_0, C^2_1, ...C^2_{2m-1}\}$.

$C_s^2 = \{<u_j, v_{j \oplus 2^s}> \mid s = 0,1,...2m-1;\ j = 0,1,...N-1\}.$

$C^2_{m+t} = \{<u_j, v_{j \ominus 2^t}>$

$\mid t = m, m+1,...2m-1;\ j = 0,1,...N-1\}.$

Then $K^2 = <C^2>$ describes the network.

(2): $N=2^m$ and $N = d^k \rightarrow 2^m = d^k \rightarrow d = 2^{\frac{m}{k}}$,

$\rightarrow \frac{m}{k} = 1,2,...m \rightarrow d = 2^{\frac{m}{k}} \rightarrow d = 2,4,...N.$

(3): (1), (2) $\rightarrow C_a^1 \subseteq C_s^2$.

$C_a^1 \subseteq C_s^2$ and $C_b^1 \subseteq C_t^2 \rightarrow \forall C_p^1 \in C^1 \quad \exists C_n^2 \in C^2 \ni$:

$C_p^1 = C_n^2$.

By Theorem 4.6.7 $K^1 \subseteq_c K^2$.

(4): (1), (2), and $d = 2 \rightarrow C_a^1 = C_s^2$.

$C_a^1 = C_s^2$ and $C_b^1 = C_t^2 \rightarrow C^1 = C^2$.

$C^1 = C^2$ and Theorem 4.6.8 imply $K^1 = K^2$.

4.7 Systems and Subsystems

In this section the problem of modeling systems and subsystems and analysis of different relationships between systems is being discussed. A system, informally, consists of a set of devices, an interconnection network, and a method of use of the network. A typical device can be a processor, memory, or a processor/memory pair. Each device has two logical ports, input and

output, possibly physically implemented as the same set of physical I/O pins.

Three types of systems are recognized based upon the method of use. Broadly speaking, a device can use the network in two basically different ways. A device can have its output connected to the input of the network and its input port connected to the output of the network. If this holds for all the devices in the system then this method will result in a recirculating system. From a communication point of view, these paths from the output of the network through the devices back to the input of the network can be used to generate different connection patterns using multiple passes through the network. Alternatively, a system could be constructed where there is a device such that a device where only the device's output is connected to the network, but its input is from outside of the system. If this holds for all the devices in the system than this configuration will result in a nonrecirculating system. This can be used to model systems such as a real time digital signal processing systems. A real time digital signal processing systems typically consist of several functional sets of (one or more) processors, each set optimized to perform a class of operations, together with an interconnection network between each pair of functional sets. From the communication point of view, these systems can not generate different connection patterns using multiple passes, because the paths from the outputs of the network to the network inputs through the devices are missing. Hybrid systems consisting of some (but not all) devices having return paths are also possible, for example a binary tree type networks, where the links are unidirectional and one of the leaves is connected to the root device.

Several relationships between two systems can hold, the systems can be completely different, they can be equal or they can have some degree of

similarity. The relationship of subsystems is discussed here. Informally, if there is a system over $V_I^1 \times V_O^1$, and another system over $V_I^2 \times V_O^2$, and $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$, then it is possible that the systems can be ordered in some sense. The ordering considered here is the subsystem relationship defined later. The idea is this. If both systems have the same method of use of the network, and additionally the network of one system contains some or all the states of the network of the other system, then the second system is in some sense a similar to the first. Using this concept, three different types of subsystems are defined and some examples presented.

Definition 4.7.1:

Let $K \in K[V_I \times V_O]$, $K = <C>$ be a network. If the usage of the network is such that data outputted at $v_y \in V_O$ can be fed back in $v_x \in V_I$, then $<v_x,v_y> \in C_F$. C_F is called *feedback correspondence.*

Physical implications: This describes the situation where a processor or any other device is connected to both v_y and v_x. The device inputs data into $v_x \in V_I$ and receives data at $v_y \in V_O$. Thus if $<v_x,v_y> \in C_F$ then the same device is attached to v_x and v_y. If $<v_x,v_y> \notin C_F$ then a separate device is attached to each of v_x and v_y. Since it is assumed that each device has only one input and one output, and that a vertex can have at most one device connected to it, the C_F has the following properties:

(a) if $<v_x,v_y>$, $<v_w,v_y> \in C_F$ then $v_x = v_w$;

(b) if $<v_x,v_y>$, $<v_x,v_z> \in C_F$ then $v_y = v_z$;

(c) $C_F \subset V_I \times V_O$.

Theorem 4.7.2:

C_F is a map, 1:1, onto from X to Y, where $X \subseteq V_I$ and $Y \subseteq V_O$.

Proof:

Obvious by definition of C_F and properties (a), (b), (c).

□

Definition 4.7.3:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle = \langle \{C_m\} \rangle$ be a network, and let C_F be a feedback correspondence, $C_F \subset V_I \times V_O$, then $S = \langle C, C_F \rangle = \langle \{C_m\}, C_F \rangle$ is called the *system over* $V_I \times V_O$.

Physical implications: The C_F precisely describes the usage of a network in a system. If $s(C_F) = V_I$ and $d(C_F) = V_O$, then the system is *fully recirculating.* If $C_F \neq \emptyset$ and either $s(C_F) \neq V_I$ or $d(C_F) \neq V_O$ (or both), then the system is *partially recirculating.* If $C_F = \emptyset$ then the system is *nonrecirculating.* An example of a system is given in Figure 4.2. The properties of C_F have implications on whether the system can generate different correspondences by using multiple passes through the network. Multiple passes require that $C_F \neq \emptyset$, that is the system must be partially or fully recirculating. At the end of this section examples of each type of the system are presented in detail.

Definition 4.7.4:

The set $\{S \mid S = \langle C, C_F \rangle$ is a system over $V_I \times V_O\}$ is called the *S-set over* $V_I \times V_O$, denoted by $S[V_I \times V_O]$.

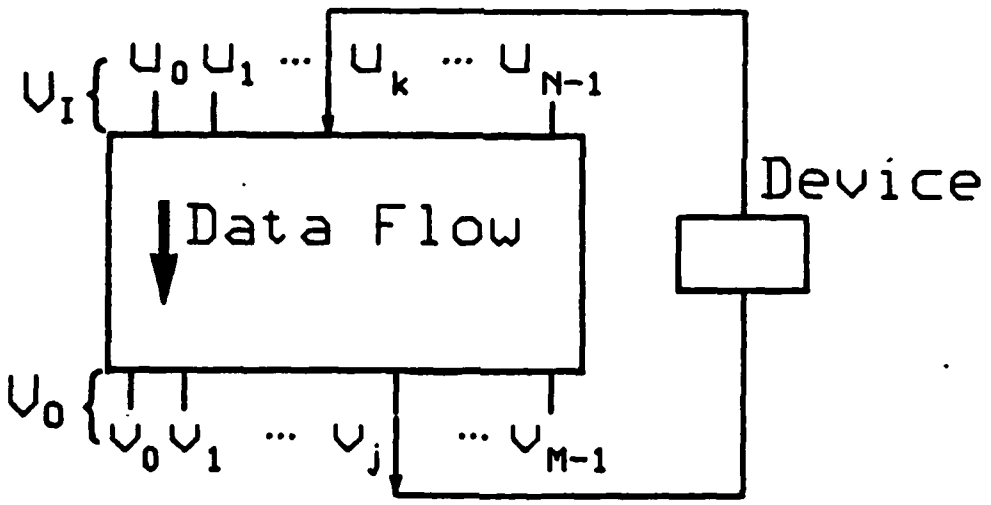


Figure 4.2:
Example of a system.

Definition 4.7.5:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems. If (1) $V_I^1 = V_I^2$, $V_O^1 = V_O^2$; (2) $C_F^1 = C_F^2$; and (3) $C^1 = C^2$, then S^1 *is equal to* S^2. *Notation:* $S^1 = S^2$.

Physical implication: S^1 and S^2 are completely interchangeable.

Theorem 4.7.6:

Sufficiency condition for equality of systems. If (3) holds in Definition 4.7.5 then (1) holds.

Proof:

(a): Show: (3) $\rightarrow V_I^1 = V_I^2$.

$V_I^1 = s(C^1) = s(C^2) = V_I^2$.

(b): Show: (3) $\rightarrow V_O^1 = V_O^2$.

$V_O^1 = d(C^1) = d(C^2) = V_O^2$.

□

The implication of this theorem is that to check two systems for equality it is only necessary to examine C_F and C.

In the following part, the definitions of different categories of relationship between two systems are formally given. Note that the similarity relationship here is an extensions of the relationship between networks (Section 4.6) that include the comparison of the feedback correspondences. To facilitate the understanding of the material, it is presented as follows. The categories of the similarity relationship are presented in the order of increasing strictness. Immediately after each formal definition, an example is presented.

Definition 4.7.7:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems. If

(1) $V_I^1 \subseteq V_I^2$ and $V_O^1 \subseteq V_O^2$;

(2) $C_F^1 = C_F^2 \mid_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(3) $\forall C_m^1 \in C^1 \;\; \exists C_n^2 \in C^2 \;\ni\; C_m^1 \subseteq C_n^2 \cup C_F^2$

then S^1 *is subsystem type a of* S^2. ("$\ni$" means "such that") *Notation:* $S^1 \subseteq_a S^2$.

Example of subsystem type *a*.

Let

$S^2 = <C^2, C_F^2>$ be a system.

$V_I^2 = \{v_0, v_1, v_2\}$,

$V_O^2 = \{u_0, u_1, u_2\}$,

$C_F^2 = \{<v_0, u_0>, <v_1, u_1>, <v_2, u_2>\}$,

$C^2 = \{C_0^2, C_1^2, C_2^2\}$,

$C_0^2 = \{<v_0, u_0>, <v_0, u_1>, <v_2, u_2>\}$,

$C_1^2 = \{<v_1, u_1>, <v_1, u_2>\}$,

$C_2^2 = \{<v_2, u_1>, <v_2, u_2>\}$.

Let

$V_I^1 = \{v_0, v_1\}$,

$V_O^1 = \{u_0, u_1\}$,

$C_F^1 = \{<v_0, u_0>, <v_1, u_1>\}$,

$C^1 = \{C_0^1, C_1^1\}$,

$C_0^1 = \{<v_0, u_0>, <v_1, u_1>\}$,

$C_1^1 = \{<v_0, u_1>\}$.

Then

(1) $<C^1, C_F^1>$ is a system (denoted S^1).

(2) (a) $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$

(b) $C_F^1 = C_F^2 \mid_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(c) $C_0^1 \subseteq C_F^2 \subseteq C_0^2 \quad \cup C_F^2$, $\quad C_1^1 \subseteq C_0^2 \subseteq C_0^2 \quad \cup C_F^2$

$\rightarrow S^1 \subseteq_a S^2$.

Definition 4.7.8:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems. If

(1) $V_I^1 \subseteq V_I^2, V_O^1 \subseteq V_O^2$;

(2) $C_F^1 = C_F^2 \mid_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(3) $\forall C_m^1 \in C^1 \quad \exists C_n^2 \in C^2 \ni C_m^1 \subseteq C_n^2$

then S^1 *is subsystem type b of* S^2. *Notation:* $S^1 \subseteq_b S^2$.

Example of subsystem type *b*.

Let

$S^2 = <C^2, C_F^2>$ be a system.

$V_I^2 = \{v_0, v_1, v_2\}$,

$V_O^2 = \{u_0, u_1, u_2\}$,

$C_F^2 = \{<v_0, u_0>, <v_1, u_1>, <v_2, u_2>\}$,

$C^2 = \{C_0^2, C_1^2, C_2^2\}$,

$C_0^2 = \{<v_0, u_0>, <v_0, u_1>, <v_2, u_2>\}$,

$C_1^2 = \{<v_1, u_1>, <v_1, u_2>\}$,

$C_2^2 = \{<v_2, u_1>, <v_2, u_2>\}$.

Let

$V_I^1 = \{v_0, v_1\}$,

$V_O^1 = \{u_0, u_1\}$,

$C_F^1 = \{<v_0, u_0>, <v_1, u_1>\}$,

$C^1 = \{C_0^1, C_1^1\}$,

$C_0^1 = \{<v_0, u_0>\}$,

$C_1^1 = \{<v_0, u_0>, <v_0, u_1>\}$.

Then

(1) $<C^1, C_F^1>$ is a system (denoted S^1).

(2) (a) $V_I^1 \subseteq V_I^2, V_O^1 \subseteq V_O^2$

(b) $C_F^1 = C_F^2|_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(c) $C_0^1 \subseteq C_0^2$, $C_1^1 \subseteq C_0^2 \rightarrow S^1 \subseteq b\ S^2$.

Definition 4.7.9:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems. If

(1) $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$;

(2) $C_F^1 = C_F^2|_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(3) $\forall C_m^1 \in C^1 \quad \exists\ C_n^2 \in C^2 \ni: C_m^1 = C_n^2$

then S^1 *is subsystem type c of* S^2. *Notation:* $S^1 \subseteq c\ S^2$.

Example of subsystem type *c*.

Let

$S^2 = <C^2, C_F^2>$ be a system.

$V_I^2 = \{v_0, v_1, v_2\}$,

$V_O^2 = \{u_0, u_1, u_2\}$,

$C_F^2 = \{<v_0, u_0>, <v_1, u_1>\}$,

$C^2 = \{C_0^2, C_1^2, C_2^2\}$,

$C_0^2 = \{<v_0, u_0>, <v_0, u_1>, <v_2, u_2>\}$,

$C_1^2 = \{<v_1, u_1>, <v_1, u_2>\}$,

$C_2^2 = \{<v_2, u_1>, <v_2, u_2>\}$.

Let

$V_I^1 = \{v_1, v_2\}$,

$V_O^1 = \{u_1, u_2\}$,

$C_F^1 = \{<v_1, u_1>\}$,

$C^1 = \{C_0^1, C_0^1\}$,

$C_0^1 = \{<v_1, u_1>, <v_1, u_2>\}$,

$C_1^1 = \{<v_2, u_1>, <v_2, u_2>\}$.

Then

(1) $<C^1, C_F^1>$ is a system (denoted S^1).

(2) (a) $V_I^1 \subseteq V_I^2$, $V_O^1 \subseteq V_O^2$

(b) $C_F^1 = C_F^2 |_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)}$;

(c) $C_0^1 = C_1^2$, $C_1^1 = C_2^2 \rightarrow S^1 \subseteq_c S^2$.

The Theorems 4.7.10 to 4.7.14 discuss the sufficiency conditions for the different relationships between systems to hold.

Theorem 4.7.10:

Sufficiency condition for subsystem type a.

If (2) and (3) hold in Definition 4.7.7 then (1) holds.

Proof:

(a): Show: (2), (3) $\rightarrow V_I^1 \subseteq V_I^2$.

$S^1 \subseteq_a S^2 \rightarrow \forall C_m^1 \in C^1 \quad \exists \; C_n^2 \in C^2 \ni C_m^1 \subseteq C_n^2 \cup C_F^2$.

$\rightarrow V_I^1 = s(C^1) = s(\bigcup_m C_m^1) \subseteq s(\bigcup_m (C_n^2 \cup C_F^2))$

$\rightarrow s(\bigcup_m (C_n^2 \cup C_F^2)) \subseteq s(\bigcup_n (C_n^2 \cup C_F^2)) \quad \forall n \ni C_n^2 \in C^2$

$\rightarrow s(\bigcup_n (C_n^2 \cup C_F^2)) = s(\bigcup_n C_n^2) \cup s(C_F^2) = V_I^2$

$\rightarrow V_I^1 \subseteq V_I^2$.

(b): Show: (2) and (3) $\rightarrow V_O^1 \subseteq V_O^2$.

Similar to (a), with $s(C^1)$ and $s(C^2)$ replaced by $d(C^1)$ and $d(C^2)$ respectively.

□

Theorem 4.7.11:

Sufficiency condition for subsystem type b.

If (3) holds in Definition 4.7.8 then (1) holds.

Proof:

Analogous to proof of Theorem 4.7.10 (note that (2) is not needed since C_F is not part of (3)).

□

Theorem 4.7.12:

Sufficiency condition for subsystem type c.

If (3) holds in Definition 4.7.9 then (1) holds.

Proof:

Analogous to proof of Theorem 4.7.11.

□

Theorem 4.7.13:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems.

(1) If $S^1 = S^2$ then $S^1 \subseteq_c S^2$.

(2) If $S^1 \subseteq_c S^2$ then $S^1 \subseteq_b S^2$.

(3) If $S^1 \subseteq_b S^2$ then $S^1 \subseteq_a S^2$.

Proof:

Obvious, follows from definitions of subsystems.

□

Theorem 4.7.14:

Let $S^1 = <C^1, C_F^1>$ and $S^2 = <C^2, C_F^2>$ be two systems. If

(1) $S^1 \subseteq_c S^2$ and

(2) $S^2 \subseteq_c S^1$, then $S^1 = S^2$.

Proof:

Show:

(1): $V^1_I = V^2_I$, $V^1_O = V^2_O$;

(2): $C_F^1 = C_F^2$; and

(3): $C^1 = C^2$.

(a): Show: $V^1_I = V^2_I$, $V^1_O = V^2_O$.

From Theorem 4.7.12 it is known that (3) $\rightarrow$ (1), so only (2) and (3) have to be shown.

(b): Show: $C_F^1 = C_F^2$.

$S^1 \subseteq_c S^2 \rightarrow C_F^1 \subseteq C_F^2$, $S^2 \subseteq_c S^1 \rightarrow C_F^2 \subseteq C_F^1$

$\rightarrow C_F^1 = C_F^2$.

(c): Show: $C^1 = C^2$.

$\forall C_m^1 \in C^1 \ \exists$ unique $C_n^2 \in C^2 \ni C_m^1 = C_n^2$.

Similarly $\forall C_q^2 \in C^2 \ \exists$ unique $C_p^1 \in C^1 \ni C_q^2 = C_p^1$

$\rightarrow C^1 = C^2$.

(d): $C^1_F = C^2_F$, $C^1 = C^2 \rightarrow <C^1, C^1_F> = <C^2, C^2_F>$.

□

In the following part, detail examples of the three types of systems: fully recirculating, partially recirculating, and nonrecirculating are given.

Example 4.7.15:

Consider the following system:

$V_I = \{u_0, u_1, u_2, u_3\}$, $V_O = \{v_a, v_b, v_c, v_d\}$,

$C_1 = \{<u_0, v_a>, <u_1, v_b>, <u_2, v_c>, <u_3, v_d>\}$,

$C_2 = \{<u_0, v_c>, <u_1, v_d>, <u_2, v_a>, <u_3, v_b>\}$,

$C_F = \{<u_0, v_d>, <u_1, v_a>, <u_2, v_b>, <u_3, v_c>\}$,

$S \in S[\{u_0, u_1, u_2, u_3\} \times \{v_a, v_b, v_c, v_d\}]$,

$S = <C_1, C_F> = <\{C_1, C_2\}, C_F>$.

Find the type of the system.

Solution:

Based on the C_F, the system is a fully recirculating system. In particular, the system is isomorphic to a bidirectional ring $S = <\{R_{+1}, R_{-1}\}$, identity map $>$. See Figure 4.3.

Example 4.7.16:

Consider the following system:

$V_I = \{u_0, u_1, u_2, u_3\}$, $V_O = \{v_a, v_b, v_c, v_d\}$,

$C_1 = \{<u_0, v_a>, <u_1, v_b>, <u_2, v_c>, <u_3, v_d>\}$,

$C_2 = \{<u_0, v_c>, <u_1, v_d>, <u_2, v_a>, <u_3, v_b>\}$,

$C_F = \{<u_1, v_a>, <u_2, v_b>, <u_3, v_c>\}$,

$S \in S[\{u_0, u_1, u_2, u_3\} \times \{v_a, v_b, v_c, v_d\}]$,

$S = <C_1, C_F> = <\{C_1, C_2\}, C_F>$..

Find the type of the system.

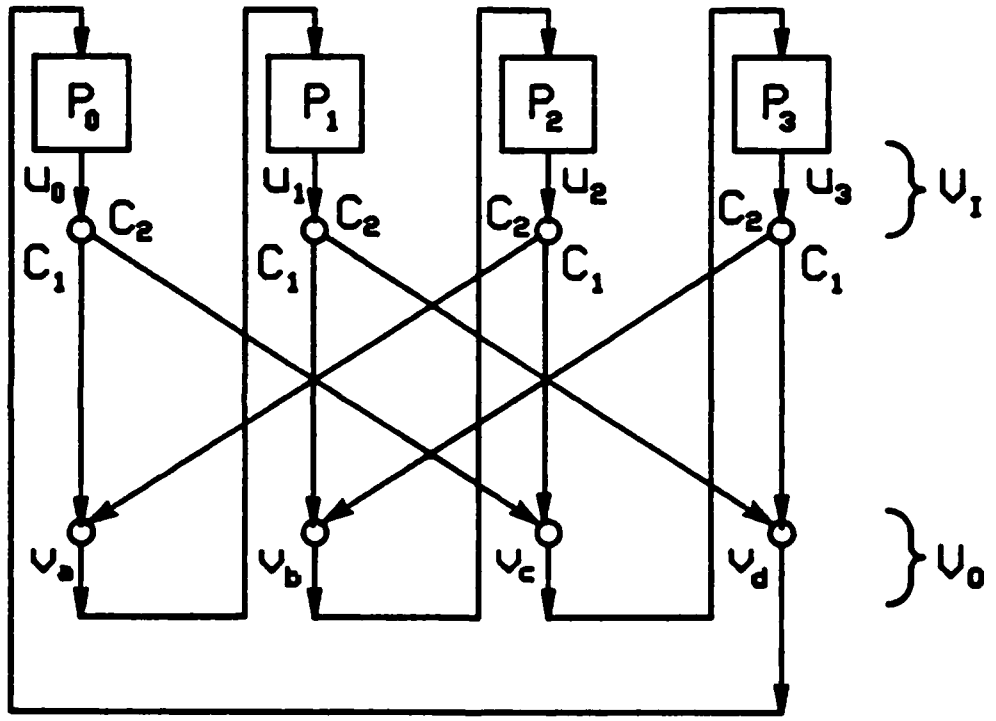


Figure 4.3:
Example of a fully recirculating system.

Solution:

Based on the C_F, the system is a partially recirculating system. In particular, the system is isomorphic to a reconfigurable pipeline with C_1 for algorithm 1 and C_2 for algorithm 2. See Figure 4.4.

Example 4.7.17:

Consider the following system:

$V_I = \{u_0, u_1, u_2, u_3\}$, $V_O = \{v_a, v_b, v_c, v_d\}$,

$C_1 = \{<u_0, v_a>, <u_1, v_b>, <u_2, v_c>, <u_3, v_d>\}$,

$C_2 = \{<u_0, v_c>, <u_1, v_d>, <u_2, v_a>, <u_3, v_b>\}$,

$C_F = \emptyset$,

$S \in S\ [\{u_0, u_1, u_2, u_3\} \times \{v_a, v_b, v_c, v_d\}]$,

$S = <C_1, C_F> = <\{C_1, C_2\}, C_F>$.

Find the type of the system.

Solution:

Based on the C_F, the system is a nonrecirculating system. In particular, the system is isomorphic to a distributed signal processing system. See Figure 4.5.

Example 4.7.18:

Consider a system with three processor/memory pairs, where each processor has a single physical port. The processors communicate via a shared bus. The physical port can reconfigured as *either* a logical input or a logical output port.

Construct a model of this system.

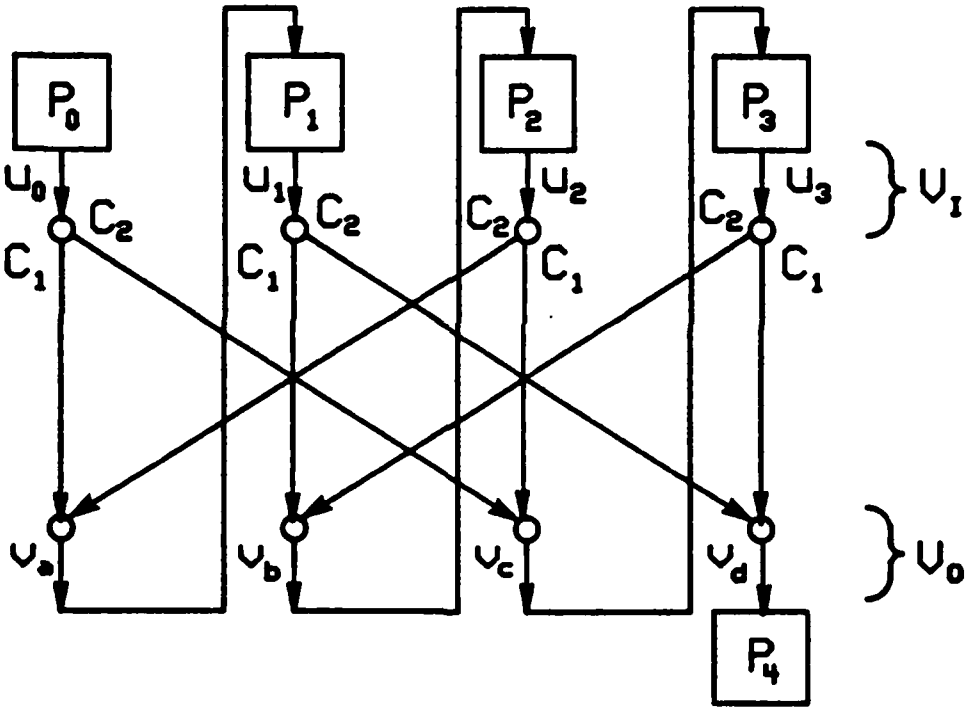


Figure 4.4:
Example of a partially recirculating system.

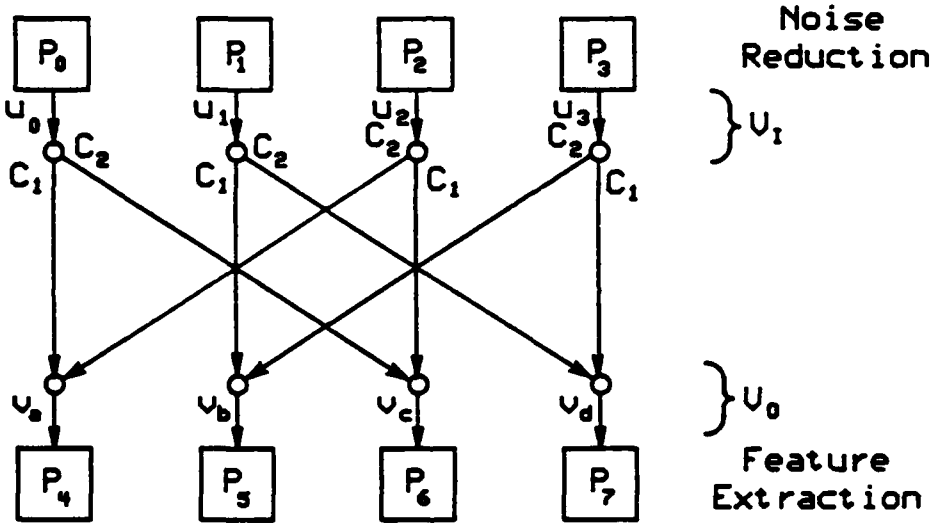


Figure 4.5:
Example of a nonrecirculating system.

Solution:

(1): Denote the input and output label sets by $V_I = \{u_0,u_1,u_2\}$ and $V_O = \{v_0,v_1,v_2\}$. Each device d_i has its output port connected to the input label u_i and has its input port connected to the output label v_i, i=0,1,2.

(2): Based upon the given information, the feedback correspondence is $C_F = \{<u_i,v_i> \mid i=0,1,2\}$.

(3): The states of the network are as follows.

$A(k) = \{<u_k,v_j>, \mid j,k = 0,1,2;\ k \neq j\}$.

Let $A = \{A(k) \mid k = 0,1,2\}$, (set of all 1:1 connections).

$B_{i,j}(k) = \{<u_k,v_i>,<u_k,v_j> \mid i,j,k = 0,1,2;\ k \neq i;\ k \neq j;\ i \neq j\}$.

Let $B = \{B_{i,j}(k) \mid k = 0,1,2\}$, (set of all two way broadcasts).

(4): The model then will be as follows.

$C = A \cup B$, $C_F = \{<u_i,v_i> \mid i=0,1,2\}$, and $S = <C,C_F>$.

Example 4.7.19:

Consider a system consisting of the Illiac network with N = 64 processors as in Example 4.6.9. The network is used in a fully recirculating system. The system can be modeled as follows.

$V_I^1 = \{u_j \mid j=0,1,...63\}$, $V_O^1 = \{v_k \mid k=0,1,...63\}$.

$C^1 = \{C_0^1, C_1^1, C_2^1, C_3^1\}$.

Let $\oplus$ denote addition modulo 64 and $\ominus$ subtraction modulo 64.

$C_0^1 = \{<u_j,v_{j\oplus 1}> \mid j = 0,1,...63\}$,

$C_1^1 = \{<u_j,v_{j\ominus 1}> \mid j = 0,1,...63\}$,

$C_2^1 = \{<u_j,v_{j\oplus 8}> \mid j = 0,1,...63\}$,

$C_3^1 = \{<u_j,v_{j\ominus 8}> \mid j = 0,1,...63\}$.

$C_F = \{<u_i,v_i> \mid i = 0,1,...N-1\}$.

Then $S^1 = <C^1, C^1_F>$ describes the system.

Consider the single stage PM2I network with N = 64 processors as in Example 4.6.9. The network is used in a nonrecirculating system. The system can be described as follows.

$V_I^2 = \{u_j \mid j=0,1,...63\}$, $V_O^2 = \{v_k \mid k=0,1,...63\}$.

$C^2 = \{C_0, C_1, ...C_{11}\}$,

$C^2_s = \{<u_j, v_{j \oplus 2^s}> \mid s = 0,1,...5;\ j = 0,1,...63\}$.

$C^2_{6+t} = \{<u_j, v_{j \ominus 2^t}> \mid t = 0,1,...5;\ j = 0,1,...63\}$.

$C^2_F = \emptyset$.

Then $S^2 = <C^2, C^2_F>$ describes the system.

What is the relationship between the two systems.

Solution:

From Example 4.6.9 it was found that the Illiac is a subnetwork of type c of PM2I, but it would not be correct to conclude that $S^1 \subset_c S^2$ since $(C_F^1 \neq C_F^2 \mid_{(V_I^1 \times V_O^2) \cup (V_I^2 \times V_O^1)})$. For example, the S^1 is capable of executing the interconnection function $A_3 = \{<u_i, v_{i \oplus 3}> \mid i = 0,1,...N-1\}$ using multiple passes through the network. S^2 is not capable of executing the function A_3 because it is a nonrecirculating system and therefore not capable of multiple passes through the network.

It is important therefore, to consider the feedback connections when evaluating the relationships between systems. Therefore one must conclude, that the comparison between systems is not possible and does not make sense if the systems use their respective networks differently.

4.8 Conclusions

In this chapter the following problems were addressed. A general model, implementation independent, for modeling of topologically arbitrary interconnection networks was developed. Several important relationships between networks were rigidly formulated such as equality and subnetworks. A similarity relationship between networks was defined. The relationship has the following categories in the order of increasing strictness: (a) networks are equal, (b) K^1 is subnetwork of type c of K^2, (c) K^1 is subnetwork of type b of K^2, and (d) none of the above. Note that this is an extension of the previously used method which categorizes networks into two classes only: isomorphic and nonisomorphic.

A system and different types of subsystems were defined. A system informally, consists of a set of devices, an interconnection network and the method of use of the network by the devices. Three different types of systems were defined, based upon the method of use of the network and several relationships between two systems were analyzed. The systems types recognized are recirculating, nonrecirculating, and partially recirculating. In a recirculating system each device d_i has its logical output port connected to an input label of the interconnection network and its input port connected to an output label of the interconnection network. For a fully recirculating system $|V_I| = |V_O|$. A partially recirculating system contains some, but not all, devices each of which has its output port connected to the network input label and its input port connected to an output label of the network. If $|V_I| \neq |V_O|$, than the system cannot be recirculating and can be only

partially recirculating or nonrecirculating, because each device has only one input port and one output port. In a fully or partially recirculating it is possible to generate different connection patterns using multiple passes through the network. In a nonrecirculating system, each device is connected only to the network input or (exclusive) to a network output. This type of configuration appears frequently in real time digital signal processing systems. The result of this configuration is that no new connection patterns can be achieved by multiple passes, since it is not possible to move the data from the output of the network back to its input.

A similarity relationship between systems was defined. It is an extension of classification of networks which takes into the consideration the C_F properties. The relationship has the following categories in the order of increasing strictness: (a) systems are equal, (b) S^1 is subsystem of type c of S^2, (c) S^1 is subsystem of type b of S^2, (d) S^1 is subsystem of type a of S^2, and (e) none of the above. Note that this is an extension of the previously used method which categorizes systems into two classes only: isomorphic and nonisomorphic.

5 QUASIMORPHISM AND EMULATION

5.1 Introduction

In Chapter 4 a restricted problem of a measure of similarity between two systems was studied. It was assumed that given a system S^1 over $V_I^1 \times V_O^1$ and S^2 over $V_I^2 \times V_O^2$, the labeling is such that $V_I^1 \subseteq V_I^2$ and $V_O^1 \subseteq V_O^2$. If above does not hold that does not mean that the two systems are dissimilar, it just could mean that the V_I and V_O labeling is not helpful.

To study the problem of comparison of randomly labeled systems the concept of quasimorphism of systems was developed [SeS84a]. This concept is related to the classification of groups in the field of abstract algebra and group theory. The theory of group classification is based upon the concept of morphism. Morphism is a measure of similarity of behaviors of group operations of two groups. This measure ignores the labeling of the elements of the groups and is concerned strictly with the structure which is determined by the group operation.

In the domain of parallel computer systems the structure of interest is the structure of the correspondences of the system's network in the graph theoretical sense. The quasimorphism of systems allows a method of comparison of randomly labeled, topologically arbitrary parallel computer systems. The quasimorphism facilitates the analysis of following problems in parallel processing:

(a) system A emulating system B (three different degrees of strictness of emulation are discussed);

(b) fault tolerance/reliability (achieved by multiple mapping of same problem into a system);

(c) partitioning of a system.

The quasimorphism is analyzed with respect to properties similar to the properties of reflexivity, symmetry, and transitivity. Several examples of quasimorphism of different types are presented.

Also in this chapter the problem of emulation of one system by another is discussed. Three different types of emulation are considered. Several measures of efficiency of emulation are defined and the three types of emulation are evaluated using these criteria. Two examples of emulation of arbitrary systems are presented.

5.2 Overview

In Section 5.3 the problems discussed in this chapter are defined. In Section 5.4, the previous related work is described. In Section 5.5 the basic definitions and concepts are given. In Section 5.6 the measure of similarity of systems called quasimorphism is developed. In Section 5.7 an application of quasimorphism in emulation of one system by another is researched. In Section 5.8 the conclusions of this chapter are given.

5.3 Problem Statement

It is intuitively obvious that some systems have different topologies than others, yet not much has been done in the past research to quantify the differences. In the past, researchers used only two wide categories, two networks are isomorphic or two networks are not isomorphic. In this chapter a refinment of the measure of similarity between two systems is explored. In the domain of parallel computer systems the structure of interest is the structure of the correspondences of the system's network in the graph theoretical sense. The dynamic behavior of the system's reconfigurable network which generate a set of correspondences as a function of the control strategy, must be taken into the consideration. Based upon the idea of morphism of groups, the concept of morphism of parallel computer system topology is developed. This measure is called quasimorphism and is based upon the concept of morphism of groups in group theory. It allows a comparison of topologically arbitrary parallel computer systems. The measure is used in the analysis of emulation of one system by another. Three different types emulation are defined and their properties are explored.

5.4 Previous Work

The following are the three research areas related to the topics explored in this chapter. The simple (two class only) similarity measure between two networks was used to show that multi-stage Shuffle-Exchange network is isomorphic to the n-Cube [Law75]. The emulation definition here is a generalized form of the definition used in [FiF82] to study quotient networks. Another work, related to our research developed here, can be found in the classification of groups in the field of the abstract algebra and group theory [Han68, Her75]. The theory of group classification is based upon the concept of morphism. Morphism is a measure of similarity between behaviors of two groups. This measure ignores the labeling of the elements of the groups and is concerned strictly with the structure, which is determined by the group operation.

5.5 Basic Concepts

The analysis of relationships between systems can be described mathematically as finding correspondences between two sets of systems, or S-sets. This problem is very complex to handle directly and therefore it will be broken into two major parts. Note that each system is defined over an underlying $V_I \times V_O$. A structure called T-element will be defined over the

same underlying $V_I \times V_O$ set. The T-element has less constraints than a system and therefore is easier analyze.

The first major part, presented in this section, will then be the analysis of relationships between T-elements. The major part will in turn be broken into finding relationships between the underlying substructures of the T-elements. The set of all T-elements over particular $V_I \times V_O$ is called the T-set.

The second major part, presented in Section 5.6, will consist of analysis of relationships between two S-sets. Section 5.6 will use the relationships derived in this section to discuss the relationships between S-sets. As intended some relationships between two T-sets will be directly applicable to the relationships between two S-sets, and some others will be applicable in somewhat weakened form.

To resolve the ambiguity in the notation $\{<u_a,u_b>\}$, assume that it will indicate a set of pairs unless specifically described as a singleton. Definition 5.5.1 identifies the universe of discourse for this section.

Definition 5.5.1:

Let $T[V_I \times V_O] \triangleq \{<\{E_m \mid m=1,2,...n\}, E_F> \mid E_m \in P[V_I \times V_O], E_F \in P[V_I \times V_O]\}$. Then $T[V_I \times V_O]$ is called the *T-set over* $V_I \times V_O$.

The maps ϕ_I and ϕ_O are the basic elements in the discussion of the relationship between two T-elements. Since the analysis is very complex, some auxiliary intermediate maps and correspondences are defined in Definitions 5.5.3 to 5.5.5. For a pictorial representation of the genealogy of these maps and correspondences see Figure 5.1.

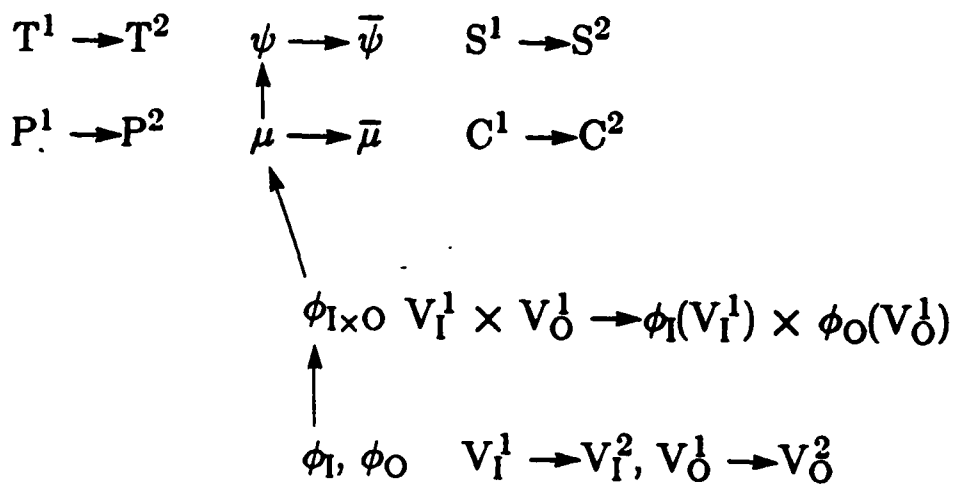


Figure 5.1:
Genealogy of the maps and correspondences.

Definition 5.5.2:

Define ϕ_I *-map* ϕ_I and ϕ_O *-map* ϕ_O as follows:

$\phi_I : V_I^1 \rightarrow \phi_I(V_I^1)$, map; and $\phi_O : V_O^1 \rightarrow \phi_O(V_O^1)$, map.

Definition 5.5.3:

Let $\phi_I : V_I^1 \rightarrow \phi_I(V_I^1)$, be a map and $\phi_O : V_O^1 \rightarrow \phi_O(V_O^1)$, be a map. Define a $\phi_{I\times O}$*-map from* $V_I \times V_O$ *to* $\phi_I(V_I^1) \times \phi_O(V_O^1)$ to be any map such that:

$\phi_{I\times O} : V_I^1 \times V_O^1 \rightarrow \phi_I(V_I^1) \times \phi_O(V_O^1)$, $\forall <v_a,v_b> \in V_I^1 \times V_O^1$,

$\phi_{I\times O}(<v_a,v_b>) \triangleq <\phi_I(v_a), \phi_O(v_b)>$.

Note that $\phi_{I\times O}$ is generated by ϕ_I and ϕ_O, which given $\phi_{I\times O}$ are clearly unique by definition. Clearly the $\phi_{I\times O}$ is an onto map.

Definition 5.5.4:

Let $\phi_{I\times O}$ be a $\phi_{I\times O}$-map from $V_I^1 \times V_O^1$ to $\phi_I(V_I^1) \times \phi_O(V_O^1)$. Define a μ*-map from* $P[V_I^1 \times V_O^1]$ *to* $P[\phi_I(V_I^1) \times \phi_O(V_O^1)]$ to be any map such that:

$\mu : P[V_I^1 \times V_O^1] \rightarrow P[\phi_I(V_I^1) \times \phi_O(V_O^1)]$,

$\forall E = \{<v_a,v_b>\} \in P[V_I^1 \times V_O^1]$,

$\mu(E) = \mu(\{<v_a,v_b>\}) \triangleq \{\phi_{I\times O}(<v_a,v_b>)\}$.

Note that μ is generated by $\phi_{I\times O}$. Clearly the μ is an onto map.

Definition 5.5.5:

Let $T^1 = T[V_I^1 \times V_O^1]$ and $T^2 = T[\phi_I(V_I^1) \times \phi_O(V_O^1)]$ be two T-sets. Let μ be a μ-map from $P[V_I^1 \times V_O^1]$ to $P[\phi_I(V_I^1) \times \phi_O(V_O^1)]$. Define a ψ*-map from* T^1 *to* T^2 to be any map such that:

$\psi : T^1 \rightarrow T^2$, $\forall <\{E_m \mid m=1,2,...n\}, E_F> \in T^1$,

$\psi(<\{E_m \mid m=1,2,...n\}, E_F>) \triangleq <\{\mu(E_m) \mid m=1,2,...n\}, \mu(E_F)>$.

Note that ψ is generated by μ. Clearly the ψ is an onto map.

Unless otherwise noted for the rest of this chapter this notation will be used:

$P^1 = P[V_I^1 \times V_O^1]$, $P^2 = P[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

$C^1 = C[V_I^1 \times V_O^1]$, $C^2 = C[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

$K^1 = K[V_I^1 \times V_O^1]$, $K^2 = K[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

$S^1 = S[V_I^1 \times V_O^1]$, $S^2 = S[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

$T^1 = T[V_I^1 \times V_O^1]$, $T^2 = T[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

The Lemmas and Theorems 5.5.6 to 5.5.10 discuss the heritage of some properties between the auxiliary maps and correspondences. Alone these results are not of practical importance, however, the results will be used in Section 5.7 to discuss the properties of quasimorphism, which is the main goal of these two sections.

Lemma 5.5.6:

Let ϕ_I and ϕ_O be the ϕ_I-map and ϕ_O-map generating $\phi_{I\times O}$.

ϕ_I and ϕ_O are 1:1 maps iff $\phi_{I\times O}$ is 1:1 map.

Proof:

Let $\phi_{I\times O} : V_I^1 \times V_O^1 \rightarrow \phi_I(V_I^1) \times \phi_O(V_O^1)$ be the $\phi_{I\times O}$-map.

Case 1: Show : $\forall <u_a,u_b>, <v_a,v_b> \in V_I^1 \times V_O^1$,

$\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>) \rightarrow <u_a,u_b> = <v_a,v_b>$.

(1): $\phi_{I\times O}(<u_a,u_b>) = <\phi_I(u_a), \phi_O(u_b)>$, $\phi_{I\times O}(<v_a,v_b>) = <\phi_I(v_a),\phi_O(v_b)>$, and $\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>)$ $\rightarrow <\phi_I(u_a),\phi_O(u_b)> = <\phi_I(v_a),\phi_O(v_b)> \rightarrow \phi_I(u_a) = \phi_I(v_b)$, $\phi_O(u_b) = \phi_O(v_b)$.

(2): (1) and ϕ_I,ϕ_O 1:1 $\rightarrow u_a = v_a$, $u_b = v_b$ $\rightarrow <u_a,u_b> = <v_a,v_b>$.

Case 2: Show: $\forall\, u_a, v_a \in V_I,\ \phi_I(u_a) = \phi_I(v_a) \rightarrow u_a = v_a$;

and $\forall\, u_b, v_b \in V_O,\ \phi_O(u_b) = \phi_O(v_b) \rightarrow u_b = v_b$.

(1): $\phi_I(u_a) = \phi_I(v_a),\ \phi_O(u_b) = \phi_O(v_b) \rightarrow <\phi_I(u_a),\phi_O(u_b)> = <\phi_I(v_a),\phi_O(v_b)> \rightarrow \phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>)$.

(2): $\phi_{I\times O}$ 1:1 and (1) $\rightarrow <u_a,u_b> = <v_a,v_b>$

$\rightarrow u_a = v_a,\ u_b = v_b$.

□

Lemma 5.5.7:

Let $\phi_{I\times O}$ be the $\phi_{I\times O}$-map generating μ.

$\phi_{I\times O}$ is 1:1 map iff μ is 1:1 map.

Proof:

Let $\mu : P^1 \rightarrow P^2$ be the μ-map.

Case 1: Show: $\forall\, E_m, E_n \in P^1,\ \mu(E_m) = \mu(E_n) \rightarrow E_m = E_n$.

(1): Let $E_m = \{<u_a,u_b>\}$ and $E_n = \{<v_a,v_b>\}$.

$\mu(E_m) = \mu(\{<u_a,u_b>\}) = \{\phi_{I\times O}(<u_a,u_b>)\}$,

$\mu(E_n) = \mu(\{<v_a,v_b>\}) = \{\phi_{I\times O}(<v_a,v_b>)\}$ and $\mu(E_m) = \mu(E_n)$

$\rightarrow \{\phi_{I\times O}(<u_a,u_b>)\} = \{\phi_{I\times O}(<v_a,v_b>)\}$.

(2): Show $E_m \subseteq E_n$.

(2a): $<u'_a,u'_b> \in E_m \rightarrow \phi_{I\times O}(<u'_a,u'_b>) \in \mu(E_m)$

$\rightarrow \phi_{I\times O}(<u'_a,u'_b>) \in \{\phi_{I\times O}(<u_a,u_b>)\}$.

(2b): $\mu(E_m) = \mu(E_n)$ and (2a) $\rightarrow \phi_{I\times O}(<u'_a,u'_b>) \in \mu(E_n)$

$\rightarrow \exists\ <v'_a,v'_b> \in E_n \ni : \phi_{I\times O}(<u'_a,u'_b>) = \phi_{I\times O}(<v'_a,v'_b>)$.

(2c): (2b) and $\phi_{I\times O}$ 1:1 $\rightarrow$ $<u_a',u_b'> = <v_a',v_b'>$ $\rightarrow$ $<u_a',u_b'> \in E_n$ $\rightarrow$ $E_m \subseteq E_n$.

(3): Similarly $E_n \subseteq E_m$.

(4): (2c), (3) $\rightarrow E_m = E_n$.

Case 2: Show: $\forall <u_a,u_b>, <v_a,v_b> \in V_I^1 \times V_O^1$, $\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>) \rightarrow <u_a,u_b> = <v_a,v_b>$.

(1): Let $E_m,E_n \in P^1$, $E_m = \{<u_a,u_b> \mid \text{singleton}\}$, $E_n = \{<v_a,v_b> \mid \text{singleton}\}$. $\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>) \rightarrow \{\phi_{I\times O}(<u_a,u_b>) \mid \text{singleton}\} = \{\phi_{I\times O}(<v_a,v_b>) \mid \text{singleton}\} \rightarrow \mu(E_m) = \mu(E_n)$.

(2): μ 1:1 and (1) $\rightarrow E_m = E_n \rightarrow \{<u_a,u_b> \mid \text{singleton}\} = \{<v_a,v_b> \mid \text{singleton}\} \rightarrow <u_a,u_b> = <v_a,v_b>$.

□

Lemma 5.5.8:

Let μ be the μ-map generating ψ.

μ is 1:1 map iff ψ is 1:1 map.

Proof:

Let $\psi : T^1 \rightarrow T^2$ be the ψ-map.

Case 1: Show: $\forall T^{1,i}, T^{1,j} \in T^1$, $\psi(T^{1,i}) = \psi(T^{1,j}) \rightarrow T^{1,i} = T^{1,j}$.

(1): Let $T^{1,i} = <\{E_m^{1,i} \mid m=1,2,...p\}, E_F^{1,i}>$

and $T^{1,j} = <\{E_n^{1,j} \mid n=1,2,...q\}, E_F^{1,j}>$.

$\psi(T^{1,i}) = \psi(<\{E_m^{1,i} \mid m=1,2,...p\}, E_F^{1,i}>) =$

$<\{\mu(E_m^{1,i}) \mid m=1,2,...p\}, \mu(E_F^{1,i})>$,

$\psi(T^{1,j}) = \psi(<\{E_n^{1,j} \mid n=1,2,...q\}, E_F^{1,j}>) =$

$<\{\mu(E_n^{1,j}) \mid n=1,2,...q\}, \mu(E_F^{1,j})>$ and

$\psi(T^{1,i}) = \psi(T^{1,j}) \rightarrow <\{\mu(E_m^{1,i}) \mid m=1,2,...p\}, \mu(E_F^{1,i})>$

$= <\{\mu(E_n^{1,j}) \mid n=1,2,...q\}, \mu(E_F^{1,j})>$.

(2): (1) $\rightarrow \mu(E_F^{1,i}) = \mu(E_F^{1,j})$.

(2a): (2) and μ 1:1 $\rightarrow E_F^{1,i} = E_F^{1,j}$.

(3): Show $\{E_m^{1,i} \mid m=1,2,...p\} \subseteq \{E_n^{1,j} \mid n=1,2,...q\}$.

(3a): $E_m^{1,i} \in \{E_m^{1,i} \mid m=1,2,...p\} \rightarrow \mu(E_m^{1,i}) \in \{\mu(E_m^{1,i}) \mid m=1,2,...p\}$.

(3b): (3a), (1) $\rightarrow \mu(E_m^{1,i}) \in \{\mu(E_n^{1,j}) \mid n=1,2,...q\}$

$\rightarrow \exists\, E_o^{1,j} \in \{E_n^{1,j} \mid n=1,2,...q\} \ni : \mu(E_m^{1,i}) = \mu(E_o^{1,j})$.

(3c): (3b) and μ 1:1 $\rightarrow E_m^{1,i} = E_o^{1,j} \rightarrow E_m^{1,i} \in \{E_n^{1,j} \mid n=1,2,...q\}$

$\rightarrow \{E_m^{1,i} \mid m=1,2,...p\} \subseteq \{E_n^{1,j} \mid n=1,2,...q\}$.

(4): Similarly $\{E_n^{1,j} \mid n=1,2,...q\} \subseteq \{E_m^{1,i} \mid m=1,2,...p\}$.

(5): (3c), (4) $\rightarrow \{E_m^{1,i} \mid m=1,2,...p\} = \{E_n^{1,j} \mid n=1,2,...q\}$.

(6): (1), (5) $\rightarrow <\{E_m^{1,i} \mid m=1,2,...p\}, E_F^{1,i}> =$

$<\{E_n^{1,j} \mid n=1,2,...q\}, E_F^{1,j}> \rightarrow T^{1,i} = T^{1,j}$.

Case 2: Show: $\forall E_a, E_b \in P^1$, $\mu(E_a) = \mu(E_b) \rightarrow E_a = E_b$.

(1): Consider $\{E_m^{1,i} \mid m=1,2,...p\}$ and $\{E_n^{1,j} \mid n=1,2,...q\}$

$\ni : \{E_m^{1,i} \mid m=1,2,...p\} = \{E_n^{1,j} \mid n=1,2,...q\}$ and $E_m^{1,i}, E_n^{1,j} \in P^1$.

Consider $E_a, E_b \in P^1 \ni : \mu(E_a) = \mu(E_b)$. Then

$<\{E_m^{1,i} \mid m=1,2,...p\}, E_a> \in T[V_I^1 \times V_O^1]$ and

$<\{E_n^{1,j} \mid n=1,2,...q\}, E_b> \in T[V_I^1 \times V_O^1]$, denoted $T^{1,i}$ and $T^{1,j}$ respectively.

(2): $<\{\mu(E_m^{1,i})\}, \mu(E_a)> = \psi(T^{1,i})$, $<\{\mu(E_n^{1,j})\}, \mu(E_b)> = \psi(T^{1,j})$ and

(1) $\rightarrow \{\mu(E_m^{1,i}) \mid m=1,2,...p\} = \{\mu(E_n^{1,j}) \mid n=1,2,...q\}$.

(3): (2) and $\mu(E_a) = \mu(E_b) \rightarrow \psi(T^{1,i}) = \psi(T^{1,j})$.

(4): (3) and ψ 1:1 $\rightarrow T^{1,i} = T^{1,j} \rightarrow E_a = E_b$.

□

Theorem 5.5.9:

Let ϕ_I and ϕ_O be the ϕ_I-map and ϕ_O-map generating $\psi : T^1 \rightarrow T^2$.

ϕ_I and ϕ_O are 1:1 maps iff ψ is 1:1 map.

Proof:

Follow directly from Lemmas 5.5.6, 5.5.7, and 5.5.8.

□

Theorem 5.5.10:

Let $\psi : T^1 \rightarrow T^2$ be a ψ-map generated by μ.

μ is 1:1 map iff T^1 and T^2 are isomorphic T-sets.

Proof:

Let $\mu : P^1 \rightarrow P^2$.

Case 1: Show ψ is 1:1 and onto and ψ is morphism.

(1): Show ψ is 1:1.

(1.1): Lemma 5.5.8 and μ 1:1 $\rightarrow \psi$ is 1:1.

(2): Show ψ is onto.

(2.1): ψ clearly onto.

(3): Show ψ is morphism.

(3.1): By definition of ψ it is a morphism.

Case 2: Show μ is 1:1 and onto.

(1): Show μ is 1:1.

(1.1): Lemma 5.5.8 and ψ 1:1 $\rightarrow$ μ is 1:1.

(2): Show μ is onto.

(2.1): μ clearly onto.

□

The following two examples and Theorem 5.5.13 illustrate some properties of the T-sets.

Example 5.5.11:

Consider the structure with the input and output label sets $V_I = \{u_0\}$, $V_O = \{v_0, v_1\}$.

Describe the T-set over $V_I \times V_O$.

Solution:

(1): $T[V_I \times V_O] = \{<E, E_F>\} =$
$\{<\{E_m | m = 1,2,...n\}, E_F> | E_m \in P[V_I \times V_O], E_F \in P[V_I \times V_O]\}$.

(2): $|V_I \times V_O| = |V_I| \times |V_O| = 2$.

(3): $|P[V_I \times V_O]| = 2^2 = 4$.

(4): The number of different E is equal to the number of subsets of $P[V_I \times V_O] = 2^4$.

(5): The number of different E_F is $|P[V_I \times V_O]|$.

(6): $|T[V_I \times V_O]| = 2^4 \times 4 = 2^6 = 64$.

Example 5.5.12:

Consider the structure with the input and output label sets $V_I = \{u_0,u_1\}$, $V_O = \{v_0,v_1,v_2\}$.

Describe the T-set over $V_I \times V_O$.

Solution:

(1): $T[V_I \times V_O] = \{<E, E_F>\} = \{<\{E_m | \, m = 1,2,...n\}, E_F> | \, E_m \in P[V_I \times V_O], \; E_F \in P[V_I \times V_O]\}$.

(2): $|V_I \times V_O| = |V_I| \times |V_O| = 6$.

(3): $|P[V_I \times V_O]| = 2^6 = 64$.

(4): The number of different E is equal to the number of subsets of $P[V_I \times V_O] = 2^{64}$.

(5): The number of different E_F is $|P[V_I \times V_O]|$.

(6): $|T[V_I \times V_O]| = 2^{64} \times 64 = 2^{70}$.

Although $T[V_I \times V_O]$ includes all possible systems over $V_I \times V_O$, there are many structures included that are not systems. An instance of such structure is any structure that contains the correspondence $\{<u_0,v_0>,<u_1,v_0>\}$ which is a destructive correspondence. Because the T-set has less restrictions on valid structure members, it is easier to work with. Also for the same reasons it will contain a superset of the S-set $S[V_I \times V_O]$. From above the following theorem is derived.

Theorem 5.5.13:

Let $S[V_I \times V_O]$ and $T[V_I \times V_O]$ be the S-set over $V_I \times V_O$ and T-set over $V_I \times V_O$ respectively. Then the cardinality of the T-set is $(2 \exp(2 \exp(|V_I| \times |V_O|))) \times (2 \exp(|V_I| \times |V_O|))$, which is also an upper bound on the cardinality of the S-set.

Proof:

(1): $|T[V_I \times V_O]| =$ (number of different E) $\times$ (number of different E_F) $= = (2 \exp|P[V_I \times V_O]|) \times (2 \exp|V_I \times V_O|)$

$= (2 \exp(2 \exp|V_I \times V_O|)) \times (2 \exp|V_I \times V_O|)$

$= (2 \exp(2 \exp|V_I| \times |V_O|)) \times (2 \exp(|V_I| \times |V_O|))$.

(2): $S[V_I \times V_O] \subseteq T[V_I \times V_O] \rightarrow |S[V_I \times V_O]| \leq |T[V_I \times V_O]|$

$\rightarrow |T[V_I \times V_O]|$ is an upper bound of the cardinality of $S[V_I \times V_O]$.

□

The properties of T-sets will be used in the next section to discuss the S-sets which is our primary goal. The mappings ϕ_I, ϕ_O, $\phi_{I\times O}$, μ, and ψ will have their counter part in the domain of S-sets and some properties derived in the domain of T-sets will carry into the domain of S-sets.

The results of the preceding discussion can be summarized as follows. Given $T^1 = T[V_I^1 \times V_O^1]$ and $T^2 = T[\phi_I(V_I^1) \times \phi_O(V_O^1)]$ two T-sets, where ϕ_I is a ϕ_I-map and ϕ_O is a ϕ_O-map. Then the ϕ_I and ϕ_O maps uniquely determine a $\phi_{I\times O}$-map $\phi_{I\times O}$, $\phi_{I\times O} : V_I^1 \times V_O^1 \rightarrow \phi_I(V_I^1) \times \phi_O(V_O^1)$. The $\phi_{I\times O}$-map $\phi_{I\times O}$, then uniquely determines a μ-map μ, $\mu : P[V_I^1 \times V_O^1] \rightarrow P[\phi_I(V_I^1) \times \phi_O(V_O^1)]$. The μ-map μ, then uniquely determines a ψ-map ψ,

$\psi : T[V_I^1 \times V_O^1] \rightarrow T[\phi_I(V_I^1) \times \phi_O(V_O^1)]$.

Conversely, given a ψ-map ψ, it uniquely determines a μ-map μ. The μ-map μ, then uniquely determines a $\phi_{I\times O}$-map $\phi_{I\times O}$. The $\phi_{I\times O}$-map $\phi_{I\times O}$, then uniquely determines a ϕ_I and ϕ_O maps. To summarize, the ψ-map ψ uniquely determines ϕ_I and ϕ_O maps and ϕ_I and ϕ_O maps uniquely determine a ψ-map ψ.

Another important result of this section is that certain properties of ϕ_I and ϕ_O maps are inherited by the ψ-map ψ and vice versa. Specifically proven here was the important fact that the diagram $(\phi_I, \phi_O) \leftrightarrow (\phi_{I\times O}) \leftrightarrow (\mu) \leftrightarrow (\psi)$ commutes when each map is 1:1 (Figure 5.1.) That means that not only ϕ_I, ϕ_O 1:1 maps induce a ψ 1:1, but also if ψ is 1:1 then it induces ϕ_I, ϕ_O 1:1 maps.

5.6 Quasimorphism

In this section, based upon the concept of morphism of groups, a new similarity measure between systems is defined that allows a comparison between arbitrary (regular and irregular) systems. This measure is called quasimorphism and is completely specified by two mappings called ϕ_I and ϕ_O. The quasimorphism will facilitate the analysis of following problems in parallel processing:

(a) system A emulating system B (three different degrees of strictness of emulation are discussed);

(b) fault tolerance/reliability (achieved by multiple mapping of same problem into a system);

(c) partitioning of a system.

In this section the relationships among systems will be explored. Since $S[V_I \times V_O] \subseteq T[V_I \times V_O]$ or the S-set over $V_I \times V_O$ is a subset of the T-set over $V_I \times V_O$, it will be shown that most relationships among systems treated as elements of T-set carry from the T-set domain to the S-set domain, while other relationships carry over in a somewhat weakened form. All the maps ϕ_I, ϕ_O, $\phi_{I\times O}$, μ, and ψ from T-sets will have their counterpart in the context of S-sets. Since the S-set $S[V_I \times V_O]$ and the T-set $T[V_I \times V_O]$ are both defined over the same underlying set $V_I \times V_O$ the following maps defined based on V_I^1, V_O^1 and V_I^2, V_O^2 are directly applicable for analysis of relationships between systems.

Definition 5.6.1:

These maps have identical meaning in the context of S-sets as in the context of T-sets. The maps are:

$\phi_I : V_I^1 \rightarrow \phi_I(V_I^1)$, onto.

$\phi_O : V_O^1 \rightarrow \phi_O(V_O^1)$, onto.

$\phi_{I\times O} : V_I{}^1 \times V_O{}^1 \rightarrow \phi_I(V_I{}^1) \times \phi_O(V_O{}^1)$, onto.

$\mu : P[V_I{}^1 \times V_O{}^1] \rightarrow P[\phi_I(V_I{}^1) \times \phi_O(V_O{}^1)]$, onto.

$\psi : T[V_I{}^1 \times V_O{}^1] \rightarrow T[\phi_I(V_I{}^1) \times \phi_O(V_O{}^1)]$, onto.

The Definitions 5.6.2 and 5.6.3 are intermediate steps used to define the quasimorphism formally.

Definition 5.6.2:

Let $\mu : P^1 \rightarrow P^2$ be a μ-map, generated by ϕ_I and ϕ_O. Let C^1 and C^2 be two C-sets. Define a *$\bar{\mu}$-correspondence from C^1 to C^2* to be any correspondence such that:

$\bar{\mu} : C^1 \rightarrow C^2$; $\bar{\mu} \triangleq \mu|_{C^1 \times C^2}$.

Note that if $<C_m^1, C_n^2> \in \bar{\mu}$ then $\bar{\mu}(C_m^1) = \mu(C_n^1)$. Clearly if μ is generated by a $\phi_{I \times O}$ then $\bar{\mu}$ is generated by the same $\phi_{I \times O}$.

Definition 5.6.3:

Let $\psi : T^1 \rightarrow T^2$ be a ψ-map, generated by ϕ_I and ϕ_O, $T^1 = T[V_I^1 \times V_O^1]$, $T^2 = T[\phi_I(V_I^1) \times \phi_O(V_O^1)]$. Let $S^1 = S[V_I^1 \times V_O^1]$ and $S^2 = S[\phi_I(V_I^1) \times \phi_O(V_O^1)]$ be two S-sets. Define a *$\bar{\psi}$-correspondence from S^1 to S^2* to be the correspondence such that:

$\bar{\psi} : S^1 \rightarrow S^2$; $\bar{\psi} \triangleq \psi|_{S^1 \times S^2}$.

Note that if $<S^{1,i}, S^{2,j}> \in \bar{\psi}$ then $\bar{\psi}(S^{1,i}) = \psi(S^{1,i})$. Clearly if ψ is generated by a μ then ψ is generated by the same μ.

Definition 5.6.4:

Let $\bar{\psi} : S^1 \rightarrow S^2$ be a $\bar{\psi}$-correspondence. Let $<S^{1,i}, S^{2,j}> \in \bar{\psi}$. Then $\bar{\psi}$ is called a *quasimorphism from $S^{1,i}$ to $S^{2,j}$*.

The Lemmas 5.6.5 to 5.6.7 describe the heritage of properties between the auxiliary correspondences. These results will be used in the Theorems 5.6.8 and 5.6.9 to describe the heritage of properties between the elementary maps ϕ_I and ϕ_O and the $\bar{\psi}$-correspondence which is the basis of quasimorphism.

Lemma 5.6.5:

Let $\bar{\mu} : C^1 \rightarrow C^2$ be a $\bar{\mu}$-correspondence generated by ϕ_I and ϕ_O, $C^1 = C[V_I^1 \times V_O^1]$, $C^2 = C[\phi_I(V_I^1) \times \phi_O(V_O^1)]$. Then $\bar{\mu}$ is an onto correspondence.

Proof:

Show $\forall\ C_m^{2,j} \in C^2\ \exists\ C_n^{1,i} \in C^1, \bar{\mu}(C_n^{1,i}) = C_m^{2,j}$.

(1): Let $C_m^{2,j} = \{<v_a,v_b>,...\}$ and $C_n^{1,i} = \{<u_a,u_b>,...\}$.

(2): $C_m^{2,j} \in C^2 \rightarrow C_m^{2,j} \in P^2$.

(3): (2), Definition 5.5.4 $\rightarrow\ \exists\ E_n^{1,i} \in P^1, \mu(E_n^{1,i}) = C_m^{2,j}$.

(4): Construct $C_n^{1,i} \subseteq E_n^{1,i}$, $C_n^{1,i} \in C^1$, $\bar{\mu}(C_n^{1,i}) = C_m^{2,j}$.

(4.1): If $E_n^{1,i}$ nondestructive then go to (5) else: $\exists\ <u_a,u_b>, <u_c,u_b> \in E_n^{1,i} \rightarrow \phi_{I\times O}(<u_a,u_b>), \phi_{I\times O}(<u_c,u_b>) \in C_m^{2,j} \rightarrow <\phi_I(u_a),\phi_O(u_b)>, <\phi_I(u_c),\phi_O(u_b)> \in C_m^{2,j}$.

(4.2): $C_m^{2,j}$ nondestructive $\rightarrow \phi_I(u_a) = \phi_I(u_c)$.

Let $C_n^{1,i} = E_n^{1,i} - \{<u_a,u_b> \mid \text{singleton}\}$.

(4.3): Let $E_n^{1,i} = C_n^{1,i}$ and go to (4.1).

(5): $C_n^{1,i} = E_n^{1,i}$, $C_n^{1,i} \in C^1$ and $\bar{\mu}(C_n^{1,i}) = C_m^{2,j}$.

□

Lemma 5.6.6:

Let $\phi_{I\times O}$ be the $\phi_{I\times O}$-map generating $\bar{\mu}$.

$\phi_{I\times O}$ is 1:1 map iff $\bar{\mu}$ is 1:1 correspondence

Proof:

Let $\bar{\mu} : C^1 \rightarrow C^2$ be the $\bar{\mu}$-correspondence.

Case 1: Show: $\bar{\mu}$ is 1:1 correspondence.

(1): $\phi_{I\times O}$ 1:1 map and Lemma 5.5.7 $\rightarrow$ μ is 1:1 map.

(2): (1) and $\bar{\mu}$ restriction correspondence of μ $\rightarrow$ $\bar{\mu}$ is 1:1 correspondence.

Case 2: Show: $\forall$ $<u_a,u_b>$, $<v_a,v_b> \in V_I^1 \times V_O^1$, $\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>)$ $\rightarrow$ $<u_a,u_b> = <v_a,v_b>$.

(1): Let C_m, $C_n \in C^1$, $C_m = \{<u_a,u_b> \mid \text{singleton}\}$, $C_n = \{<v_a,v_b> \mid \text{singleton}\}$. $\phi_{I\times O}(<u_a,u_b>) = \phi_{I\times O}(<v_a,v_b>)$ $\rightarrow$ $\{\phi_{I\times O}(<u_a,u_b>) \mid \text{singleton}\} = \{\phi_{I\times O}(<v_a,v_b>) \mid \text{singleton}\}$ $\rightarrow$ $\bar{\mu}(C_m) = \bar{\mu}(C_n)$.

(2): $\bar{\mu}$ 1:1 and (1) $\rightarrow$ $C_m = C_n$ $\rightarrow$ $\{<u_a,u_b> \mid \text{singleton}\} = \{<v_a,v_b> \mid \text{singleton}\}$ $\rightarrow$ $<u_a,u_b> = <v_a,v_b>$.

□

Lemma 5.6.7:

Let $\mu : P^1 \rightarrow P^2$ be a μ-map. Let $\bar{\mu}$ be the restriction $\bar{\mu}$-correspondence of the μ.

μ is 1:1 map iff $\bar{\mu}$ is 1:1 correspondence.

Proof:

Let $\bar{\mu} : C^1 \rightarrow C^2$ be the $\bar{\mu}$-correspondence.

Case 1: Show: $\bar{\mu}$ is 1:1 correspondence.

(1): μ 1:1 map and $\bar{\mu}$ restriction of μ $\rightarrow$ $\bar{\mu}$ is 1:1 correspondence.

Case 2: Show μ is 1:1 map.

(1): $\bar{\mu}$ 1:1 and Lemma 5.6.6 $\rightarrow$ $\phi_{I \times O}$ is 1:1 map.

(2): (1) and Lemma 5.5.7 $\rightarrow$ μ is 1:1 map.

□

The next two theorems relate the properties of ϕ_I and ϕ_O and the $\bar{\psi}$-correspondence. The significance of these properties is described at the end of this section in details.

Theorem 5.6.8:

Let $\bar{\psi} : S^1 \rightarrow S^2$ be the $\bar{\psi}$-correspondence, generated by ϕ_I and ϕ_O. If ϕ_I and ϕ_O are 1:1 maps then $\bar{\psi}$ is 1:1 correspondence.

Proof:

(1): Theorem 5.5.9 $\rightarrow$ ψ is 1:1 map.

(2): $\bar{\psi}$ restriction correspondence of ψ $\rightarrow$ $\bar{\psi}$ is 1:1 correspondence.

□

Theorem 5.6.9:

Let $\bar{\psi} : S^1 \rightarrow S^2$ be the $\bar{\psi}$-correspondence generated by ϕ_I and ϕ_O.

If $\bar{\psi}$ is 1:1 correspondence then ϕ_I and ϕ_O are 1:1 maps.

Proof:

The procedure will be done using contradiction. Let $\bar{\psi}$ be 1:1 correspondence and assume that ϕ_I or ϕ_O (or both) is (are) not 1:1. For each case construct $S^{1,i}, S^{1,j} \in S^1 \ni \bar{\psi}(S^{1,i}) = \bar{\psi}(S^{1,j}) \in S^2$ and $S^{1,i} \neq S^{1,j}$ therefore implying $\bar{\psi}$ is not 1:1 correspondence which is a contradiction.

Case 1: ϕ_I not 1:1 map. Let $V_I^1 = \{u_1, u_2, \ldots u_m\}$, $V_O^1 = \{v_1, v_2, \ldots v_n\}$,

$\phi_I(V_I^1) = \{w_1,w_2,...w_r\}$, $\phi_O(V_O^1) = \{x_1,x_2,...x_s\}$.

(1): ϕ_I not 1:1 $\rightarrow$ $\exists$ $u_a,u_b \in V_I^1$, $\phi_I(u_a) = \phi_I(u_b)$.

(2): Construct $S^{1,i} \in S^1$ as follows:

(2.1): $C^{1,i} = \{C_p^{1,i} \mid p=1,2,...m\} \cup \{\emptyset_c\} = \{\{<u_p,v_c> \mid v_c \in V_O^1\} \mid u_p \in V_I^1, p=1,2,...m\} \cup \{\emptyset_c\}$.

(2.2): $C_F^{1,i} = \{<u_a,v_1> \mid \text{singleton}\}$.

(2.3): $S^{1,i} = <C^{1,i},C_F^{1,i}>$.

(3): Construct $S^{1,j} \in S^1$ as follows:

(3.1): $C^{1,j} = C^{1,i}$.

(3.2): $C_F^{1,j} = \{<u_b,v_1> \mid \text{singleton}\}$.

(3.3): $S^{1,j} = <C^{1,j},C_F^{1,j}>$.

(4): (1), (2), and (3) $\rightarrow$ $\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\} = \{\mu(\emptyset_c), \mu(C_p^{1,j}) \mid p=1,2,...,m\}$.

(5): $\mu(C_F^{1,i}) = \mu(\{<u_a,v_1> \mid \text{singleton}\}) = \{<\phi_I(u_a),\phi_O(v_1)> \mid \text{singleton}\} = \{<\phi_I(u_b),\phi_O(v_1)> \mid \text{singleton}\} = \mu(\{<u_b,v_1> \mid \text{singleton}\}) = \mu(C_F^{1,j})$.

(6): (4), (5) $\rightarrow$ $\psi(S^{1,i}) = \psi(S^{1,j}) \in T^2$.

(7): (2.2), (3.2) $\rightarrow$ $C_F^{1,j} \neq C_F^{1,j}$ $\rightarrow$ $S^{1,i} \neq S^{1,j}$.

(8): Show $\psi(S^{1,i}) = \psi(<C^{1,i},C_F^{1,i}>) \in S^2$.

(8.1): Show $<\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\}> \in K^2$.

(8.1.1): Show $\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\} \subseteq C^2$.

(8.1.1.1): $\mu(\emptyset_c) = \emptyset_c \in C^2$.

(8.1.1.2): $\mu(C_p^{1,i}) = \mu(\{<u_p,v_c> \mid v_c \in V_O^1\}) = \{<\phi_I(u_p),\phi_O(v_c)> \mid v_c \in V_O^1\} \in C^2$.

(8.1.1.3): (8.1.1.1), (8.1.1.2) $\rightarrow \{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\} \subseteq C^2$.

(8.1.2): Show $\phi_I(V_I^1) = s(\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\})$.

(8.1.2.1): $s(\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\}) = s(\{\mu(C_p^{1,i}) \mid p=1,2,...m\}) = s(\{\mu(\{<u_p,v_c> \mid v_c \in V_O^1\}) \mid u_p \in V_I^1\}) = s(\{\{<\phi_I(u_p),\phi_O(v_c)> \mid v_c \in V_O^1\}) \mid u_p \in V_I^1\}) = \{\phi_I(u_p) \mid u_p \in V_I^1\} = \phi_I(V_I^1)$.

(8.1.3): Show $\phi_O(V_O^1) = d(\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\})$.

(8.1.3.1): $d(\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\}) = d(\{\mu(C_p^{1,i}) \mid p=1,2,...m\}) = d(\{\mu(\{<u_p,v_c> \mid v_c \in V_O^1\}) \mid u_p \in V_I^1\}) = d(\{\{<\phi_I(u_p), \phi_O(v_c)> \mid v_c \in V_O^1\}) \mid u_p \in V_I^1\}) = \{\phi_O(v_c) \mid v_c \in V_O^1\} = \phi_O(V_O^1)$.

(8.1.4): Show $|\{\mu(\emptyset_c),\mu(C_p^{1,i}) \ni p=1,2,...m\}| \geq 2$.

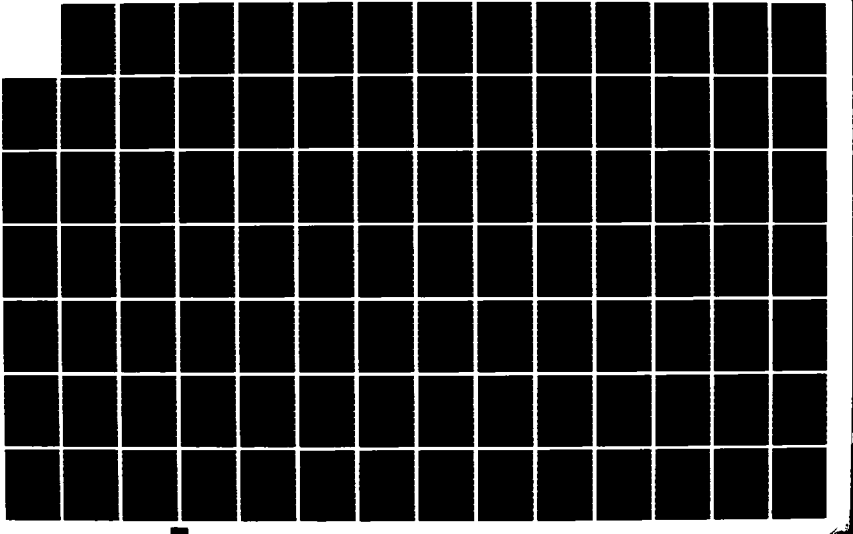
(8.1.4.1): $\mu(\emptyset_c) = \emptyset_c$, $\mu(C_p^{1,i}) \neq \emptyset_c \rightarrow |\{\mu(\emptyset_c), \mu(C_p^{1,i}) \ni p=1,2,...m\}| \geq 2$.

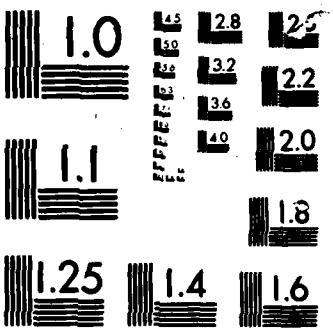
(8.1.5): (8.1.1), (8.1.2), (8.1.3), and (8.1.4) $\rightarrow <\{\mu(\emptyset_c), \mu(C_p^{1,i}) \mid p=1,2,...m\}> \in K^2$.

(8.2): Show $\mu(C_F^{1,i})$ is a feedback correspondence over $\phi_I(V_I^1) \times \phi_O(V_O^1)$.

(8.2.1): $\mu(C_F^{1,i}) = \mu(\{<u_a,v_1> \mid \text{singleton}\}) = \{<\phi_I(u_a),\phi_O(v_1)> \mid \text{singleton}\} \rightarrow \mu(C_F^{1,i})$ is a feedback correspondence over $\phi_I(V_I^1) \times \phi_O(V_O^1)$.

AD-A167 336 DISTRIBUTED COMPUTING FOR SIGNAL PROCESSING: 2/3
TOPOLOGICAL PROPERTIES OF IN..(U) PURDUE UNIV LAFAYETTE
IN R R SEBAN DEC 85 ARO-18790.17-EL-APP-E
UNCLASSIFIED DAAG29-82-K-0101 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.2
2.0
1.8

(8.3): (8.1), (8.2) $\rightarrow$ $\psi(S^{1,i}) = S^{2,k} \in S^2$.

(9): (8) $\rightarrow$ $<S^{1,i},S^{2,k}> \in \psi$ and $<S^{1,j},S^{2,k}> \in \bar{\psi}$.

(10): (6), (9) $\rightarrow$ $\bar{\psi}(S^{1,i}) = \bar{\psi}(S^{1,j})$.

(11): (7), (10) $\rightarrow$ $\bar{\psi}$ not 1:1 $\rightarrow$ contradiction $\rightarrow$ ϕ_I is 1:1 map.

Case 2: ϕ_O not 1:1 map. Let $V_I^1 = \{u_1,u_2,...u_m\}$, $V_O^1 = \{v_1,v_2,...v_n\}$, $\phi_I(V_I^1) = \{w_1,w_2,...w_r\}$, $\phi_O(V_O^1) = \{x_1,x_2,...x_s\}$.

(1): ϕ_O not 1:1 $\rightarrow$ $\exists$ $v_a,v_b \in V_O^1$, $\phi_O(v_a) = \phi_O(v_b)$.

(2): Steps (2) to (11) are same as Case 1 except (2.2) and (3.2).

(2.2): $C_F^{1,i} = \{<u_1,v_a> \mid \text{singleton}\}$.

(3.2): $C_F^{1,j} = \{<u_1,v_b> \mid \text{singleton}\}$.

□

The theoretical work presented in this section has the following physical implications. Given two systems S^1 and S^2 with arbitrary vertex descriptions, if there exist $\bar{\psi}$ that is, a ϕ_I and ϕ_O with the proper constraints from S^1 to S^2, then S^1 and S^2 are similar in some sense. If the $\bar{\psi}$ quasimorphism is 1:1, then in fact the systems are isomorphic, that is identical up to the naming of the vertices. The $\bar{\psi} = <\phi_I, \phi_O>$ will be used in the following problems: (1) emulation of systems; (2) identifying equivalent systems; and (3) partitioning of a network.

The following two theorems discuss some basic properties of quasimorphism $\bar{\psi}$. In the study of mathematical relations, three properties are of utmost importance. The properties are: reflexive, symmetric, and transitive which, if they hold, say that the relation is an equivalence relation. Although $\bar{\psi}$-correspondence is not a relation, properties similar to the three above can be

defined. They do have physical significance concerning quasimorphism between systems as described in the end of this section.

Theorem 5.6.10:

Let S^1, S^2, and S^3 be three systems. The quasimorphism has the following properties.

(1) $\exists\ \bar{\psi}$ such that $\bar{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2 \not\rightarrow \exists\ \bar{\psi}^2$ such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$ and $\bar{\psi}^2(S^2) = S^3 \rightarrow \exists\ \bar{\psi}$, $\bar{\psi}(S^1) = S^3$.

Proof:

(1): Need to show $\exists\ \bar{\psi}$ such that $\bar{\psi}(S^1) = S^1$. Let $\bar{\psi} = <\phi_I,\phi_O>$ be such that ϕ_I,ϕ_O are identity maps. The rest is straightforward.

(2): Must show $\bar{\psi}^1(S^1) = S^2 \not\rightarrow \exists\ \bar{\psi}^2$ such that $\bar{\psi}^2(S^2) = S^1$. Construct an example of S^1 and S^2 such that $<S^1,S^2> \in \bar{\psi}$ and there does not exist $\bar{\psi}^2 \ni \bar{\psi}^2(S^2) = S^1$.

Let	Let
$S^1 = <C^1, C_F^1>$,	$S^2 = <C^2, C_F^2>$,
$V_I^1 = \{v_a, v_b\}$, $V_O^1 = \{u_c, u_d\}$,	$V_I^2 = \{w_a\}$, $V_O^2 = \{x_c, x_d\}$,
$C_F^1 = \{<v_a,u_c>, <v_b,u_d>\}$,	$C_F^2 = \{<w_a,x_c>\}$,
$C^1 = \{C_0^1, C_1^1\}$,	$C^2 = \{C_0^2, C_1^2\}$,
$C_0^1 = \{<v_b,u_d>\}$,	$C_0^2 = \{<w_b,x_c>\}$,
$C_1^1 = \{<v_a,u_c>, <v_a,u_d>\}$.	$C_1^2 = \{<w_a,x_c>, <w_a,x_d>\}$.

Then $\bar{\psi} = <\phi_I,\phi_O>$ with $\phi_I(v_a) = w_a$, $\phi_I(v_b) = w_a$, $\phi_O(u_c) = x_d$, $\phi_O(u_d) = x_c$ is a quasimorphism $\bar{\psi}$, $\bar{\psi}(S^1) = S^1$. but there does

not exist a quasimorphism from S^2 to S^1.

(3): Must show: $\bar{\psi}^1(S^1) = S^2, \bar{\psi}^2(S^2) = S^3 \rightarrow \exists\, \bar{\psi}, \bar{\psi}(S^1) = S^3$. This will be shown by exhibiting quasimorphism $\bar{\psi}, \bar{\psi}(S^1) = S^3$.

(1): Let $S^1 = <C^1, C_F^1>$; $S^2 = <C^2, C_F^2>$; and $S^3 = <C^3, C_F^3>$ be three systems.

(2): $\bar{\psi}^1(S^1) = S^2$

$\rightarrow \exists\, \phi_I^1 : V_I^1 \rightarrow \phi_I^1(V_I^1)$, and $\exists\, \phi_O^1 : V_O^1 \rightarrow \phi_O^1(V_O^1)$.

(3): (2) $\rightarrow \exists\, \phi_{I \times O}^1 : V_I^1 \times V_O^1 \rightarrow \phi_I^1(V_I^1) \times \phi_O^1(V_O^1)$ and

$\exists\, \mu^1 : P[V_I^1 \times V_O^1] \rightarrow P[\phi_I^1(V_I^1) \times \phi_O^1(V_O^1)]$.

(4): $\bar{\psi}^2(S^2) = S^3$

$\rightarrow \exists\, \phi_I^2 : \phi_I^1(V_I^1) \rightarrow \phi_I^2(\phi_I^1(V_I^1))$, and

$\exists\, \phi_O^2 : \phi_O^1(V_O^1) \rightarrow \phi_O^2(\phi_O^1(V_O^1))$,

(5): (4) $\rightarrow \exists\, \phi_{I \times O}^2 : \phi_I^1(V_I^1) \times \phi_O^1(V_O^1)$

$\rightarrow \phi_I^2(\phi_I^1(V_I^1)) \times \phi_O^2(\phi_O^1(V_O^1))$, and

$\exists\, \mu^2 : P[\phi_I^1(V_I^1) \times \phi_O^1(V_O^1)]$

$\rightarrow P[\phi_I^2(\phi_I^1(V_I^1)) \times \phi_O^2(\phi_O^1(V_O^1))]$.

(6): Define: $\phi_I = \phi_I^2 \circ \phi_I^1 : V_I^1 \rightarrow \phi_I^2(\phi_I^1(V_I^1))$. ("$\circ$" *is composition of maps*)

Clearly: ϕ_I is map.

(7): Define: $\phi_O = \phi_O^2 \circ \phi_O^1 : V_O^1 \rightarrow \phi_O^2(\phi_O^1(V_O^1))$.

Clearly: ϕ_O is map.

(8): Define: $\phi_{I \times O} = \phi_{I \times O}^2 \circ \phi_{I \times O}^1 : \phi_I^1(V_I^1) \times \phi_O^1(V_O^1)$

$\rightarrow \phi_I^2(\phi_I^1(V_I^1)) \times \phi_O^2(\phi_O^1(V_O^1))$.

Clearly: ϕ_{IxO} is map.

(9): Define: $\mu = \mu^2 \circ \mu^1 : P[\phi_I^1(V_I^1) \times \phi_O^1(V_O^1)]$
$\rightarrow P[\phi_I^2(\phi_I^1(V_I^1)) \times \phi_O^2(\phi_O^1(V_O^1))]$,

Clearly: μ is map.

(10): Claim: $\overline{\psi} = <\phi_I,\phi_O>$ is quasimorphism, $\overline{\psi}(S^1) = S^3$.

(11): Show: $\mu(C_F^1) = C_F^3$.

$\overline{\psi}^1(S^1) = S^2 \rightarrow \mu^1(C_F^1) = C_F^2$.

$\overline{\psi}^2(S^2) = S^3 \rightarrow \mu^2(C_F^2) = C_F^3$.

$\rightarrow \mu^2(\mu^1(C_F^1)) = C_F^3 \rightarrow (\mu^2 \circ \mu^1)(C_F^1) = \mu(C_F^1) = C_F^3$.

(12): Show: $\forall C_m^1 \in C^1 \ \exists C_q^3 \in C^3 \ni : \mu(C_m^1) = C_q^3$.

(a): $\overline{\psi}^1(S^1) = S^2$

$\rightarrow \forall C_m^1 \in C^1 \ \exists C_n^2 \in C^2 \ni : \mu^1(C_m^1) = C_n^2$.

(b): $\overline{\psi}^2(S^2) = S^3$

$\rightarrow \forall C_n^2 \in C^2 \ \exists C_q^3 \in C^3 \ni : \mu^2(C_n^2) = C_q^3$.

(c): (a), (b) $\rightarrow \forall C_m^1 \in C^1 \ \exists C_q^3 \in C^3$
$\ni : \mu^2(\mu^1(C_m^1)) = \mu^2(C_n^2)$.

(13): (11) and (12) $\rightarrow \overline{\psi} = <\phi_I,\phi_O>$ is quasimorphism $\overline{\psi}(S^1)$
$= S^3$.

□

Theorem 5.6.11:

Let S^1, S^2, and S^3 be three systems. The quasimorphism 1:1 has the following properties.

(1) $\exists \overline{\psi}$, 1:1 such that $\overline{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2$, 1:1 $\rightarrow \exists\ \bar{\psi}^2$, 1:1 such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$, 1:1 and $\bar{\psi}^2(S^2) = S^3$, 1:1 $\rightarrow \exists\ \bar{\psi}$, 1:1, $\bar{\psi}(S^1) = S^3$.

Proof:

(1): Need to show $\exists\ \bar{\psi}$ such that $\bar{\psi}(S^1) = S^1$. Let $\bar{\psi} = <\phi_I,\phi_O>$ be such that ϕ_I,ϕ_O are identity maps. The rest is straightforward.

(2): Must show $\bar{\psi}^1(S^1) = S^2 \rightarrow \exists\ \bar{\psi}^2$ such that $\bar{\psi}^2(S^2) = S^1$.

(1): $<S^1,S^2> \in \bar{\psi}$, 1:1 $\rightarrow \exists\ \phi_I : V_I^1 \rightarrow \phi_I(V_I^1)$, 1:1 and $\exists\ \phi_O : V_O^1 \rightarrow \phi_O(V_O^1)$, 1:1.

(2): (1) $\rightarrow \exists\ \phi_I^2 : \phi_I(V_I^1) \rightarrow V_I^1$, 1:1, $\phi_I^2 = \phi_I^{-1}$, and $\exists\ \phi_O^2 : \phi_O(V_O^1) \rightarrow V_O^1$, 1:1, $\phi_O^2 = \phi_O^{-1}$.

(3): (2) $\rightarrow \exists \phi_{I\times O}^2 : \phi_I(V_I^1) \times \phi_O(V_O^1) \rightarrow V_I^1 \times V_O^1$, 1:1, $\phi_{I\times O}^2 = \phi_{I\times O}^{-1}$.

(4): (3) $\rightarrow \exists\ \mu^2 : P[\phi_I(V_I^1) \times \phi_O(V_O^1)] \rightarrow P[V_I^1 \times V_O^1]$, 1:1, $\mu^2 = \mu^{-1}$.

The rest is straightforward.

(3): The proof is similar to the proof of (3) of Theorem 5.6.10 except the maps are 1:1.

□

Following are two examples of quasimorphism between systems, one where the quasimorphism is not 1:1 and the other where it is 1:1.

Example 5.6.12:

Consider the following two systems $S^i = \langle C^i, C_F^i \rangle$, $S^j = \langle C^j, C_F^j \rangle$ see Figure 5.2.

$S^i = \langle C^i, C_F^i \rangle$,
$V_I^i = \{u_1, u_2, u_3\}$,
$V_O^i = \{v_1, v_2, v_3\}$,
$C_F^i = \emptyset$,
$C^i = \{C_1^i, C_2^i, C_3^i\}$,
$C_1^i = \{\langle u_1, v_1 \rangle, \langle u_1, v_2 \rangle\}$,
$C_2^i = \{\langle u_3, v_2 \rangle, \langle u_3, v_3 \rangle\}$,
$C_3^i = \{\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle\}$.

$S^j = \langle C^j, C_F^j \rangle$,
$V_I^j = \{w_1, w_2\}$,
$V_O^j = \{x_1, x_2, x_3\}$,
$C_F^j = \emptyset$,
$C^j = \{C_1^j, C_2^j, C_3^j\}$,
$C_1^j = \{\langle w_1, x_1 \rangle, \langle w_1, x_2 \rangle\}$,
$C_2^j = \{\langle w_2, x_2 \rangle, \langle w_2, x_3 \rangle\}$,
$C_3^j = \{\langle w_1, x_1 \rangle, \langle w_2, x_2 \rangle\}$.

Find a quasimorphism from S^i to S^j.

Solution:

Let $\bar{\psi} = \langle \phi_I, \phi_O \rangle$, $\phi_I(u_1)=w_1$, $\phi_I(u_2)=w_2$, $\phi_I(u_3)=w_2$, $\phi_O(v_1)=x_1$, $\phi_O(v_2)=x_2$, $\phi_O(v_3)=x_3$.

$\bar{\psi}(S^i) = \langle \{\bar{\mu}(C_m^i) \mid m=1,2,3\}, \bar{\mu}(C_F^i) \rangle = S^j$. Therefore $\bar{\psi}$ is a quasimorphism. Since $\phi_I(u_2) = \phi_I(u_3)$, therefore $\bar{\psi}$ is not 1:1.

Example 5.6.19:

Consider the following two systems $S^i = \langle C^i, C_F^i \rangle$, $S^j = \langle C^j, C_F^j \rangle$ see Figure 5.3.

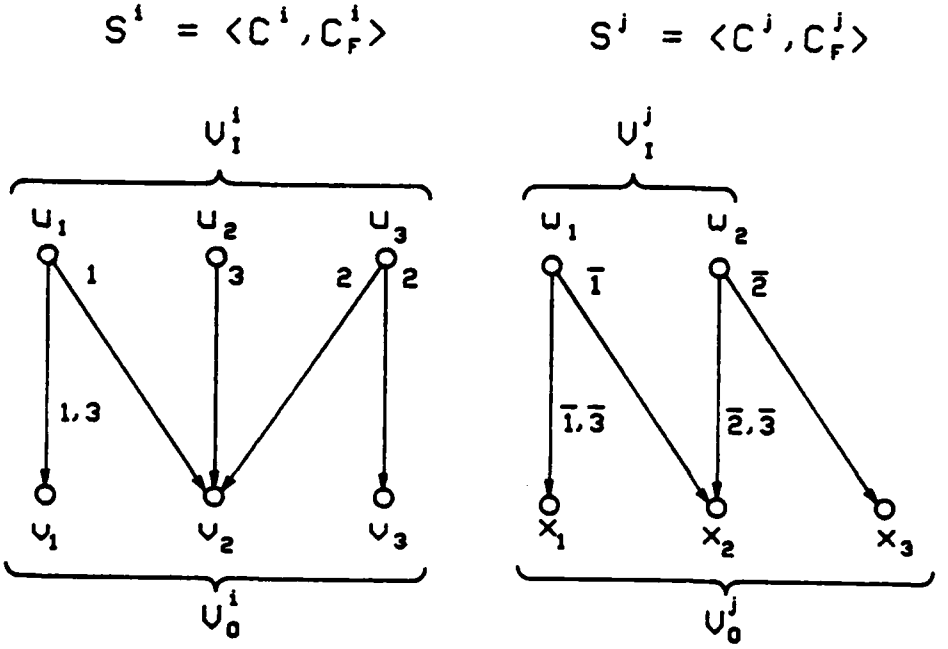


Figure 5.2:
Systems $S^i = \langle C^i, C_F^i \rangle$, $S^j = \langle C^j, C_F^j \rangle$ for Example 5.6.12.

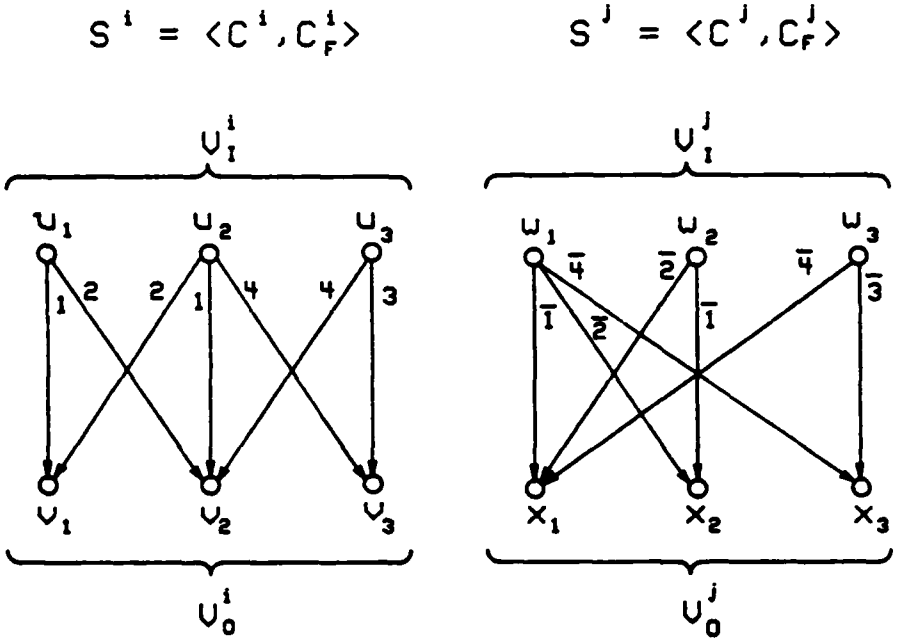


Figure 5.3:
Systems $S^i = \langle C^i, C_F^i \rangle$, $S^j = \langle C^j, C_F^j \rangle$ for Example 5.6.13.

$S^i = \langle C^i, C_F^i \rangle$,
$V_I^i = \{u_1, u_2, u_3\}$,
$V_O^i = \{v_1, v_2, v_3\}$,
$C_F^i = \emptyset$,
$C^i = \{C_1^i, C_2^i, C_3^i, C_4^i\}$,
$C_1^i = \{\langle u_1, v_1\rangle, \langle u_2, v_2\rangle\}$,
$C_2^i = \{\langle u_1, v_2\rangle, \langle u_2, v_1\rangle\}$,
$C_3^i = \{\langle u_3, v_3\rangle\}$,
$C_4^i = \{\langle u_2, v_3\rangle, \langle u_3, v_2\rangle\}$.

$S^j = \langle C^j, C_F^j \rangle$,
$V_I^j = \{w_1, w_2, w_3\}$,
$V_O^j = \{x_1, x_2, x_3\}$,
$C_F^j = \emptyset$,
$C^j = \{C_1^j, C_2^j, C_3^j, C_4^j\}$,
$C_1^j = \{\langle w_1, x_1\rangle, \langle w_2, x_2\rangle\}$,
$C_2^j = \{\langle w_1, x_2\rangle, \langle w_2, x_1\rangle\}$,
$C_3^j = \{\langle w_3, x_3\rangle\}$,
$C_4^j = \{\langle w_1, x_3\rangle, \langle w_3, x_1\rangle\}$.

Find a quasimorphism from S^i to S^j.

Solution:

Let $\bar{\psi} = \langle \phi_I, \phi_O \rangle$, $\phi_I(u_1)=w_2$, $\phi_I(u_2)=w_1$, $\phi_I(u_3)=w_3$, $\phi_O(v_1)=x_2$, $\phi_O(v_2)=x_1$, $\phi_O(v_3)=x_3$.

$\bar{\psi}(S^i) = \langle \{\bar{\mu}(C_m^i) \mid m=1,2,3,4\}, \bar{\mu}(C_F^i) \rangle = S^j$. Therefore $\bar{\psi}$ is a quasimorphism. Since $\phi_I(u_a) \neq \phi_I(u_b)$, $\phi_O(v_a) \neq \phi_O(v_b)$, therefore $\bar{\psi}$ is 1:1.

The genealogy of maps and correspondences discussed in this and preceding sections can be seen in the diagram in Figure 5.1. The diagram shows how the maps and correspondences defined in this and previous sections are related. The line with arrow represents that the map (correspondence) at the head of the arrow was defined by using the correspondence at the tail of the arrow. For example, the μ-map μ was used to define the $\bar{\mu}$-correspondence $\bar{\mu}$. In Section 5.5 the left side and the root of the tree were explored. In this section the right side of the tree was explored. It was shown that ϕ_I-map ϕ_I and ϕ_O-map ϕ_O uniquely determine a $\phi_{I\times O}$-map $\phi_{I\times O}$. The $\phi_{I\times O}$-map $\phi_{I\times O}$ then

uniquely determines a μ-map μ. The μ-map μ uniquely determines a $\bar{\mu}$-correspondence $\bar{\mu}$ and the ψ-map ψ. The ψ-map ψ uniquely determines a $\bar{\psi}$-correspondence $\bar{\psi}$. Therefore the ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O uniquely determine a $\bar{\psi}$-correspondence $\bar{\psi}$. Similarly the reverse of the procedure can be used to show that $\bar{\psi}$-correspondence $\bar{\psi}$ uniquely determines a ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O.

Several properties are also inherited from some maps by others. In particular, if ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O are both 1:1 then $\bar{\psi}$-correspondence $\bar{\psi}$ is also 1:1 correspondence. It is more surprising though that the converse hold as well, that is if $\bar{\psi}$-correspondence $\bar{\psi}$ is 1:1 correspondence, then ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O are both 1:1 maps.

5.7 Emulation of Systems

In this chapter we apply some of the theoretical developments from the previous sections. The emulation will be defined and can be viewed as an application of quasimorphism. The definition of emulation here is similar to the one used in [FiF82] in analysis of quotient networks. Examples of arbitrary system emulation are given in details in the end of the section.

Definition 5.7.1:

Let $S^i \in S[V_I^1 \times V_O^1]$ and $S^j \in S[V_I^2 \times V_O^2]$ be two systems. The *emulation of* $S^i \in S[V_I^1 \times V_O^1]$ *by* $S^j \in S[V_I^2 \times V_O^2]$ can be viewed as a two step procedure

(1) Find a relabeling and reduction (that preserves the basic structure) of S^i using quasimorphism.

(2) Find the subsystem type of the quasimorphism of S^i $\bar{\psi}(S^i)$ in S^j.

Definition 5.7.2:

Let $S^i \in S[V_I^1 \times V_O^1]$ and $S^j \in S[V_I^2 \times V_O^2]$ be two systems.

If $\bar{\psi}(S^i) \subseteq$ a S^j, then it is called *emulation type a.*

If $\bar{\psi}(S^i) \subseteq$ b S^j, then it is called *emulation type b.*

If $\bar{\psi}(S^i) \subseteq$ c S^j, then it is called *emulation type c.*

Physical implications: Let $S^1 = <C^1, C_F^1> = <\{C_m^1 \mid m=1,2,..p\}, C_F^1>$ and $S^2 = <C^2, C_F^2> = <\{C_n^2 \mid n=1,2,..q\}, C_F^2>$ be two systems. If $\bar{\psi}(S^1) \subseteq$ a S^2 then the system S^2 can emulate system S^1 as follows. The movement of the data is accomplished (a) by using the network $\{C_n^2 \mid n=1,2,..q\}$ correspondences, and (b) by using the feedback or internal connection of the device connected to both input and output of the network. This type of emulation always exists if the S^2 system is partially or fully recirculating. If the system S^2 is partially or fully recirculating then $\exists <v_x,v_y> \in C_F^2$. Then using maps $\phi_I(v_i) = v_x$, $\forall v_i \in V_I^1$, $\phi_O(v_j) = v_y$, $\forall v_j \in V_O^1$ will satisfy the necessary conditions for an emulation of the type a. This however will result in a very poor computational load balance. Great improvement in the computational load balancing optimality will result if the quasimorphism is 1:1. Then each device in $\bar{\psi}(S^1)$ (the image of S^1 under $\bar{\psi}$) will

have same amount of computation (data) as the corresponding device in S^1.

Physical implication: If $\bar{\psi}(S^1) \subseteq_b S^2$ then the system S^2 can emulate system S^1. The movement of the data is accomplished by using the network correspondences $\{C_n^2 \mid n=1,2,..q\}$. This type of emulation is harder to achieve than the type a since the C_F^2 contribution cannot be used to move the data. Again, as in type a, the load balancing optimality will greatly increase if the quasimorphism is 1:1. If the quasimorphism is 1:1 then the load balancing as well as utilization in the image of S^1 in S^2 will be identical to that in S^1.

Physical implications: Emulation type c. Since it is required in type b that $\forall C_m^1 \in C^1 \quad \exists C_n^2 \ni C_m^1 \subseteq C_n^2$ there may be some side effects caused by C_n^2 emulating the correspondence C_m^1. Moreover, these uncontrolled side effects will not allow partitions to operate independently. That is, connections that are part of C_n^2, but not part of C_m^1, may be established when C_n^2 is used to emulate C_m^1. This may or may not be a problem. To analyze this potential problem, the type c was created. With a type c emulation, when the system S^2 emulates system S^1, the movement of the data is accomplished by a subset of C^2. The difference between type b and type c is that in type c, $\forall C_m^1 \in C^1 \quad \exists C_n^2 \in C^2 \ni C_m^1 = C_n^2$. This requirement will eliminate the side effects that type b has. More importantly it means that $\bar{\psi}(S^1)$ is actually an autonomous subsystem of S^2. The autonomous property will be exploited further in later chapters studying partitionability.

The following two examples illustrate two types of emulation where in the first the quasimorphism is not 1:1 and the second has quasimorphism 1:1.

Example 5.7.3:

Consider the following two systems $S^i = <C^i, C_F^i>$, $S^k = <C^k, C_F^k>$ see Figure 5.2 for S^i and Figure 5.4 for S^k.

$S^i = <C^i, C_F^i>$,

$V_I^i = \{u_1, u_2, u_3\}$, $V_O^i = \{v_1, v_2, v_3\}$,

$C_F^i = \emptyset$, $C^i = \{C_1^i, C_2^i, C_3^i\}$,

$C_1^i = \{<u_1, v_1>, <u_1, v_2>\}$, $C_2^i = \{<u_3, v_2>, <u_3, v_3>\}$,

$C_3^i = \{<u_1, v_1>, <u_2, v_2>\}$.

$S^k = <C^k, C_F^k>$,

$V_I^k = \{w_1, w_2, w_3\}$, $V_O^k = \{x_1, x_2, x_3\}$,

$C_F^k = \emptyset$, $C^k = \{C_1^k, C_2^k, C_3^k\}$,

$C_1^k = \{<w_1, x_1>, <w_1, x_2>\}$, $C_2^k = \{<w_2, x_2>, <w_2, x_3>\}$,

$C_3^k = \{<w_1, x_1>, <w_2, x_2>, <w_3, w_3>\}$.

Find an emulation from S^i to S^k.

Solution:

Let $\bar{\psi} = <\phi_I, \phi_O>$, $\phi_I(u_1)=w_1$, $\phi_I(u_2)=w_2$, $\phi_I(u_3)=w_2$, $\phi_O(v_1)=x_1$, $\phi_O(v_2)=x_2$, $\phi_O(v_3)=x_3$, as in Example 5.6.12.

$\bar{\psi}(S^i) = <\{\bar{\mu}(C_m^i) \mid m=1,2,3\}, \bar{\mu}(C_F^i)> = S^j$ (see Figure 5.4 for S^j). Therefore $\bar{\psi}$ is a quasimorphism from S^i to S^j. Since $\phi_I(u_2) = \phi_I(u_3)$, therefore $\bar{\psi}$ is not 1:1.

Since $C_1^j=C_1^k$, $C_2^j=C_2^k$, $C_3^j \subset C_3^k$ therefore $\bar{\psi}(S^i) \subseteq$ b S^k, and this is emulation of type b, not 1:1.

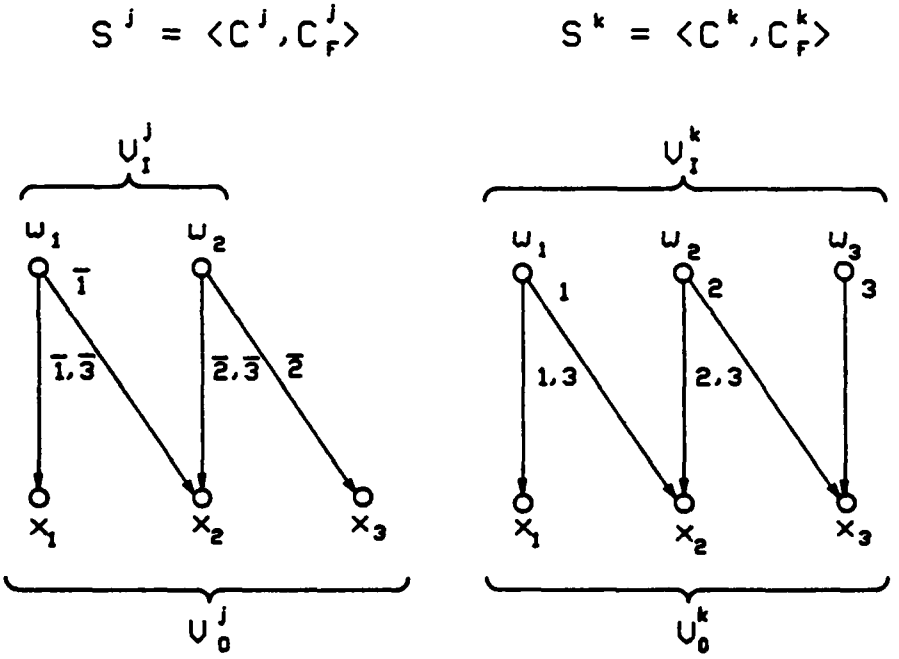


Figure 5.4:
Systems $S^j = \langle C^j, C_F^j \rangle$, $S^k = \langle C^k, C_F^k \rangle$ for Example 5.7.3.

Example 5.7.4:

Consider the following two systems $S^i = \langle C^i, C_F^i \rangle$, $S^k = \langle C^k, C_F^k \rangle$ see Figure 5.3 for S^i and Figure 5.5 for S^k.

$S^i = \langle C^i, C_F^i \rangle$,

$V_I^i = \{u_1, u_2, u_3\}$, $V_O^i = \{v_1, v_2, v_3\}$,

$C_F^i = \emptyset$, $C^i = \{C_1^i, C_2^i, C_3^i, C_4^i\}$,

$C_1^i = \{\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle\}$,

$C_2^i = \{\langle u_1, v_2 \rangle, \langle u_2, v_1 \rangle\}$, $C_3^i = \{\langle u_3, v_3 \rangle\}$.

$C_4^i = \{\langle u_2, v_3 \rangle, \langle u_3, v_2 \rangle\}$.

$S^k = \langle C^k, C_F^k \rangle$,

$V_I^k = \{w_1, w_2, w_3, w_4\}$, $V_O^k = \{x_1, x_2, x_3, x_4\}$,

$C_F^k = \emptyset$, $C^k = \{C_1^k, C_2^k, C_3^k\}$,

$C_1^k = \{\langle w_1, x_1 \rangle, \langle w_2, x_2 \rangle, \langle w_3, x_3 \rangle, \langle w_4, x_4 \rangle\}$,

$C_2^k = \{\langle w_1, x_2 \rangle, \langle w_2, x_1 \rangle\}, \langle w_3, x_4 \rangle, \langle w_4, x_3 \rangle\}$,

$C_3^k = \{\langle w_1, x_3 \rangle, \langle w_2, x_4 \rangle, \langle w_3, x_1 \rangle, \langle w_4, x_2 \rangle\}$.

Find an emulation from S^i to S^k.

Solution:

Let $\bar{\psi} = \langle \phi_I, \phi_O \rangle$, $\phi_I(u_1)=w_2$, $\phi_I(u_2)=w_1$, $\phi_I(u_3)=w_3$, $\phi_O(v_1)=x_2$, $\phi_O(v_2)=x_1$, $\phi_O(v_3)=x_3$ as in Example 5.6.13:

$\bar{\psi}(S^i) = \langle \{\bar{\mu}(C_m^i) \mid m=1,2,3,4\}, \bar{\mu}(C_F^i) \rangle = S^j$ (see Figure 5.5 for S^j).

Therefore $\bar{\psi}$ is a quasimorphism from S^i to S^j.

Since $\forall u_a, u_b \in V_I^i \quad \phi_I(u_a) \neq \phi_I(u_b)$ and $\forall v_a, v_b \in V_O^i \quad \phi_I(v_a) \neq \phi_I(v_b)$ therefore $\bar{\psi}$ is 1:1. Since $C_1^j \subset C_1^k$, $C_2^j \subset C_2^k$, $C_3^j \subset C_1^k$, and $C_4^j \subset C_3^k$ therefore $\bar{\psi}(S^i) \subseteq_b S^k$, and this is emulation of type b, 1:1.

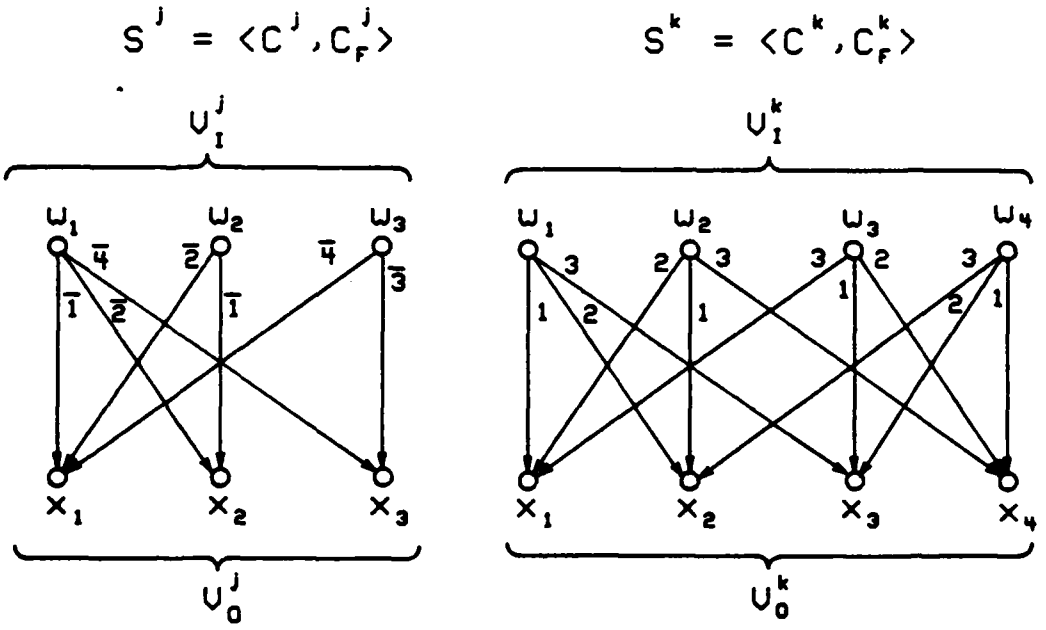


Figure 5.5:
Systems $S^j = \langle C^j, C_F^j \rangle$, $S^k = \langle C^k, C_F^k \rangle$ for Example 5.7.4.

Suppose there is a quasimorphism $\bar{\psi}^1$ such that $\bar{\psi}^1(S^1) = S^2$ and $\bar{\psi}^1(S^3) = S^2$ and $\bar{\psi}^1$ is 1:1. First, this means that $S^1 = S^3$ since $\bar{\psi}^1$ is 1:1. Second, and more important from an engineering point of view, the 1:1 guarantees an efficient emulation of S^1 by S^2. That is, if all V_I were connected to processors and V_O to memories, the emulation would be such that the processing work of one processor in S^1 would be exactly equal to the processing work of one processor in the image of S^1 in S^2. Also, the amount of data stored in a single memory unit in S^1 would be exactly equal to the amount of data stored in memory unit in the image of S^1 in S^2. In other words, the mapping is regular in some sense. Analogously, the load balancing and utilization in the image of S^1 in S^2 will be identical to that in S^1. The quasimorphism can be used to map multiple copies of system S^1 into S^2, where $\bar{\psi}^1(S^1) \cap \bar{\psi}^2(S^1) = \emptyset$ is a necessary additional constraint. This will allow tandem cross checking of partial results of a computation and therefore can be used as an error detection mechanism for fault tolerance.

In order to evaluate the efficiency and uniformity of the emulation the following criteria will be used.

Definition 5.7.5:

Let $S^{1,i} \in S[V_I^1 \times V_O^1]$ and $S^{2,j} \in S[V_I^2 \times V_O^2]$ be two systems. Let $\bar{\psi}$ be a quasimorphism such that $\bar{\psi}(S^{1,i}) \subseteq_{(a,b,c)} S^{2,k}$. Define:

input node factor: $inf \triangleq \max\{|\ \phi_I^{-1}(u_a)\ |\ \ \exists: u_a \in \phi_I(V_I^1)\}$.

output node factor: $onf \triangleq \max\{|\ \phi_O^{-1}(u_a)\ |\ \ \exists: u_a \in \phi_O(V_O^1)\}$.

side effect: $se \triangleq$ yes iff $\exists\ C_n^{1,i} \in C^{1,i} \exists \forall C_m^{2,j}\ \ \bar{\mu}(C_n^{1,i}) \subset C_m^{2,j}$.

For the detail meaning of these factors see the conclusion of this chapter.

Let $S^{1,i} \in S[V_I^1 \times V_O^1]$ and $S^{2,j} \in S[V_I^2 \times V_O^2]$ be two systems. Let $\bar{\psi}$ be a quasimorphism such that $\bar{\psi}(S^{1,i}) \subseteq_{(a,b,c)} S^{2,k}$ or $\bar{\psi}(S^{1,i}) = S^{2,k}$. The comparison of efficiency of different types of emulation is shown in Table 5.1.

5.8 Conclusions

In this chapter several problems have been discussed. The problem of comparison of topologically arbitrary systems was rigidly formulated and analyzed using a new concept called quasimorphism. Each system is defined over an underlying set $V_I \times V_O$. The set of all systems over the underlying over the underlying $V_I \times V_O$ is called the S-set over $V_I \times V_O$. Then the the problem of comparison of systems can be formulated as finding relationships between two S-sets. The problem is very complex and therefore was broken down into two major steps. First the T-set over $V_I \times V_O$ was defined. T-set has less constraints than the S-set over the same $V_I \times V_O$ and therefore it is easier to analyze relationships between T-sets than between S-sets. Auxiliary maps ϕ_I, ϕ_O, $\phi_{I\times O}$, μ, and ψ were defined and it was shown that ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O uniquely determine ψ-map ψ. Conversely, ψ-map ψ uniquely determines ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O. Informally, ψ-map ψ is measure of similarity between T-elements. It was shown that certain properties of ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O are inherited by ψ-map ψ. In particular if ϕ_Γ-map ϕ_I and ϕ_O-map ϕ_O are 1:1 maps then so is ψ-map ψ. Conversely if ψ-map ψ is 1:1

Table 5.1:
Comparison of efficiency of different types of emulation.

A. $\bar{\psi}$ not 1:1; $S^{1,i}$, $S^{2,k}$

	$\subseteq$a, not b	$\subseteq$b, not c	$\subseteq$c, not =	=
inf	>1	>1	>1	>1
onf	>1	>1	>1	>1
se	YES	YES	NO	NO

B. $\bar{\psi}$ 1:1; $S^{1,i}$, $S^{2,k}$

	$\subseteq$a, not b	$\subseteq$b, not c	$\subseteq$c, not =	=
inf	1	1	1	1
onf	1	1	1	1
se	YES	YES	NO	NO

then also ϕ_T-map ϕ_I and ϕ_O-map ϕ_O are 1:1 maps.

In the next section, the relationships between two S-sets were studied. Using the maps ϕ_I, ϕ_O, $\phi_{I\times O}$, μ, and ψ in the T-set domain, new correspondences $\bar{\mu}$ and $\bar{\psi}$ were defined in the S-set domain. Informally, $\bar{\psi}$ is a measure of similarity between two S-sets. As expected and intended, some behavior of ϕ_I, ϕ_O, $\phi_{I\times O}$, μ, and ψ was inherited by $\bar{\mu}$-correspondence $\bar{\mu}$ and $\bar{\psi}$-correspondence $\bar{\psi}$. For example if ϕ_T-map ϕ_I, and ϕ_O-map ϕ_O are 1:1 maps then $\bar{\psi}$-correspondence $\bar{\psi}$ is 1:1 correspondence. Conversely if $\bar{\psi}$-correspondence $\bar{\psi}$ is 1:1 then ϕ_T-map ϕ_I and ϕ_O-map ϕ_O are 1:1 maps. Properties of $\bar{\psi}$-correspondence $\bar{\psi}$ similar to the reflexive, symmetric, and transitive properties of relations were discussed, in particular the following were shown.

Let S^1, S^2, and S^3 be three systems. The quasimorphism has the following properties.

(1) $\exists\ \bar{\psi}$ such that $\bar{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2 \not\rightarrow \exists\ \bar{\psi}^2$ such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$ and $\bar{\psi}^2(S^2) = S^3 \rightarrow \exists\ \bar{\psi}$, $\bar{\psi}(S^1) = S^3$.

Let S^1, S^2, and S^3 be three systems. The quasimorphism 1:1 has the following properties.

(1) $\exists\ \bar{\psi}$, 1:1 such that $\bar{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2$, 1:1 $\rightarrow \exists\ \bar{\psi}^2$, 1:1 such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$, 1:1 and $\bar{\psi}^2(S^2) = S^3$, 1:1 $\rightarrow \exists\ \bar{\psi}$, 1:1, $\bar{\psi}(S^1) = S^3$.

The quasimorphism measure provides the necessary theoretical background for studying the following problems of parallel processing.

(a) Emulation of system S^1 by system S^2.

(b) Fault tolerance method achieved by a concurrent execution of multiple copies of the same problem.

(c) Partitioning of a system.

Three types of emulation were defined based upon the subsystem relationship between the image of the emulated system and the host system. Several measures of efficiency of the emulation based upon the preservation of the computational loading and other factors were defined and the emulation types were evaluated on that basis. Suppose the system S^1 consists of processors connected to the V_I^1 and memory units connected to V_O^1. If the input node factor = 1, than the amount of computation performed in the host system node $\phi_I(u_a) \in V_I^2$ is the same as the amount of computation performed in the node $u_a \in V_I^1$. If inf > 1, that means $\exists\ u_a, u_b \in V_I^1$ and $w_a \in V_I^2$ such that $\phi_I(u_a) = w_a$ and $\phi_I(u_b) = w_a$. That implies the processor connected to w_a in $\bar{\psi}(S^1)$ must perform the computation of the processors connected to the nodes u_a and u_b in S^1. If the output node factor = 1, than the amount of data stored in the memory unit connected to $\phi_O(v_a) \in V_O^2$ is the same as in the memory unit connected to $v_a \in V_O^1$ in S^1. If onf > 1, than $\exists\ v_a, v_b \in V_O^1$ and $x_a \in V_O^2$ such that $\phi_O(v_a) = \phi_O(v_b) = x_a$. That implies the memory unit connected to x_a in $\bar{\psi}(S^1)$, must contain the data contained in both memory units connected to v_a and v_b in S^1. Side effects exist if the correspondence $C_m^{2,j} \in C^2$ that is used to emulate the correspondence $C_n^{1,i} \in C^1$ has the property $\bar{\mu}(C_n^{1,i}) \subset C_m^{2,j}$. This causes $C_m^{2,j}$ to move some additional data that the $C_n^{1,i}$ did not move.

6 SINGLE STAGE NETWORKS - ANALYSIS

6.1 Introduction

In this chapter, the horizontal composition and decomposition of single stage interconnection networks will be analyzed [SeS85]. The general model of interconnection networks, defined in earlier, will be used in the analysis.

Using the horizontal composition/decomposition the partitionability property of interconnection networks will be defined. Informally the partitionability property means that the network can be divided into several parts each of which has certain degree of independence. The type of partitionability analyzed in this chapter uses all the states for consideration of partitionability and has three subtypes.

An algorithm is developed which will output one of the following:

(1) The network is not partitionable.

(2) The network is partitionable into subnetworks with common control signals and the combination of the of the subnetworks will exactly generate all interconnection patterns of the original network.

(3) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will exactly generate all interconnection patterns of the original network.

(4) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will generate a superset of interconnection patterns of the original network.

The algorithm is network topology independent and can be used to analyze topologically regular and irregular single stage networks.

The partitionability property of interconnection networks for parallel computer systems is important for the following reasons.

(1) If the network is partitionable than the system can on demand easily allocate only a subset of total resources. This can be used in several different ways as shown below.

   (a) A user can use only a small part of the machine for program development.

   (b) In a multiple user environment the partitioning provides a natural protection among users.

   (c) In a multitasking environment the partitioning provides a protection among independent tasks.

(2) If the network is partitionable the fault tolerance of the system increases as follows.

   (a) A method of graceful degradation is possible by separating the faulty section from the correctly operating ones.

   (b) If in addition to being a partitionable network, the sections are isomorphic, then an increase of reliability may be realized by multiple mappings of the same task onto the multiple sections and tandem cross checking of partial results.

   (c) It is possible to construct a fault tolerant network using a partitionable network as a core.

(3) If the network is partitionable, then there is an efficient implementation in terms of hardware and control. The network can be implemented as a set of network components each with its own set of inputs and outputs. The data path layout and under some conditions also the

controls layout is simplified on VLSI substrate or on a printed circuit board (PCB).

6.2 Overview

In Section 6.3 the problem discussed in this chapter is informally defined. In Section 6.4 the previous work on partitionability is briefly described. In Section 6.5 some basic concepts are defined. In Section 6.6 the horizontal composition and decomposition of single stage interconnection networks are formally defined and analyzed. In Section 6.7 an algorithm is presented and proven for correctness that accepts as an input a topologically arbitrary interconnection network and outputs one of following four outcomes. The network is not partitionable, or the network is partitionable in one of the three types.

6.3 Problem Statement

In this section the problem of partitionability of single stage interconnection networks will be analyzed [SeS85]. There is a large amount of work done on this subject for certain class of interconnection networks, namely

topologically regular networks [Gok76, GoL73, Sie80, Upp81]. The work here is different in two respects from the previous studies. First, the topology under discussion here is completely unrestricted and the results apply to the regular as well as irregular interconnection networks. Second, the set of states used in the consideration of partitionability here includes all the states of the networks, whereas the previous work used only a subset of the states (this will be discussed more in the future chapters.) In our work the partitionability will be defined and three different types of partitionability will be recognized. Then an algorithm which accepts as an input a topologically arbitrary interconnection network and outputs one of the four possible outputs will be presented. The outputs are as follows: (a) the network is not partitionable, (b) the network is partitionable into two networks with dependent controls, (c) the network is partitionable into two networks with independent controls where the combination produces the original network exactly, and (d) the network is partitionable into two networks with independent controls where the combination produces a superset of states of the original network.

6.4 Previous Work

The partitionability of topologically regular network has been studied extensively in the literature. It was shown in [Sie80] that single stage and multistage Cube networks are partitionable, as are PM2I and ADM. It was also shown in [Sie80] that the Illiac and Shuffle-Exchange are not partitionable.

The analysis in [Sie80] was based upon the cycle structure of the permutations admissible by the network under analysis. In [Upp81] the partitionability of regular SW banyans was discussed, and in [Gok76, GoL73] the partitionability of banyans networks was shown. All these networks are topologically regular and partitionability of arbitrary networks was not studied in the literature. The partitionability discussed in this chapter is different from the type discussed in the previous work in two respects: it considers the participation of all the states of the network, where the type studied previously considered only a subset of the states of the network, and it is applicable to networks with arbitrary, regular and irregular topology.

6.5 Basic Concepts

In this section the basic concepts are presented. These definition can be found in a text on graph theory [BoM76] and are included here for completeness only.

Definition 6.5.1:

Let V be a set of labels. Let $E \subseteq V \times V$, then $G = \langle V,E \rangle$ is called a *graph.*

6.6 Composition and Decomposition of Networks

This section describes a "horizontal" composition and decomposition of single stage networks. The discussion here is presented for the composition of two networks into one and the decomposition of one network into two. However, it can be generalized into the composition of n networks into one and decomposition of one network into n, $n > 2$. What is meant by the *horizontal composition* of two networks K^1 and K^2 is that $V_I^1 \cap V_I^2 = \emptyset$ and $V_O^1 \cap V_O^2 = \emptyset$. Similarly, the *horizontal decomposition* of K into two networks K^1 and K^2 will result in $V_I^1 \cap V_I^2 = \emptyset$ and $V_O^1 \cap V_O^2 = \emptyset$. Two types of composition (decomposition) are described. One, the *σ-composition (decomposition)* corresponds to the physical situation where the controls of the individual subnetworks of the network are independent. The other type is the *τ-composition (decomposition),* which corresponds to the physical situation where the controls of the individual subnetworks of the network are dependent upon one another.

This section conceptually consists of two parts. In part one the definition of the σ-composition is given and some of its basic properties are presented. In part two the definition of the τ-composition is given and its properties are described.

Definition 6.6.1:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that: $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset$. Define *$\sigma$-map* as follows: $K^1 \sigma K^2 = <C^1> \sigma <C^2>$

$\triangleq$ $<\{C_p^1 \cup C_r^2 \mid C_p^1 \in C^1, C_r^2 \in C^2\}>$.

This describes the composition of two networks where the controls of the two networks are independent from one another. The Lemmas and Theorems 6.6.2 to 6.6.4 discuss the properties of the σ-map composition of networks.

Lemma 6.6.2:

Let $K^1 \in K[V_I^1 \times V_O^1]$ and $K^2 \in K[V_I^2 \times V_O^2]$ be two networks such that: $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset$. Then $K^1 \sigma K^2 = K^2 \sigma K^1$.

Proof:

Obvious from the definition of σ-map and commutativity property of set union.

□

Theorem 6.6.3:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that: $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset$. Then $K^1 \sigma K^2 \in K[(V_I^1 \cup V_I^2) \times (V_O^1 \cup V_O^2)]$.

Proof:

(1): Let $\{C_p^1 \cup C_r^2 \mid C_p^1 \in C^1, C_r^2 \in C^2\} = C^3$, let $C_m^3 \in C^3$. Let $C[(V_I^1 \cup V_I^2) \times (V_O^1 \cup V_O^2)] = C^*$.

(2): Show $C^3 \subseteq C^*$.

(2.1): Clearly $C_m^3 \in P[(V_I^1 \cup V_I^2) \times a \ (V_O^1 \cup V_O^2)]$. Must show nondestructivity.

(2.1.1): $<u_a,u_b>$, $<u_c,u_d> \in C_m^3 \rightarrow$ three cases.

(2.1.2): $\langle u_a,u_b\rangle,\ \langle u_c,u_d\rangle \in C_p^1,\ C_p^1 \in C^1 \rightarrow u_b \neq u_d$.

(2.1.3): $\langle u_a,u_b\rangle,\ \langle u_c,u_d\rangle \in C_r^2,\ C_r^2 \in C^2 \rightarrow u_b \neq u_d$.

(2.1.4): $\langle u_a,u_b\rangle \in C_p^1,\ C_p^1 \in C^1$ and $\langle u_c,u_d\rangle \in C_r^2,\ C_r^2 \in C^2$. $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset \rightarrow V_O^1 \cap V_O^2 = \emptyset \rightarrow u_b \neq u_d$.

(2.1.5): (2.1.2), (2.1.3), and (2.1.4) $\rightarrow C_m^3 \in C^* \rightarrow C^3 \subseteq C^*$.

(3): Show $s(C^3) = V_I^1 \cup V_I^2$.

(3.1): $s(C^3) = s(\{C_p^1 \cup C_r^2 \mid C_p^1 \in C^1,\ C_r^2 \in C^2\}) = \{s(C_p^1) \cup s(C_r^2) \mid C_p^1 \in C^1, C_r^2 \in C^2\} = \{s(C_p^1) \mid C_p^1 \in C^1\} \cup \{s(C_r^2) \mid C_r^2 \in C^2\} = s(C^1) \cup s(C^2) = V_I^1 \cup V_I^2$.

(4): Show $d(C^3) = V_O^1 \cup V_O^2$.

(4.1): Similar to (3.1) except replace the s set by the d set.

(5): Show $|C^3| \geq 2$.

(5.1): $|C^3| = |\{C_p^1 \cup C_r^2 \ni C_p^1 \in C^1, C_r^2 \in C^2\}|$.

(5.2): $C_p^1 \cup C_r^2 \neq C_s^1 \cup C_t^2$, $p \neq s$ or $r \neq t \rightarrow$ all C_m^3 are distinct.

(5.3): (5.1), (5.2) $\rightarrow |C^3| = |C^1| \cdot |C^2| \geq 2 \cdot 2 = 4$.

□

Lemma 6.6.4:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^2 \in K[V_I^2 \times V_O^2]$, and $K^3 \in K[V_I^3 \times V_O^3]$ be three networks such that $(V_I^a \cup V_O^a) \cap (V_I^b \cup V_O^b) = \emptyset$, $a \neq b$, $a,b = 1,2,3$, then $(K^1 \sigma K^2) \sigma K^3 = K^1 \sigma (K^2 \sigma K^3)$.

Proof:

Obvious from the definition of σ-map and the associativity property of set union.

□

The next three definitions are introducing the technical nomenclature used in this chapter.

Definition 6.6.5:

Let $K \in K[V_I \times V_O]$ be a network. Let $\{K^1, K^2, \ldots, K^n \mid K^i \in K[V_I^i \times V_O^i]\}$ be a set of networks such that: $K = K^1 \sigma K^2 \sigma \cdots K^n$. Then

(1) $K^1 \sigma K^2 \sigma \cdots K^n$ is called a *σ-decomposition of K.*

(2) $\{K^1, K^2, \ldots, K^n\}$ is called a *σ-decomposition set of K.*

(3) K^i is called a *σ-decomposition element of K.*

(4) K is the *σ-composition* of $K^1 \sigma K^2 \sigma \cdots K^n$.

Definition 6.6.6:

Let $K \in K[V_I \times V_O]$ be a network. If the only possible σ-decomposition is $K = K^1$ then K is called a *σ-prime network.*

Definition 6.6.7:

Let $K \in K[V_I \times V_O]$ be a network and let $K = K^1$. Then K^1 is called the *trivial σ-decomposition of K.*

Lemma 6.6.8:

Let $K \in K[V_I \times V_O]$ be a network. Then K has a σ-decomposition.

Proof:

Let $K = K^1$ be the trivial σ-decomposition of K.

□

Definition 6.6.9:

Let $K \in K[V_I \times V_O]$ be a network. Let $K = K^1 \sigma K^2 \sigma \cdots K^n$ be a σ-composition, where $\forall j$, K^j is a σ-prime network. Then $K^1 \sigma K^2 \sigma \cdots K^n$ is called a *σ-composition prime of K.*

Notice that this implies $V_I = \bigcup_{i=1}^{n} V_I^i$ and $V_O = \bigcup_{i=1}^{n} V_O^i$.

Theorems 6.6.10 to 6.6.12 discuss some properties of the σ-decomposition of networks.

Theorem 6.6.10:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$, be a network. Let $K = K^1 \sigma K^2 \sigma \cdots K^n$ be any σ-decomposition. Then: $n \leq \log_2 |C|$.

Proof:

Let $K^j = \langle C^j \rangle$.

(1): K^j is a network $\rightarrow |C^j| \geq 2$.

(2): $|C| = \prod_{j=1}^{n} |C^j| \geq 2^n$.

(3): $n = \log_2 2^n \leq \log_2 |C|$.

□

This can be used as an upper bound on number of networks in a σ-decomposition set.

Theorem 6.6.11:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$, be a network.

(1) If K has a nontrivial σ-decomposition then $|C|$ is not a prime number.

(2) If $|C|$ is a prime number then K does not have a nontrivial σ-decomposition.

Proof:

Follows from proof of Theorem 6.6.10.

□

This counting principle introduced above can be used as a necessary condition on a σ-decomposition of a network.

Theorem 6.6.12:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = \langle C^1 \rangle$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = \langle C^2 \rangle$, be two networks such that: $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset$. Let $K^3 = K^1 \sigma K^2$ be a σ-composition.

(1) If $\emptyset_C \in C^2$ then $K^1 \subseteq_c K^3$ where $\emptyset_C$ is the correspondence consisting of no edges, i.e., no connections between the set of inputs and the set of outputs.

(2) If $\emptyset_C \notin C^2$ then $K^1 \subseteq_b K^3$, but not $K^1 \subseteq_c K^3$.

(3) If $\emptyset_C \in C^1$ then $K^2 \subseteq_c K^3$.

(4) If $\emptyset_C \notin C^1$ then $K^2 \subseteq_b K^3$, but not $K^2 \subseteq_c K^3$.

Proof:

Case 1: Show $K^1 \subseteq_c K^3$.

(1): $K^3 = K^1 \sigma K^2 \rightarrow \quad \forall C_m^1 \in C^1, \quad \forall C_n^2 \in C^2 \quad \exists C_p^3 \in C^3$ $\ni: C_p^3 = C_m^1 \cup C_n^2$.

(2): (1) and $\emptyset_C \in C^2 \rightarrow \forall C_m^1 \in C^1 \quad \exists C_p^3 \in C^3 \quad \ni: C_m^1 = C_p^3$ $\rightarrow K^1 \subseteq_c K^3$.

Case 2: Show $K^1 \subseteq_b K^3$ but not $K^1 \subseteq_c K^3$.

(1): Same as Case 1.

(2): (1) and $\emptyset_C \notin C^2 \rightarrow (\forall C_m^1 \in C^1 \; \exists C_p^3 \in C^3 \; \ni: C_m^1 \subset C_p^3)$ and $(\forall C_m^1 \in C^1 \; \nexists C_p^3 \in C^3 \; \ni: C_m^1 = C_p^3) \rightarrow K^1 \subseteq_b K^3$ and not $K^1 \subseteq_c K^3$.

Case 3 and 4: Same as Case 1 and 2 by the commutativity of the σ-composition (Lemma 6.6.2).

□

In this second part of this section, the τ-composition and decomposition of two networks will be discussed. This differs from the σ-composition (decomposition) as follows. In the σ-composition, the two networks keep independent controls, that is if C_m^1 is selected in K^1, an arbitrary correspondence C_n^2 can be selected in K^2. In the τ-composition, the two networks have joint control, that is if C_m^1 is selected in K^1, the corresponding C_n^2 must be selected in K^2.

Definition 6.6.13:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$, be two networks such that:

(a) $(V_I^1 \cup V_O^1) \cap (V_I^2 \cup V_O^2) = \emptyset$, and (b) $|C^1| = |C^2|$.

Define τ_α *-map* as follows:

(1) Define α: $C^1 \rightarrow C^2$, map 1:1 and onto.

(2) $K^1 \tau_\alpha K^2 = <C^1> \tau_\alpha <C^2> \triangleq <\{C_p^1 \cup C_r^2 \mid \alpha(C_p^1) = C_r^2, C_p^1 \in C^1, C_r^2 \in C^2\}>$.

This describes the composition of two networks where the controls are dependent in the sense that choosing a C_p^1 in C^1 means $\alpha(C_p^1)$ must be selected in C^2. Thus, the α map exactly specifies how the controls are dependent. The basic difference between the σ-map and τ_α-map is as follows. Suppose $K^1 = <C^1>$ and $K^2 = <C^2>$. If $K^3 = K^1 \sigma K^2$, $K^3 = <C^3>$, then (a) $|C^3| = |C^1| \cdot |C^2|$ and (b) C_p^1 is a subset of $|C^2|$ correspondences in C^3. If $K^3 = K^1 \tau_\alpha K^2$ then (a) $|C^3| = |C^1| = |C^2|$ and (b) C_p^1 is a subset of one correspondence in C^3, specifically $C_p^1 \cup \alpha(C_p^1)$.

This describes the composition of two networks where the controls are dependent in the sense that choosing a C_p^1 in C^1 means $\alpha(C_p^1)$ must be selected in C^2. Thus, the α map exactly specifies how the controls are dependent. The basic difference between the σ-map and τ_α-map is as follows. Suppose $K^1 = <C^1>$ and $K^2 = <C^2>$.

If $K^3 = K^1 \sigma K^2$, $K^3 = <C^3>$, then

(a) $|C^3| = |C^1| \cdot |C^2|$ and

(b) C_p^1 is a subset of $|C^2|$ correspondences in C^3.

If $K^3 = K^1 \tau_\alpha K^2$ then

(a) $|C^3| = |C^1| = |C^2|$ and

(b) C_p^1 is a subset of one correspondence in C^3, specifically $C_p^1 \cup \alpha(C_p^1)$.

The definitions 6.6.14 to 6.6.16 are providing some nomenclature involving the τ-composition and decomposition of networks.

Definition 6.6.14:

Let $K \in K[V_I \times V_O]$ be a network. Let $\{K^1, K^2, \ldots, K^n \mid K^i \in K[V_I^i \times V_O^i]\}$ be a set of networks such that: $K = K^1 \tau_\alpha K^2 \tau_\alpha \cdots K^n$. Then

(1) $K^1 \tau_\alpha K^2 \tau_\alpha \cdots K^n$ is called a *τ-decomposition of K.*

(2) $\{K^1, K^2, \ldots, K^n\}$ is called a *τ-decomposition set of K.*

(3) K^i is called a *τ-decomposition element of K.*

(4) K is the *τ-composition* of $K^1 \tau_\alpha K^2 \tau_\alpha \cdots K^n$.

Definition 6.6.15:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$, be a network. K is a prime network iff K cannot be decomposed as $K \supseteq c\ K^1 \sigma K^2$.

Definition 6.6.16:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$, be a network. If there exist $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = \langle C^1 \rangle$, and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = \langle C^2 \rangle$, two prime networks such that: (1) $V_I^1 \cup V_I^2 = V_I$, and (2) $V_O^1 \cup V_O^2 = V_O$, then:

(1) If $K^1 \tau_\alpha K^2 = K$, then K is a *τ-partitionable* network.

(2) If $K^1 \sigma K^2 = K$, then K is a *strictly σ-partitionable* network.

(3) If $K^1 \sigma K^2 \neq K$ and $K^1 \sigma K^2 \supseteq c\ K$, then K is a *$\sigma$-partitionable* network.

Note that strictly σ-partitionable implies: $|C| = |C^1| \cdot |C^2|$ and $C = \{C_s^1 \cup C_t^2 | \ C_s^1 \in C^1, \ C_t^2 \in C^2\}$. In contrast σ-partitionable implies: $|C| < |C^1| \cdot |C^2|$ and $C \subset \{ C_s^1 \cup C_t^1 | \ C_s^1 \in C^1, C_t^2 \in C^2\}$.

If K is a τ-partitionable network then it is also a σ-partitionable. It is not strictly σ-partitionable because *it is strictly σ-partitionable only if* $|C^1| \cdot |C^2| = |C|$ and it is τ-partitionable only if $|C^1| = |C^2| = |C|$, which implies $|C^1| = |C^2| = |C| = 1$; however, $|C^1|, |C^2|, |C| \geq 2$, by Definition 4.6.1. Also note that if there exists a σ-prime composition of K, then K is a strictly σ-partitionable network.

6.7 Partitionability Algorithm

In this section an algorithm is presented that has an input any general network (with an arbitrary topological structure) and which produces one of four possible outputs.

(1) The network is not partitionable.

(2) The network is τ-partitionable.

(3) The network is strictly σ-partitionable.

(4) The network is σ-partitionable .

The engineering interpretation of the four outputs is as follows:

(1) The network is not partitionable into disjoint subnetworks.

(2) The network is partitionable into subnetworks with common control signals that are dependent upon one another and the combination of the subnetworks will exactly generate all interconnection patterns of the original network.

(3) The network is partitionable into subnetworks with independent control signals and the combination of the subnetworks will exactly generate all interconnection patterns of the original network.

(4) The network is partitionable into subnetworks with independent control signals and the combination of the subnetworks will generate a superset of interconnection patterns of the original network.

The algorithm can be programmed on a computer and if the output of the algorithm is (2) or (3) then it will produce a more efficient implementation of the network in terms of data path hardware and possibly control implementation. In case (4), even though a superset of the states of the original network is obtained, the implementation produced by the algorithm will be efficient in most instances. The following definitions are needed to discuss the algorithm and prove its correctness.

Definition 6.7.1:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $C_m \in C$ and $\langle v_a, v_b \rangle \in C_m$ be an edge (directed). Denote the undirected arc associated with the directed edge of $\langle v_a, v_b \rangle$ by $\langle \overline{v_a, v_b} \rangle$. Let $G[V_I \times V_O] \triangleq \{\langle \overline{v_a, v_b} \rangle \mid \langle v_a, v_b \rangle \in C_m, \ \forall C_m \in C\}$. Then $G[V_I \times V_O]$ is the *underlying undirected graph of K.*

Definition 6.7.2:

Let $G[V_I \times V_O]$ be the underlying undirected graph of $K \in K[V_I \times V_O]$. Then the connected subgraphs of $G[V_I \times V_O]$ are called *components of* $G[V_I \times V_O]$.

Notation: Components are denoted by $B^1, B^2, \ldots, B^n$. Denote the vertices associated with B^r by V_I^r and V_O^r, $V_I^r \subseteq V_I$, $V_O^r \subseteq V_O$. In a component B^r there exists a path from each node to every other node and there is no path between any two nodes from different components. Clearly $G[V_I \times V_O] = \bigcup_r B^r$, $\bigcup_r V_I^r = V_I$, and $\bigcup_r V_O^r = V_O$.

Definition 6.7.3:

Let $G[V_I \times V_O]$ be the underlying graph of $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $C_m \in C$ and let B^r be a component of $G[V_I \times V_O]$. Define the *projection p of* C_m *onto* B^r as follows:

$p(C_m,B^r) \triangleq \{\langle v_a,v_b\rangle \in C_m \mid \langle \overline{v_a,v_b}\rangle \in B^r\}$.

Lemma 6.7.4:

Let $G[V_I \times V_O]$ be the underlying graph of $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $C_m \in C$ and let $\{B^1,B^2,\ldots,B^n\}$ be the set of all components of $G[V_I \times V_O]$. Then $C_m = p(C_m,B^1) \cup p(C_m,B^2) \cup \cdots p(C_m,B^n)$.

Proof:

(1): Show $p(C_m,B^i) \cap p(C_m,B^j) \neq \emptyset \rightarrow B^i = B^j$.

(1.1): $p(C_m,B^i) \cap p(C_m,B^j) \neq \emptyset \rightarrow$

$\langle v_a,v_b\rangle \in p(C_m,B^i)$, $\langle v_a,v_b\rangle \in p(C_m,B^j)$.

(1.2): $\langle v_a,v_b\rangle \in p(C_m,B^i) \rightarrow \langle v_a,v_b\rangle \in C_m$, $\langle \overline{v_a,v_b}\rangle \in B^i$.

(1.3): $\langle v_a,v_b\rangle \in p(C_m,B^j) \rightarrow \langle v_a,v_b\rangle \in C_m,\ \langle \overline{v_a,v_b}\rangle \in B^j.$

(1.4): $\langle \overline{v_a,v_b}\rangle \in B^i,\ \langle \overline{v_a,v_b}\rangle \in B^j$ and $G[V_I \times V_O] = \bigcup_r B^r$

$\rightarrow B^i = B^j.$

(2): Show $C_m = \bigcup_i p(C_m,B^i).$

(2.1): Show $C_m \subseteq \bigcup_i p(C_m,B^i).$

(2.1.1): $\langle v_a,v_b\rangle \in C_m \rightarrow \langle \overline{v_a,v_b}\rangle \in G[V_I \times V_O] \rightarrow$
$\exists\ B^j,\ \langle \overline{v_a,v_b}\rangle \in B^j \rightarrow \langle v_a,v_b\rangle \in p(C_m,B^j) \rightarrow$
$\langle v_a,v_b\rangle \in \bigcup_i p(C_m,B^i).$

(2.2): Show $C_m \supseteq \bigcup_i p(C_m,B^i).$

(2.2.1): $\langle v_a,v_b\rangle \in \bigcup_i p(C_m,B^i) \rightarrow$

$\exists\ B^r,\ \langle v_a,v_b\rangle \in p(C_m,B^r) \rightarrow \langle v_a,v_b\rangle \in C_m.$

(3): (1) and (2) $\rightarrow C_m = \bigcup_i p(C_m,B^i).$

□

Definition 6.7.5:

Let $G[V_I \times V_O]$ be the underlying undirected graph of $K \in K[V_I \times V_O]$, $K = \langle C\rangle$. Let B^i be a component of $G[V_I \times V_O]$. Define the ***residue set modulo*** B^i as follows: $r(B^i) \triangleq \{p(C_b,B^i) \mid \forall C_b \in C\}$.

The Theorems and Lemmas 6.7.6 to 6.7.13 are essential components of the proof of the algorithm presented later. They discuss the conditions of existence and properties of the component networks, which are the parts into which a network is decomposed if a decomposition exist.

Theorem 6.7.6:

Let B^r be a component of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $r(B^r)$ be the residue set modulo B^r, B^r over $V_I^r \times V_O^r$. If $|r(B^r)| \geq 2$ then $\langle r(B^r) \rangle \in K[V_I^r \times V_O^r]$. $\langle r(B^r) \rangle$ is called a *component network of K* denoted by $K(B^r)$.

Proof:

(1): Show $\hat{C}_s \in r(B^r) \rightarrow \hat{C}_s \in C[V_I^r \times V_O^r]$.

(1.1): $\hat{C}_s \in r(B^r) = \{p(C_a,B^r) \ni: C_a \in C\} \rightarrow$

$\exists\ C_x \in C,\ \hat{C}_s = p(C_x,B^r) \rightarrow \hat{C}_s \in C[V_I^r \times V_O^r]$.

(2): Show $s(r(B^r)) = V_I^r$.

(2.1): Show $s(\{p(C_a,B^r) \ni: C_a \in C\}) \subseteq V_I^r$.

$u_a \in s(\{p(C_a,B^r) \ni: C_a \in C\}) \rightarrow \exists\ C_b \in C, \langle u_a,u_b \rangle \in C_b,$

$\langle \overline{u_a,u_b} \rangle \in B^r \rightarrow u_a \in V_I^r$.

(2.2): Show $s(\{p(C_a,B^r) \ni: C_a \in C\}) \supseteq V_I^r$.

$u_a \in V_I^r \rightarrow \langle \overline{u_a,u_b} \rangle \in B^r \rightarrow \exists\ C_b \in C, \langle u_a,u_b \rangle \in C_b \rightarrow$

$\langle u_a,u_b \rangle \in p(C_b,B^r) \rightarrow$

$u_a \in s(\{p(C_a,B^r) \ni: C_a \in C\}) = s(r(B^r))$.

(2.3): (2.1), (2.2) $\rightarrow s(r(B^r)) = V_I^r$.

(3): Show $d(r(B^r)) = V_O^r$.

Same as (2) except replace the s set by the d set.

(4): Show $|r(B^r)| \geq 2$.

By Theorem hypothesis.

(5): (1), (2), (3) and (4) $\rightarrow \langle r(B^r) \rangle \in K[V_I^r \times V_O^r]$.

□

Given an arbitrary network it is possible that $|r(B^r)| = 1$ for some B^r; that is, $p(C_a,B^r) = p(C_b,B^r)$, $\forall C_a,C_b \in C$. Then $r(B^r)$ does not constitute a reconfigurable network as defined. To handle this case from an engineering point of view, do the following. If a network contains a such B^r, that part of the network is constant, that is, it has a single state only. So to remove this constant part from the network $K = <C>$ do the following. (1) Construct separately the constant part $r(B^r)$, $\forall B^r \ni: |r(B^r)| = 1$, as a set of nonreconfigurable links. (2) $K' \triangleq <\{C_m - <v_a,v_b> \mid C_m \in C,$ $\forall <\overline{v_a,v_b}> \in B^r, \quad \forall B^r \ni: |r(B^r)| = 1\}>$. K' then contains only the block (blocks) where $|r(B^r)| > 1$. In the following it is assumed that the constant blocks of the network have been removed already.

If $G[V_I \times V_O] = B^1$, then $K = <r(B^1)>$. In this case, K is a σ-prime network and is not partitionable. The following Lemmas and Theorems are shown for the case of $G[V_I \times V_O]$ having two components, B^1 and B^2, for reasons of simplicity. They are all applicable to the case of $B^1,B^2,...,B^n$, $n \geq 2$.

Lemma 6.7.7:

Let $\{B^1, B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = <C>$. Let $|r(B^i)| = |C|$, $\forall i$. Then $\exists\ \tau_\alpha$ such that if $<C^3> = K(B^1)\ \tau_\alpha\ K(B^2)$ then $C \subseteq C^3$.

Proof:

(1): $|r(B^i)| = |C|$, $\forall i$.

This is necessary and sufficient condition for the existence of α.

$p(C_x,B^r) \neq p(C_y,B^r)$, $\forall C_x,C_y \in C$, $x \neq y$, $\forall r$.

(2): $\langle C^3 \rangle = K(B^1)\ \tau_\alpha\ K(B^2) \rightarrow C^3 = \{p(C_a,B^1) \cup p(C_b,B^2)$
$\mid \alpha(p(C_a,B^1)) = p(C_b,B^2),\ C_a \in C,\ C_b \in C\}$.

(3): Let α: $\{p(C_a,B^1) \mid\ C_a \in C\} \rightarrow \{p(C_b,B^2) \mid C_b \in C\}$,
$\alpha(p(C_a,B^1)) = p(C_a,B^2)$.

(4): $C_s \in C \rightarrow C_s = p(C_s,B^1) \cup p(C_s,B^2)$.

(5): (2), (3) and (4) $\rightarrow C_s \in C^3 \rightarrow C \subseteq C^3$.

□

Lemma 6.7.8:

Let $\{B^1, B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $|r(B^i)| = |C|$, $\forall i$. Then $\exists\ \tau_\alpha$ such that if $\langle C^3 \rangle = K(B^1)\ \tau_\alpha\ K(B^2)$ then $C^3 \subseteq C$.

Proof:

(1): (1), (2), and (3) from proof of Lemma 6.7.7.

(2): $C_x \in C^3 \rightarrow C_x = p(C_x,B^1) \cup p(C_x,B^2)$.

(3): (1) and (2) $\rightarrow C_x \in C^3 \rightarrow C^3 \subseteq C$.

□

Theorem 6.7.9:

Let $\{B^1, B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = \langle C \rangle$. Let $|r(B^i)| = |C|$, $\forall i$. Then $\exists\ \tau_\alpha$ such that $K(B^1)\ \tau_\alpha\ K(B^2) = K$.

Proof:

(1): Let α: $\{p(C_m,B^1)|\ C_m \in C\} \rightarrow \{p(C_n,B^2)|\ C_n \in C\}$, $\alpha(p(C_m,B^1)) = p(C_m,B^2)$. Let $K(B^1)\ \tau_\alpha\ K(B^2) = <C^3>$.

(2): Lemma 6.7.7 $\rightarrow$ $C \subseteq C^3$.

(3): Lemma 6.7.8 $\rightarrow$ $C^3 \subseteq C$.

(4): (2), and (3) $\rightarrow$ $C^3 = C$.

(5): Theorem 4.6.8 $\rightarrow$ $C^3 = C$ $\rightarrow$ $K(B^1)\ \tau_\alpha\ K(B^2) = K$.

□

Lemma 6.7.10:

Let $\{B^1, B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = <C>$. Let $K(B^1)\ \sigma\ K(B^2) = <C^3>$. Then $C \subseteq C^3$.

Proof:

(1): $C_m \in C \rightarrow C_m = p(C_m,B^1) \cup p(C_m,B^2)$.

(2): $<C^3> = K(B^1)\ \sigma\ K(B^2) = <\{p(C_a,B^1)\ |\ C_a \in C\}>\ \sigma\ <\{p(C_b,B^2)|\ C_b \in C\}> \rightarrow C_m \in C^3 \rightarrow C \subseteq C^3$.

□

Theorem 6.7.11:

Let $\{B^1,B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = <C>$. Let $K(B^1)\ \sigma\ K(B^2) = <C^3>$. Let $|r(B^1)| \cdot |r(B^2)| = |C|$. Then $K(B^1)\ \sigma\ K(B^2) = K$.

Proof:

(1): By Lemma 6.7.10 $C \subseteq C^3$.

By Theorem hypothesis $|r(B^1)| \cdot |r(B^2)| = |C^3| = |C| \rightarrow C = C^3$.

(2): By Theorem 4.6.8 and (1) $\rightarrow$ $K(B^1)\ \sigma\ K(B^2) = K$.

□

Theorem 6.7.12:

Let $\{B^1,B^2\}$ be the set of components of the underlying graph $G[V_I \times V_O]$ of $K \in K[V_I \times V_O]$, $K = <C>$. Let $K(B^1)\ \sigma\ K(B^2) = <C^3>$. Let $|r(B^1)| \cdot |r(B^2)| > |C|$. Then $K(B^1)\ \sigma\ K(B^2) \supseteq_c K$ and $K(B^1)\ \sigma\ K(B^2) \neq K$.

Proof:

(1): By Lemma 6.7.10 $C \subseteq C^3$.

Theorem hypothesis $|r(B^1)| \cdot |r(B^2)| = |C^3| > |C| \rightarrow C \subset C^3$.

(2): By Theorem 4.6.7 and (1) $\rightarrow$ $K(B^1)\ \sigma\ K(B^2) \supseteq_c K$, and Theorem 4.6.8 and (1) $\rightarrow$ $K(B^1)\ \sigma\ K(B^2) \neq K$.

□

Definition 6.7.13:

If $B^1,B^2,\ldots,B^n$ are the components of $G[V_I \times V_O]$, where $G[V_I \times V_O]$ is the underlying graph of K, then $K(B^1)$, $K(B^2),\ldots,K(B^n)$ is a *prime decomposition of K.*

The algorithm is presented below. The input is an arbitrary network $K \in K[V_I \times V_O]$, $K = \langle C \rangle$, with the constant part removed. The output is one of (1) K is not partitionable, (2) K is r-partitionable, (3) K is strictly σ-partitionable, (4) K is σ-partitionable. In cases (2), (3), and (4) the algorithm also produces the component networks $K(B^1), K(B^2),\ldots,K(B^n)$, in step (7).

Algorithm :

Input: $K \in K[V_I \times V_O]$, $K = \langle C \rangle$.

Output: (1): K is not partitionable,

or (2): K is r-partitionable,

or (3): K is strictly σ-partitionable,

or (4): K is σ-partitionable.

(1) Construct the underlying graph $G[V_I \times V_O]$ of K.

(2) Find components $B^1,B^2,\ldots,B^n$ of $G[V_I \times V_O]$.

(3) If $(n=1)$ return (1).

(4) Find $p(C_m,B^i)$, $\forall C_m \in C$, $i = 1,2,\ldots,n$.

(5) Find $r(B^i) = \{p(C_m,B^i) \mid \forall C_m \in C\}$, $i = 1,2,\ldots,n$.

(6) Construct $K(B^i) = \langle r(B^i) \rangle$, $i = 1,2,\ldots,n$.

(7) If $(|r(B^r)| = |C|, r = 1,2,\ldots,n)$

then return (2).

(8) If $(\prod_{i=1}^{n} |r(B^i)|) = |C|$

then return (3).

(9) Else return (4).

Proof:

The proof of correctness is directly implied by Theorems 6.7.9, 6.7.11, and 6.7.12.

□

The outputs of the algorithm can be used in the following ways. If the output is "1" (not partitionable), then the system designer will know that the network cannot be divided into individual subnetworks. If the output is "3" (strictly σ-partitionable), then the network can be partitioned and the composition of the component networks will produce a set of correspondences identical to that of the original network. Note that if a network is strictly σ-partitionable it is not τ-partitionable nor σ-partitionable. If the output is "2", the network is τ-partitionable. Any network that is τ-partitionable is also σ-partitionable. However, if a network is τ-partitionable then $|r(B^i)| = |r(B^j)| = |C|$, $1 \leq i, j \leq n$, which is not true in general for a σ-partitionable network. Since $|r(B^i)| = |r(B^j)| = |C|$, $1 \leq i, j \leq n$, the number of correspondences in each component network $<r(B^i)>$ is the same ($|C|$) for i, $1 \leq i \leq n$. This property means that the same control decoders can be used in all network components in a τ-partitionable network. If the output is "4" (σ-partitionable), then the network can be partitioned and the composition of the component networks will produce a set of correspondences that is a superset of that of the original network.

The output of the algorithm applies only to the reconfigurable part of the network because partitionability is defined in terms of a decomposition into "reconfigurable" network components ($|r(B^i)| > 1$). If the original network had some B^i such that $|r(B^i)| = 1$, then those constant component(s) should be

added to the network component(s) generated by the algorithm in order to reproduce the original network.

There are different types of partitionability than the discussed here. For example, study of the partitionability of networks where some of the network correspondences are not used, e.g., as can be done with the cube network was discussed in [Sie80, Sie85].

6.8 Conclusions

In this chapter the interconnection network properties of composition, decomposition, and partitionability were analyzed. The general model of interconnection networks, defined in Chapter 4, was used to describe composition, decomposition, and partitionability properties of networks. The τ- and σ-composition and the τ- and σ-decomposition discussed here are of horizontal type and they are described in detail in the text.

The importance of the partitionability property is described in the introduction. It was found that there actually are many different types of partitionability and the type that uses all states for consideration of partitionability, was discussed in detail here. This type of partitionability consisting of three subtypes, is analyzed in this chapter. The three subtypes the τ-partitionability, σ-partitionability, and strict σ-partitionability were defined and analyzed. An algorithm to determine whether a network is partitionable and if it is which subtype of the three was presented and proven

correct. The algorithm is network topology independent and can be used to analyze topologically regular and irregular interconnection networks.

7 SINGLE STAGE PARTITIONABLE NETWORKS - SYNTHESIS

7.1 Introduction

In this chapter the problem of synthesis of single stage partitionable interconnection networks is analyzed, consequently this chapter may be viewed as an application section of the chapter on analysis. For a designer, the analysis allows an evaluation of networks and their properties [AdS82b, Gok76, Law75, McS82, SeS85], in contrast to the synthesis which provides a construction method for partitionable networks. The body of this chapter consists of two major parts, each of which containing some examples to illuminate the issues.

In the first part, an example of a single stage partitionable network will be presented. Then, an algorithm to generate a large class of single stage partitionable networks will be developed and proven correct. This algorithm is based upon the results presented in the chapter on analysis. For ease of presentation the discussion will presented for the case of networks with $|V_I| = |V_O|$ and with two network components only, however it can easily be generalized to networks where $|V_I| \neq |V_O|$ and to networks with more than two components.

The second part of the body of this chapter discusses the problem of synthesis of a special case of partitionable networks. The special class of networks consists of those networks that are isomorphic to a direct product of groups [Han68, Her75]. Since groups have been studied in abstract algebra extensively, techniques are known to determine the possibility of d .composition of a given group into a direct product of groups. Again, for ease of presentation, the discussion is shown for the direct product of two groups only,

but can be generalized to a product of multiple groups.

7.2 Overview

In Section 7.3 the problem is defined. In Section 7.4 the previous work done is outlined. In Section 7.5 the basic concepts are presented. In Section 7.6 some examples and algorithms to synthesize a large classes of single stage partitionable networks are presented. In addition, a special case of partitionable interconnection networks that are isomorphic to a direct product of groups is described. In Section 7.7 the conclusions for this chapter are presented.

7.3 Problem Statement

In this chapter, the results presented in the chapter on analysis are used to synthesize partitionable networks. Based upon the examples and the work in the previous chapter, an algorithm is developed that allows the synthesis of a large class of partitionable networks. An interesting, special class of networks which is isomorphic to a direct product of groups is analyzed. Since the problem of decomposition of groups into a direct product of groups is well

known in abstract algebra [Han68, Her75], it can be used to evaluate the partitionability of these networks.

7.4 Previous Work

The material in this chapter, the synthesis of a partitionable interconnection networks, is directly based on the material in the chapter on the analysis of partitionable interconnection networks. The synthesis procedure is based on the chapter on analysis, consequently this chapter can be viewed as an application section of the material discussed there.

7.5 Basic Concepts

In this section the basic concepts are presented. Some definitions can be found in a text on abstract algebra and are included here for completeness only [Han68, Her75].

Definition 7.5.1:

Let G be a set with θ a binary operation with closure. Let θ be associative.

(1) $\exists\ 1 \in G \ni: \quad 1 \cdot g = g \cdot 1 = g, \ \forall g \in G.$ (1 is the identity element.)

(2) $\exists\ g^{-1} \in G \ \ni: g \cdot g^{-1} = \ g^{-1} \cdot g = 1, \quad \forall g \in G.$ (Each element has an inverse.)

Then $<G, \theta>$ is called a *group*.

Definition 7.5.2:

Let $C[V_I \times V_O]$ be a C-set. Let $V_I = \{u_0,u_1,...u_{m-1}\}$ and $V_O = \{v_0,v_1,...v_{m-1}\}$.

Define a binary *operation* γ on $C[V_I \times V_O]$ as follows:

$\gamma : C[V_I \times V_O] \times C[V_I \times V_O] \rightarrow C[V_I \times V_O]$,

$C_a \gamma C_b \triangleq \{<u_i,v_l> \mid \ <u_i,v_j> \in C_a, <u_k,v_l> \in C_b, j=k\}.$

7.6 Synthesis of Single Stage Partitionable Networks

This section consists of two major parts. In the first part, an example of a single stage partitionable network will be presented. Based upon the example and the material in the chapter on analysis an algorithm will be developed which allows the synthesis of a large class of partitionable networks. The discussion will be restricted to the case of $|V_I| = |V_O|$ for the ease of

presentation, however the results are applicable to the case $|V_I| \neq |V_O|$. The construction is given in terms of constraints on the structure of the I/O correspondences of the network. In the second part an interesting special case of single stage partitionable interconnection networks will be discussed. It will be shown that this class is isomorphic to a class of groups. For that special class of networks, it will be shown that a network is strictly σ-partitionable if and only if it is isomorphic to a direct product of two groups. Since groups have been studied extensively in abstract algebra, analytical methods are known to find a (possible) decomposition into a direct product.

Example 7.6.1:

Let $K \in K[V_I \times V_O]$, $K = \langle C \rangle$ be a network. Let $|V_I| = |V_O| = m$. Denote $V_I = \{u_0,u_1,\ldots u_{m-1}\}$ and $V_O = \{v_0,v_1,\ldots v_{m-1}\}$. Let r, $0 < r < m-1$. Let $C = \{C_p, C_q\}$, and $\oplus_r$ be addition modulo r. Let a, b, c, d be arbitrary integers such that $a \neq c \bmod r$ and $b \neq d \bmod m-r$.

$$C_p = \{\langle u_i,v_j\rangle \mid j = i \oplus_r a,\ 0 \leq i < r\} \cup$$
$$\{\langle u_i,v_j\rangle \mid j = r+(i \oplus_{m-r} b),\ r \leq i < m\},$$
$$C_q = \{\langle u_i,v_j\rangle \mid j = i \oplus_r c,\ 0 \leq i < r\} \cup$$
$$\{\langle u_i,v_j\rangle \mid j = r+(i \oplus_{m-r} d),\ r \leq i < m\}.$$

Show that the network is partitionable.

Solution:

(1): Denote $V_I^1 = \{u_0,u_1,\ldots u_{r-1}\}$, $V_O^1 = \{v_0,v_1,\ldots v_{r-1}\}$ and $V_I^2 = \{u_r,u_{r+1},\ldots u_{m-1}\}$, $V_O^2 = \{v_r,v_{r+1},\ldots v_{m-1}\}$. Intuitively the network can be partitioned into one network over $V_I^1 \times V_O^1$ and second network over $V_I^2 \times V_O^2$.

(2): Although the components of the underlying graph of K, B^1 and B^2 could be finer than described below, it is guaranteed that there are at least two components B^1 and B^2 as follows. $C_p, C_q \rightarrow$ there are at least two components B^1 and B^2.

$$B^1 = \{\overline{<u_i, v_j>} \mid j = i \oplus_r a, 0 \le i < r\} \cup \{\overline{<u_i, v_j>} \mid j = i \oplus_r c, 0 \le i < r\}.$$

$$B^2 = \{\overline{<u_i, v_j>} \mid j = r + (i \oplus_{m-r} b), r \le i < m-r\} \cup \{\overline{<u_i, v_j>} \mid j = r + (i \oplus_{m-r} d), r \le i < m-r\}.$$

(3): Let $K(B^1) \in K[V_I^1 \times V_O^1]$, $K(B^1) = <C^1>$. $C^1 = \{C_p^1, C_q^1\}$, $C_p^1 = p(C_p, B^1) = \{<u_i, v_j> \mid j = i \oplus_r a, 0 \le i < r\}$, $C_q^1 = p(C_q, B^1) = \{<u_i, v_j> \mid j = i \oplus_r c, 0 \le i < r\}$. Let $K(B^2) \in K[V_I^2 \times V_O^2]$, $K(B^2) = <C^2>$. $C^2 = \{C_p^2, C_q^2\}$, $C_p^2 = p(C_p, B^2) = \{<u_i, v_j> \mid j = r + (i \oplus_{m-r} b), r \le i < m\}$, $C_q^2 = p(C_q, B^2) = \{<u_i, v_j> \mid j = r + (i \oplus_{m-r} d), r \le i < m\}$.

(4): Then $K(B^1)\ \sigma\ K(B^2) \supseteq_c K$, therefore K is σ-partitionable.

The example can be generalized into the following algorithm to generate a large class of partitionable interconnection networks.

Algorithm 7.6.2:

Let $K \in K[V_I \times V_O]$, $K = <C>$ be a network.

(1) Let $V_I = \{u_0, u_1, \ldots u_{m-1}\}$ and $V_O = \{v_0, v_1, \ldots v_{m-1}\}$.

(2) Let $C_1, C_2, \ldots C_n \in C$, and r, $0 < r < m-1$.

$$C_1 = \{<u_i, v_j> \mid j = f_1(i) \bmod r, 0 \le i < r\} \cup \{<u_i, v_j> \mid j = r + (g_1(i) \bmod (m-r)), r \le i < m\},$$

$$C_2 = \{<u_i, v_j> \mid j = f_2(i) \bmod r, 0 \le i < r\} \cup \{<u_i, v_j> \mid j = r + (g_2(i) \bmod (m-r)), r \le i < m\}, \cdots$$

$C_n = \{<u_i,v_j> \mid j = f_n(i) \bmod r, 0 \leq i < r\} \cup$

$\{<u_i,v_j> \mid j = r+(g_n(i) \bmod (m-r)), r \leq i < m\}.$

The functions $f_k(i)$, $g_k(i)$ are arbitrary integer functions such that: If $0 \leq x, y < r$ then $(f_k(x) \bmod r) = (f_k(y) \bmod r)$ iff $x=y$. If $r \leq x, y < m$ then $(g_k(x) \bmod (m-r)) = (g_k(y) \bmod (m-r))$ iff $x=y$. The above is necessary to insure that the constructed correspondences are nondestructive.

Theorem 7.6.3:

Every network constructed using the Algorithm 7.6.2 is a σ-partitionable network.

Proof:

Similar to solution of Example 7.6.1.

□

The following algorithm will generate a large class of r-way partitionable networks, where r is the number of components. It is easier to use than the Algorithm 7.6.2 and the class is smaller than the one generated by Algorithm 7.6.2. In addition the component networks of the partitionable networks generated by Algorithm 7.6.4 are isomorphic to each other.

Algorithm 7.6.4:

Let r be any integer, let $N = r^m$. Construct a network K over $V_I \times V_O$, $K = <C>$ as follows.

(1) Let $V_I = \{p_{m-1}p_{m-2} \cdots p_0 \mid p_i=0,1,...r-1\}$, and

$V_O = \{q_{m-1}q_{m-2} \cdots q_0 \mid q_i=0,1,...r-1\}$.

(2) Construct $C_j \in C$ as follows:

$C_j = \{<p_{m-1}p_{m-2}\cdots p_0,\ p_{m-1}\pi_j(p_{m-2}\cdots p_0)>\}$, where π_j is any permutation of letters $p_{m-2}\cdots p_0$.

Theorem 7.6.5:

Every network constructed using Algorithm 7.6.4 is a σ-partitionable network.

Proof:

Intuitively, there will be r subnetworks, where the kth subnetwork has input labels of form $kp_{m-2}\cdots p_1p_0$ and output labels of the form $kq_{m-2}\cdots q_1q_0$. Using similar steps as in Example 7.6.1, it can be shown that the component networks are $\{K^k = <C^k> \mid K^k \in K[V_I^k \times V_O^k],\ k=0,1,...r-1\}$ with $V_I^k = \{kp_{m-2}\cdots p_0\}$, and $V_O^k = \{kq_{m-2}\cdots q_0\}$, $k=0,1,...r-1$.

□

Example 7.6.6:

Let r = 2, let m = 4, N = 16. Construct a network $K \in K[V_I \times V_O]$, $K = <C>$ as follows.

Let $V_I = \{p_3p_2p_1p_0 \mid p_i=0,1\}$, and

$V_O = \{q_3q_2q_1q_0 \mid q_i=0,1\}$. Let $C = \{C_0,C_1,C_2\}$.

$C_0 = \{<p_3p_2p_1p_0,\ p_3p_2p_0p_1> \mid\ p_i = 0,1\ \}$,

$C_1 = \{<p_3p_2p_1p_0,\ p_3p_0p_2p_1> \mid\ p_i = 0,1\ \}$,

$C_2 = \{<p_3p_2p_1p_0,\ p_3p_1p_0p_2> \mid\ p_i = 0,1\ \}$.

Show that the network is partitionable.

Solution:

Although there may be more then two components, it is guaranteed that there are at least two component networks. The kth subnetwork has input labels of form $kp_{m-2} \cdots p_1p_0$ and output labels of the form $kq_{m-2} \cdots q_1q_0$, $k = 0,1$. The component networks are $\{K^k = <C^k> \mid K^k \in K[V_I^k \times V_O^k],\ k=0,1\}$.

Now consider a special network $K \in K[V_I \times V_O]$, $K = <C>$ such that C, γ is a group. It is possible to view the correspondence $C_k = \{<p_i, q_j> \mid p_i \in V_I, q_j \in V_O\}$ as the permutation $\pi_k = \{<i, j> \mid i, j \in A\}$. For example let $C_k = \{<p_i, q_j> \mid p_i \in V_I, q_j \in V_O, \ j = i \oplus k \bmod m\}$ be a correspondence, the induced permutation is $\pi_k = \{<i, j> \mid j = i \oplus k \bmod m,\ i, j \in A\}$. The partitionability of this class of networks is related to the direct product composition of groups in abstract algebra as will be shown by the following theorems.

Theorem 7.6.7:

Let $K \in K[V_I \times V_O]$, $K = <C>$ be a strictly σ-partitionable network. Let C, γ be a group, where γ is the composition of maps (see definition 7.5.2). Then C, γ is isomorphic to a direct product of two groups.

Proof:

(1): As stated previously, it is assumed for simplicity of presentation that the network has only two components, call them $K(B^1) = <C^1>$ and $K(B^2) = <C^2>$. Let $C^1 = \{p(C_i,B^1) \mid C_i \in C\}$ and $C^2 = \{p(C_i,B^2) \mid C_i \in C\}$ be the two sets of correspondences. It will be shown that $<C^1, \gamma>$ and $<C^2, \gamma>$ are groups and their direct product is isomorphic

to $<C, \gamma>$.

(2): Show $<C^1, \gamma>$ is a group.

(2.1): Show $<C^1, \gamma>$ has closure: Show $\exists\ C_k \in C \ni p(C_i, B^1)\ \gamma\ p(C_j,B^1) = p(C_k, B^1)$.

(2.1.1): Let $C_k = C_i\ \gamma\ C_j$. $\rightarrow$ $p(C_k, B^1) = p(C_i\ \gamma\ C_j, B^1) = \{<u_a, v_b> \mid <u_a,v_c> \in C_i, <u_c,v_b> \in C_j, \overline{<u_a,v_b>} \in B^1\}$.

(2.1.2): $<u_a,v_c> \in C_i$, $\overline{<u_a,v_b>} \in B^1 \rightarrow \overline{<u_a,v_c>} \in B^1 \rightarrow <u_a,v_c> \in p(C_i, B^1)$.

(2.1.3): $<u_c,v_b> \in C_j$, $\overline{<u_a,v_b>} \in B^1 \rightarrow \overline{<u_c,v_b>} \in B^1 \rightarrow <u_c,v_b> \in p(C_j, B^1)$.

(2.1.4): $p(C_k, B^1) = \{<u_a, v_b> \mid <u_a,v_c> \in C_i, <u_c,v_b> \in C_j, \overline{<u_a,v_b>} \in B^1\} = \{<u_a, v_b> \mid <u_a,v_c> \in p(C_i, B^1), <u_c,v_b> \in p(C_j, B^1)\} = p(C_i, B^1)\ \gamma\ p(C_j,B^1)$.

(2.2): Show γ is associative:

By definition of the operation.

(2.3): Let $C_t \in C$, $C_t\ \gamma\ C_i = C_i\ \gamma\ C_t = C_i$ then $p(C_t,B^1)$ is the identity in C^1.

(2.4): Show C^1 contains inverses. Let $C_i\ \gamma\ C_j = C_t$ then $p(C_i,B^1)\ \gamma\ p(C_j,B^1) = p(C_t,B^1)$.

(3): Similarly can show $<C^2, \gamma>$ is a group.

(4): Construct direct product group $<C^1 \times C^2, \otimes>$.

(4.1): Define $\otimes$:

$<p(C_i, B^1), p(C_j, B^2)> \otimes <p(C_k, B^1), p(C_l, B^2)> = <p(C_i, B^1)\ \gamma\ p(C_k, B^1), p(C_j, B^2)\ \gamma\ p(C_l, B^2)>$.

From algebra it is known that the direct product of groups is a group.

(5): Define θ: $C \rightarrow C^1 \times C^2$, $\theta(C_i) = <p(C_i, B^1), p(C_i, B^2)>$.

(5.1): Show θ is group homomorphism.

Show: $\theta(C_i \gamma C_j) = \theta(C_i) \otimes \theta(C_j)$.

(5.1.1): $\theta(C_i \gamma C_j) = \theta(C_k) \ni C_k = C_i \gamma C_j$.

(5.1.2): $\theta(C_k) = <p(C_k, B^1), p(C_k, B^2)>$.

(5.1.3): (2) and (3) $\rightarrow$ $p(C_i, B^1) \gamma p(C_j,B^1) = p(C_k, B^1)$, and $p(C_i, B^2) \gamma p(C_j,B^2) = p(C_k, B^2)$

$\rightarrow$ $<p(C_k, B^1), p(C_k, B^2)>$

$= <p(C_i, B^1) \gamma p(C_j,B^1), p(C_i, B^2) \gamma p(C_j,B^2)>$

$= <p(C_i, B^1), p(C_i, B^2)> \otimes <p(C_j, B^1), p(C_j, B^2)>$

$= \theta(C_i) \otimes \theta(C_j)$. Therefore θ is a group homomorphism.

(5.2): $<p(C_i, B^1), p(C_j, B^2)> \in C^1 \times C^2$ and K is strictly σ-partitionable $\rightarrow$ $\exists C_k \ni p(C_i, B^1) = p(C_k, B^1)$ and $p(C_j, B^2) = p(C_k, B^2)$ therefore θ is onto.

(5.3): Show θ is 1:1.

Kernel of $\theta = \{C_t\}$, where C_t is identity of $<C, \gamma> \rightarrow \theta$ is 1:1.

(6): (5) $\rightarrow$ $<C, \gamma>$ isomorphic to $<C^1 \times C^2, \otimes>$.

□

Theorem 7.6.8:

Let $<C, \gamma>$ be a group that is isomorphic to a direct product of two groups $<C^1, \gamma>$ and $<C^2, \gamma>$. Then $K = <C>$ is a strictly σ-partitionable network.

Proof:

(1): Let $<C, \gamma>$ be the group. Let $<C, \gamma> \approx <C, \gamma> \oplus <C, \gamma>$ where $\otimes$ is the direct product and the sets C^1 and C^2 are as in the proof of Theorem 7.6.7.

(2): Then the rest of the proof consists of reversing the steps of proof of Theorem 7.6.7.

□

The next two examples show an application of Theorem 7.6.8.

Example 7.6.9:

Let $K \in K[V_I \times V_O]$, $K = <C>$ be a network. Let $V_I = \{u_0,u_1,u_2,u_3,u_4,u_5\}$ and $V_O = \{v_0,v_1,v_2,v_3,v_4,v_5\}$.

Let $C = \{C_1,C_2,C_3,C_4\}$,

$C_1 = \{<u_i,v_i> \mid i=0,1,...5\}$,

$C_2 = \{<u_0,v_1>, <u_1,v_0>, <u_i,v_i> \mid i=2,3,...5\}$,

$C_3 = \{<u_0,v_0>, <u_1,v_1>, <u_2,v_3>, <u_3,v_2>,$
$<u_4,v_5>, <u_5,v_4>\}$,

$C_4 = \{<u_0,v_1>, <u_1,v_0>, <u_2,v_3>, <u_3,v_2>,$
$<u_4,v_5>, <u_5,v_4>\}$.

Show that the network is strictly σ-partitionable.

Solution:

(1): Show that C, γ is a group.

From the multiplication table, Table 7.1, it can be seen that $<C, \gamma>$ is a group.

(2): Consider $<C^1,\gamma>$. Let $C^1 = \{C_1^1,C_2^1\}$, $C_1^1 = \{<u_0,v_0>, <u_1,v_1>\}$, and $C_2^1 = \{<u_0,v_1>, <u_1,v_0>\}$. Then $<C^1, \gamma>$ is a group.

(3): Consider $<C^2,\gamma>$. Let $C^2 = \{C_1^2,C_2^2\}$, $C_1^2 = \{<u_i,v_i> \mid i=2,3,...5\}$, and $C_2^2 = \{<u_2,v_3>, <u_3,v_2>, <u_4,v_5>, <u_5,v_4>\}$. Then $<C^2, \gamma>$ is a group.

(4): Let the direct product group be $<C^1 \times C^2, \otimes>$ where $<C_i^1, C_j^2> \otimes <C_k^1, C_l^2> = <C_i^1 \gamma C_k^1, C_j^2 \gamma C_l^2>$.

(5): Given $\theta : C \rightarrow C^1 \times C^2$ a map, $\theta(C_i) = <p(C_i,B^1), p(C_i,B^2)>$, and $C_1 = C_1^1 \cup C_1^2$, $C_2 = C_2^1 \cup C_1^2$, $C_3 = C_1^1 \cup C_2^2$, $C_4 = C_2^1 \cup C_2^2$, then it is easy to show that θ is homomorphism, onto and 1:1.

(6): Consequently K is a strictly σ-partitionable network.

Example 7.6.10:

Let $K \in K[V_I \times V_O]$, $K = <C>$ be a network. Let $V_I = \{u_0,u_1,u_2,u_3,u_4\}$ and $V_O = \{v_0,v_1,v_2,v_3,v_4\}$.

Let $C = \{C_1,C_2,C_3,C_4, C_5,C_6\}$, $C_1 = \{<u_i,v_i> \mid i=0,1,...4\}$,

$C_2 = \{<u_0,v_0>, <u_1,v_2>, <u_2,v_3>, <u_3,v_1>, <u_4,v_4>\}$,

$C_3 = \{<u_0,v_0>, <u_1,v_3>, <u_2,v_1>, <u_3,v_2>, <u_4,v_4>\}$,

$C_4 = \{<u_0,v_4>, <u_4,v_0>, <u_i,v_i> \mid i=1,2,3\}$,

$C_5 = \{<u_0,v_4>, <u_1,v_2>, <u_2,v_3>, <u_3,v_1>, <u_4,v_0>\}$,

Table 7.1:
The multiplication table for $<C, \gamma>$, Example 7.6.9.

γ	C_1	C_2	C_3	C_4
C_1	C_1	C_2	C_3	C_4
C_2	C_2	C_1	C_4	C_3
C_3	C_3	C_4	C_1	C_2
C_4	C_4	C_3	C_2	C_1

$C_6 = \{<u_0,v_4>, <u_1,v_3>, <u_2,v_1>, <u_3,v_2>, <u_4,v_0>\}$.

Show that the network is strictly σ-partitionable.

Solution:

(1): Show that $<C, \gamma>$ is a group.

Constructing multiplication table, Table 7.2 $<C, \gamma>$ is a group.

(2): Consider $<C^1,\gamma>$. Let $C^1 = \{C_1^1,C_2^1\}$, $C_1^1 = \{<u_0,v_0>, <u_4,v_4>\}$, and $C_2^1 = \{<u_0,v_4>, <u_4,v_0>\}$. Then $<C^1, \gamma>$ is a group.

(3): Consider $<C^2,\gamma>$. Let $C^2 = \{C_1^2,C_2^2,C_3^2\}$, $C_1^2 = \{<u_i,v_i> \mid i=1,2,3\}$, $C_2^2 = \{<u_1,v_2>, <u_2,v_3>, <u_3,v_1>\}$, $C_3^2 = \{<u_1,v_3>, <u_2,v_1>, <u_3,v_2>\}$. Then $<C^2, \gamma>$ is a group.

(4): Let the direct product group be $<C^1 \times C^2, \otimes>$ where $<C_i^1, C_j^2> \otimes <C_k^1, C_l^2> = <C_i^1 \gamma C_k^1, C_j^2 \gamma C_l^2>$.

(5): Same as Example 7.6.9. θ is homomorphism, onto and 1:1.

(6): Consequently K is a strictly σ-partitionable network.

7.7 Conclusions

In this chapter the problem of synthesis of single stage partitionable networks was studied. This chapter can also be viewed as an application

Table 7.2:
The multiplication table for $<C, \gamma>$, Example 7.6.10.

γ	C_1	C_2	C_3	C_4	C_5	C_6
C_1	C_1	C_2	C_3	C_4	C_5	C_6
C_2	C_2	C_3	C_1	C_5	C_6	C_4
C_3	C_3	C_1	C_2	C_6	C_4	C_5
C_4	C_4	C_5	C_6	C_1	C_2	C_3
C_5	C_5	C_6	C_4	C_2	C_3	C_1
C_6	C_6	C_4	C_5	C_3	C_1	C_2

section of the chapter on analysis. This chapter contains two algorithms and a theorem describing the construction of σ-partitionable networks. The first algorithm is the most general and produces a large class of σ-partitionable networks. The second algorithm is easier to use but it generates a smaller class of σ-partitionable networks. The theorem describes the existence of a class of strictly σ-partitionable networks and it can be used in bidirectional sense, that is (a) can be used to decide whether a certain class of networks is strictly σ-partitionable and (b) can be used to construct a class of strictly σ-partitionable networks. The theorem says the following. Let $K = <C>$ be a network. If C, γ is a group and K is strictly σ-partitionable, then C, γ must be isomorphic to a direct product of groups. The problem of decomposing groups into direct products has been studied extensively in the group theory so the results derived in abstract algebra can be directly applied here.

8 MULTISTAGE NETWORKS - ANALYSIS

8.1 Introduction

In this chapter the analysis of multistage networks will be addressed [AdS82b, Bat74, Ben74, BoD72, Fen81, McS82]. This extends the work done on single stage networks in previous chapters into the domain of multistage interconnection networks. Although parts of the work done on single stage networks are transferable to the domain of multistage networks, the concepts are more complicated.

The material in this chapter will be presented as follows. A vertical composition of networks will be defined and its properties shown. Using vertical composition and the model of single stage networks, multistage networks will be defined. By using the single stage model, which was analyzed earlier, as a building block for multistage networks, some results from the study of single stage networks can be applied to multistage networks. The multistage network model is very general since each stage can be a completely general single stage network. This model differs from some of the previous models of regular multistage networks by being completely general and therefore applicable to all multistage networks. Several examples of applications of the model are discussed.

8.2 Overview

The organization of the chapter is as follows. In Section 8.3 the problem will be informally defined. In Section 8.4 the previous work will be reviewed. In Section 8.5 the basic definitions such as vertical composition of networks will be presented and its properties analyzed. In Section 8.6 the formal definition of multistage network is developed and several examples of applications of the model are given. The chapter is summarized in Section 8.7.

8.3 Problem Statement

In this chapter a formal definition of multistage networks will be developed. First a vertical composition of single stage networks is defined. Some properties of the composition are exhibited. Then multistage networks are defined by the vertical composition of single stage networks where V_O of the ith network is equal to the V_I of the i+1st network. The multistage network model is very general since each stage consists of a completely general single stage network.

8.4 Previous Work

Multistage networks have been modeled by researchers using different techniques. Examples of such networks are ADM, IADM [McS82], indirect binary n-cube [Pea77], Generalized Cube [SiM81], STARAN [Bat77], Omega [Law75], baseline [WuF80], Shuffle-Exchange [ThN81], binary tree [BeK79], Benes [Ben65], and Banyan [GoL73, Upp81]. Modeling methods used are graph-theoretic [McS82, GoL73, Upp81], algebraic [Ben65], and others. Our method is based on the general model of single stage networks, presented earlier as well as the vertical composition of single stage networks using algebraic operations. This approach allows the analysis of multistage networks to draw on some results valid for single stage networks.

8.5 Basic Concepts

In this section the basic definitions and concepts are presented as well as some of their properties. These basic concepts will be used in this chapter as well in the chapter on synthesis of multistage networks.

Definition 8.5.1:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$. $C_p^1 \gamma C_q^2 \triangleq \{<u_a,v_b> \mid <u_a,w_c> \in C_p^1, <w_c,v_b> \in C_q^2\}$. Then γ is called a γ *composition of I/O correspondences.*

The Theorems, Lemmas and Definitions 8.5.2 to 8.5.11 discuss the properties of the γ composition of I/O correspondences which will be used to define the β-map. The β-map is the network composition used to construct multistage networks and its properties are determined by the properties of the γ composition of I/O correspondences.

The following theorem shows that the γ composition of two nondestructive I/O correspondences is a nondestructive I/O correspondence.

Theorem 8.5.2:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$. Then $C_p^1 \gamma C_q^2 \in C[V_I^1 \times V_O^2]$.

Proof:

(1): Clearly $C_p^1 \gamma C_q^2 \in P[V_I^1 \times V_O^2]$, that is $C_p^1 \gamma C_q^2$ is an I/O correspondence over $V_I^1 \times V_O^2$.

(2): Show $C_p^1 \gamma C_q^2 \in C[V_I^1 \times V_O^2]$, that is $C_p^1 \gamma C_q^2$ is a nondestructive I/O correspondence over $V_I^1 \times V_O^2$.
Assume $C_p^1 \gamma C_q^2 \notin C[V_I^1 \times V_O^2]$
$\rightarrow$ $<u_a,w_b>$, $<u_c,w_b> \in C_p^1 \gamma C_q^2$, that is $C_p^1 \gamma C_q^2$ is not a nondestructive I/O correspondence over $V_I^1 \times V_O^2$.

(2.1): Case 1: $<u_a,v_x>$, $<u_c,v_x> \in C_p^1$ and $<v_x,w_b> \in C_q^2$
$\rightarrow$ contradiction since $C_p^1 \in C[V_I^1 \times V_O^1]$, (that is $<u_a,v_x>$,

$<u_c,v_x> \in C_p^1 \rightarrow C_p^1$ is not nondestructive).

(2.2): Case 2: $<u_a,v_x>$, $<u_c,v_y> \in C_p^1$ and

$<v_x,w_b>$, $<v_y,w_b> \in C_q^2$

$\rightarrow$ contradiction since $C_q^2 \in C[V_I^2 \times V_O^2]$, (that is, $<v_x,w_b>$, $<v_y,w_b> \in C_q^2 \rightarrow C_q^2$ is not nondestructive).

□

The following theorem shows that the γ composition of I/O correspondences is associative.

Theorem 8.5.3:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$, $C_q^2 \in C[V_I^2 \times V_O^2]$, and $C_r^3 \in C[V_I^3 \times V_O^3]$.

Let $V_O^1 = V_I^2$ and $V_O^2 = V_I^3$.

Then $C_p^1 \gamma (C_q^2 \gamma C_r^3) = (C_p^1 \gamma C_q^2) \gamma C_r^3$.

Proof:

(1): Show $C_p^1 \gamma (C_q^2 \gamma C_r^3) \subseteq (C_p^1 \gamma C_q^2) \gamma C_r^3$.

(1.1): $<u_a,x_b> \in C_p^1 \gamma (C_q^2 \gamma C_r^3)$

$\rightarrow <u_a,v_c> \in C_p^1$, $<v_c,x_b> \in C_q^2 \gamma C_r^3$

$\rightarrow <v_c,w_d> \in C_q^2$, $<w_d,x_b> \in C_r^3$.

(1.2): $<u_a,v_c> \in C_p^1$, $<v_c,w_d> \in C_q^2$

$\rightarrow <u_a,w_d> \in C_p^1 \gamma C_q^2$.

(1.3): $<u_a,w_d> \in C_p^1 \gamma C_q^2$, $<w_d,x_b> \in C_r^3$

$\rightarrow <u_a,x_b> \in (C_p^1 \gamma C_q^2) \gamma C_r^3$

$\rightarrow C_p^1 \gamma (C_q^2 \gamma C_r^3) \subseteq (C_p^1 \gamma C_q^2) \gamma C_r^3$.

(2): Similarly can show $C_p^1 \gamma (C_q^2 \gamma C_r^3) \supseteq (C_p^1 \gamma C_q^2) \gamma C_r^3$.

(3): (1) and (2) $\rightarrow$ $C_p^1 \gamma (C_q^2 \gamma C_r^3) = (C_p^1 \gamma C_q^2) \gamma C_r^3$.

□

The following theorem shows that the γ composition of I/O correspondences is not commutative.

Theorem 8.5.4:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$ and $V_O^2 = V_I^1$. Then $C_p^1 \gamma C_q^2 \neq C_q^2 \gamma C_p^1$ in general.

Proof:

For example, if $C_p^1 = \{<u_a,w_b>\}$ and $C_q^2 = \{<w_b,v_c>\}$, then $C_p^1 \gamma C_q^2 = \{<u_a,v_c>\}$, and $C_q^2 \gamma C_p^1 = \emptyset$.

□

Definition 8.5.5:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$.

$C_p^1 \gamma C^2 \triangleq \{C_p^1 \gamma C_q^2 \mid C_q^2 \in C^2\}$.

The following theorem says that the γ composition of a nondestructive correspondence with a set of nondestructive correspondences produces a set of nondestructive correspondences.

Theorem 8.5.6:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$. Then $C_p^1 \gamma C^2 \subseteq C[V_I^1 \times V_O^2]$.

Proof:

From Theorem 8.5.2, $\forall C_q^2 \in C^2$, $C_p^1 \gamma C_q^2 \in C[V_I^1 \times V_O^2] \rightarrow$ $C_p^1 \gamma C^2 \subseteq C[V_I^1 \times V_O^2]$.

□

The following theorem shows that the γ composition of I/O correspondence with a set of I/O correspondences is not commutative.

Theorem 8.5.7:

Let $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$ and $V_O^2 = V_I^1$.

Then $C_p^1 \gamma C^2 \neq C^2 \gamma C_p^1$ in general.

Proof:

Similar to the proof of Theorem 8.5.4.

□

Definition 8.5.8:

Let $C^1 = \{C_p^1 \mid p=1,2,...m\}$, $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$.

$C^1 \gamma C^2 \triangleq \{C_p^1 \gamma C_q^2 \mid C_p^1 \in C^1, C_q^2 \in C^2\}$. Then γ is called a γ *composition of sets of I/O correspondences.*

The following theorem says that the γ composition of two a sets of I/O correspondences produces a set of I/O correspondences.

Theorem 8.5.9:

Let $C^1 = \{C_p^1 \mid p=1,2,...m\}$, $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$.

Then $C^1 \gamma C^2 \subseteq C[V_I^1 \times V_O^2]$.

Proof:

Theorem 8.5.2 $\rightarrow$ $C_p^1 \gamma C_q^2 \in C[V_I^1 \times V_O^2]$

$\rightarrow$ $C^1 \gamma C^2 \subseteq C[V_I^1 \times V_O^2]$.

□

The following theorem shows that the γ composition of sets of I/O correspondences is associative.

Theorem 8.5.10:

Let $C^1 = \{C_p^1 \mid p=1,2,...m\}$, $C_p^1 \in C[V_I^1 \times V_O^1]$, $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$, and $C^3 = \{C_r^3 \mid r=1,2,...o\}$, $C_r^3 \in C[V_I^3 \times V_O^3]$. Let $V_O^1 = V_I^2$ and $V_O^2 = V_I^3$.

Then $C^1 \gamma (C^2 \gamma C^3) = (C^1 \gamma C^2) \gamma C^3$.

Proof:

Theorem 8.5.3 $\rightarrow$ $C_p^1 \gamma (C_q^2 \gamma C_r^3) = (C_p^1 \gamma C_p^2) \gamma C_r^3$.

$\rightarrow$ $C^1 \gamma (C^2 \gamma C^3) = (C^1 \gamma C^2) \gamma C^3$.

□

The following theorem shows that the γ composition of sets of I/O correspondences is not commutative.

Theorem 8.5.11:

Let $C^1 = \{C_p^1 \mid p=1,2,...m\}$, $C_p^1 \in C[V_I^1 \times V_O^1]$ and $C^2 = \{C_q^2 \mid q=1,2,...n\}$, $C_q^2 \in C[V_I^2 \times V_O^2]$. Let $V_O^1 = V_I^2$ and $V_O^2 = V_I^1$.

Then $C^1 \gamma C^2 \neq C^2 \gamma C^1$ in general.

Proof:

Apply Theorem 8.5.7.

□

In the following part the β-map will be defined and its properties studied. The β-map is used to define multistage networks and its properties are based on the properties of the γ-composition of I/O correspondences discussed earlier.

Definition 8.5.12:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = <C^1>$ and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = <C^2>$. Let $V_O^1 = V_I^2$.

Define β-map as follows: $K^1 \beta K^2 = <C^1> \beta <C^2> \triangleq <\{C_p^1 \gamma C_q^2 \mid C_p^1 \in C^1, C_q^2 \in C^2\}>$.

The β-map describes the composition of networks where all outputs of the first network K^1 are connected into (all) inputs of the second network ($V_O^1 = V_I^2$). This is referred to as *vertical composition* of networks. This situation arises in the construction of multistage and is motivated by existing multistage networks such as ADM, Cube or STARAN network, where each stage may be considered a network.

The following theorem shows that the β composition of two networks results in a network over $V_I^1 \times V_O^2$.

Theorem 8.5.13:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = \langle C^1 \rangle$ and $K^2 \in K[V_I^2 \times V_O^2]$, $K^2 = \langle C^2 \rangle$. Let $V_O^1 = V_I^2$.

Let $K^1 \beta K^2 = \langle C^1 \rangle \beta \langle C^2 \rangle = \langle \{C_p^1 \gamma C_q^2 \mid C_p^1 \in C^1, C_q^2 \in C^2\} \rangle$. Then $\langle \{C_p^1 \gamma C_q^2 \mid C_p^1 \in C^1, C_q^2 \in C^2\} \rangle \in K[V_I^1 \times V_O^2]$.

Proof:

(1): Theorem 8.5.9 $\rightarrow$ $C^1 \gamma C^2 \subseteq C[V_I^1 \times V_O^2]$.

(2): Show $s(C^1 \gamma C^2) = V_I^1$.

$\forall u_a \in V_I^1 \ \exists v_x \in V_O^1$ and $C_p^1 \in C^1 \ni \langle u_a, v_x \rangle \in C_p^1$,

$\forall v_x \in V_O^1 \ \exists w_b \in V_O^2$ and $C_q^2 \in C^2 \ni \langle v_x, w_b \rangle \in C_q^2$

$\rightarrow \forall u_a \in V_I^1 \ \exists C_p^1 \in C^1$ and $C_q^2 \in C^2 \ni \langle u_a, w_b \rangle \in C_p^1 \gamma C_q^2 \rightarrow s(C^1 \gamma C^2) = V_I^1$.

(3): Show $d(C^1 \gamma C^2) = V_O^2$,

$\forall w_a \in V_O^2 \ \exists v_x \in V_I^2$ and $C_q^2 \in C^2 \ni \langle v_x, w_a \rangle \in C_q^2$,

$\forall v_x \in V_I^2 \ \exists u_b \in V_I^1$ and $C_p^1 \in C^1 \ni \langle u_b, v_x \rangle \in C_p^1$

$\rightarrow \forall w_a \in V_O^2 \ \exists C_p^1 \in C^1$ and $C_q^2 \in C^2 \ni \langle u_b, w_a \rangle \in C_p^1 \gamma C_q^2 \rightarrow d(C^1 \gamma C^2) = V_O^2$.

(4): Show $|C^1 \gamma C^2| \geq 2$.

(4.1): $|C^1| \geq 2 \rightarrow \exists C_p^1, C_q^1 \in C^1, C_p^1 \neq C_q^1$

$\rightarrow \exists \langle u_a, v_b \rangle \in C_p^1, \langle u_a, v_b \rangle \notin C_q^1$.

(4.2): $v_b \in V_O^1 \rightarrow v_b \in V_I^2 \rightarrow \exists C_r^2 \in C^2, \langle v_b, w_c \rangle \in C_r^2$, $C_r^2 \in C[V_I^2 \times V_O^2] \rightarrow \langle v_x, w_c \rangle \notin C_r^2, v_x \neq v_b$.

(4.3): (4.1), (4.2) $\rightarrow \langle u_a, w_c \rangle \in C_p^1 \gamma C_r^2, \langle u_a, w_c \rangle \notin C_q^1 \gamma C_r^2$

$\rightarrow C_p^1 \gamma C_r^2 \neq C_q^1 \gamma C_r^2 \rightarrow |C^1 \gamma C^2| \geq 2$.

□

The analysis of the partitionability of multistage networks will necessitate the analysis of a network with a stage fixed at a given correspondence. Consequently, the fixed stage no longer qualifies as a reconfigurable network as originally defined. Therefore one cannot use the β-map to describe the vertical composition of the reconfigurable stages and the fixed stage of the network. To handle the problem, one could either define a new map, or use the β-map with the understanding that the fixed stage is not a reconfigurable network. The latter approach will be used here.

The following corollary shows that the β composition of a network and a fixed network stage results in a network over $V_I^1 \times V_O^2$.

Corollary 8.5.14:

Let $K^1 \in K[V_I^1 \times V_O^1]$, $K^1 = \langle C^1 \rangle$ and $C_q^2 \in C[V_I^2 \times V_O^2]$, $s(C_q^2) = V_I^2$, $d(C_q^2) = V_O^2$. Let $V_O^1 = V_I^2$.

Then $K^1 \beta C_q^2 = \langle C^1 \rangle \beta C_q^2 = \langle \{C_p^1 \gamma C_q^2 \mid C_p^1 \in C^1\} \rangle \in K[V_I^1 \times V_O^2]$.

Proof:

Similar to proof of Theorems 8.5.6 and 8.5.13.

□

8.6 Multistage Network Model and Applications

In this section multistage networks will be formally defined. The definition is based upon the examples of widely known multistage networks such as ADM, Cube, STARAN, and others.

Definition 8.6.1:

Let $K^r \in K[V_I^r \times V_O^r]$, $K = <C^r>$, $r = 0,1,...t-1$ be a set of interconnection networks. Let $V_O^r = V_I^{r+1}$ $r = 0,1,...t-2$. Then $K = K^0 \beta K^1 \beta \cdots K^{t-1}$ is a multistage network over $V_I^0 \times V_O^{t-1}$. Note that K can also be represented as $K = <C>$, $C = C^0 \gamma C^1 \gamma \cdots C^{t-1}$.

Intuitively K^r describes the rth stage of the multistage network.

In this part some applications of the model of multistage networks will be presented. Although parts of the research done on single stage networks are transferable to the domain of the multistage networks, the concepts are more complicated. Examples of some artificially constructed networks will be given in details. The examples of the networks are constructed in such way as to illuminate the different types of partitionability of multistage networks. Informally partitionability in multistage networks is achieved by selecting specific controls in some stages and letting all other stages dynamically select their correspondences. Hereafter, the former stages will be referred to as fixed and the latter as free stages. Although the material is presented for the case of partitionability into two component networks, it is easily generalized into $r > 2$ component networks.

Example 8.6.2:

Consider the following multistage network (Figure 8.1.)

The network has the following functionality.

(1) There are two stages denoted by G^0, G^1.

(2) There are two switching elements E_i^r, i=0,1 in each stage r.

(3) Each switching element E_i^r, i=0,1 has the following functionality. $V_I = \{a,b\}$, $V_O = \{c,d\}$, $C = \{C_0,C_1\}$, $C_0 = \{<a,c>, <b,d>\}$, $C_1 = \{<a,d>, <b,c>\}$. (This is the same as the straight and exchange settings, respectively, of a multistage Cube type network [Law75].)

It can be shown that if in stage G^0, in E_i^0, i=0,1 the C_0 is selected, then the network can be partitioned into

$K^0 \in K[\{u_0,u_2\} \times \{w_0,w_2\}]$ and $K^1 \in K[\{u_1,u_3\} \times \{w_1,w_3\}]$.

Example 8.6.3:

Consider the following multistage network (Figure 8.2.)

The network has the following functionality.

(1) There are two stages denoted by G^0, G^1.

(2) There is one switching element E^r in each stage r.

(3) Each switching element has the following functionality. $V_I = \{a,b,c,d\}$, $V_O = \{e,f,g,h\}$, $C = \{C_0,C_1\}$,
$C_0 = \{<a,e>, <b,f>, <c,g>, <d,h>\}$,
$C_1 = \{<a,f>, <b,e>, <c,h>, <d,g>\}$.

It can be shown that if in stage G^0, the C_0 is selected, then the network can be partitioned into

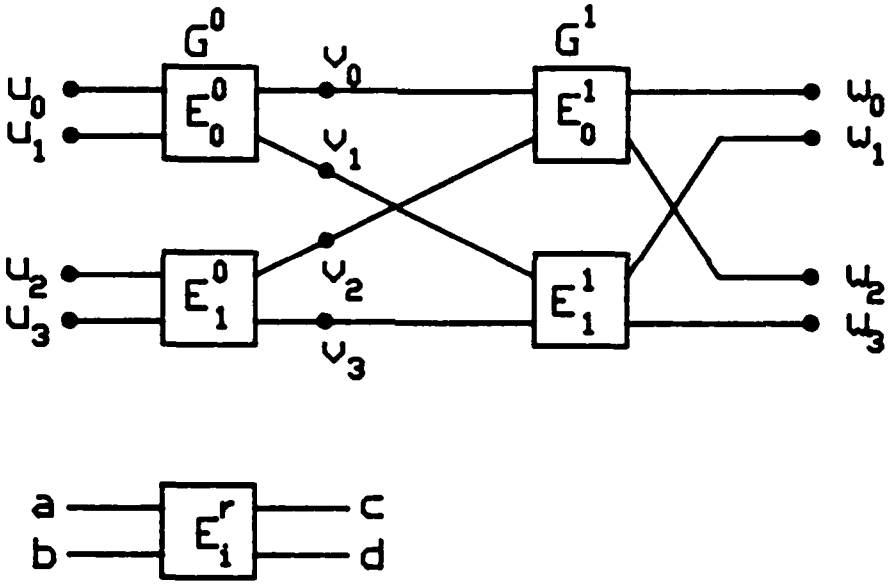


Figure 8.1:
Multistage network for Example 8.6.2.

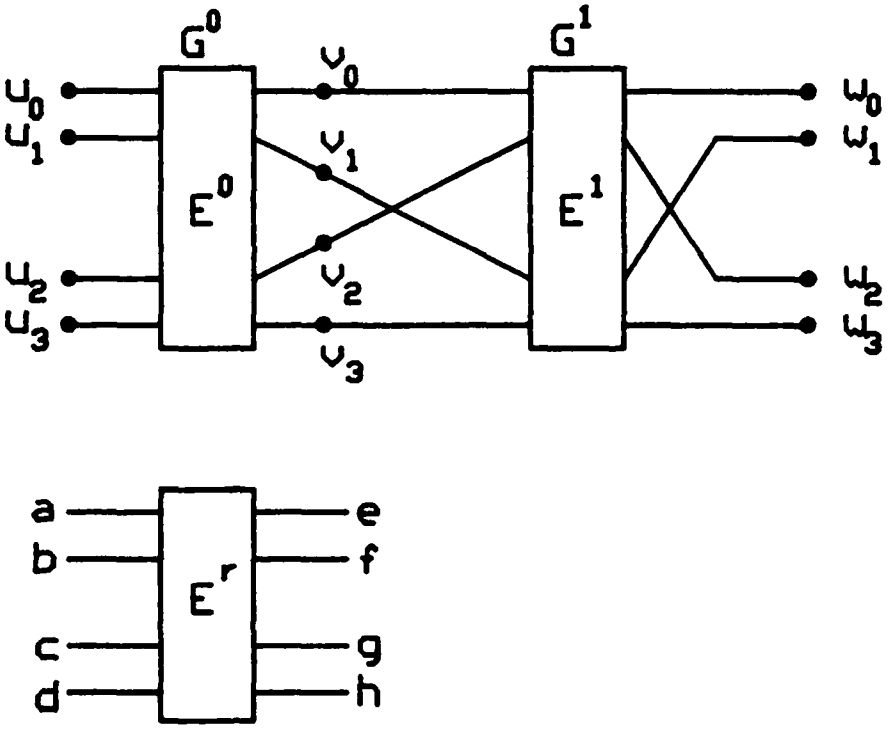


Figure 8.2:
Multistage network for Example 8.6.3.

$K^0 \in K[\{u_0,u_2\} \times \{w_0,w_2\}]$ and $K^1 \in K[\{u_1,u_3\} \times \{w_1,w_3\}]$.

Example 8.6.4:

Consider the following multistage network (Figure 8.3.)

The network has the following functionality.

(1) There are two stages denoted by G^0, G^1.

(2) There are two switching elements in E_0^0, E_1^0 in stage G^0 and one switching element E_0^1 in stage G^1.

(3) Each switching element in stage G^0 has the following functionality. $V_I = \{a,b\}$, $V_O = \{c,d\}$, $C = \{C_0,C_1\}$, $C_0 = \{<a,c>, <b,d>\}$, $C_1 = \{<a,d>, <b,c>\}$.

(4) The switching element in stage G^1 has the following functionality. $V_I = \{a,b,c,d\}$, $V_O = \{e,f,g,h\}$, $C = \{C_0,C_1\}$,
$C_0 = \{<a,e>, <b,f>, <c,g>, <d,h>\}$,
$C_1 = \{<a,f>, <b,e>, <c,g>, <d,h>\}$.

It can be shown that if in stage G^1, in the switching element E_0^1 the C_0 is selected, then the network can be partitioned into $K^0 \in K[\{u_0,u_1\} \times \{w_0,w_1\}]$ and $K^1 \in K[\{u_2,u_3\} \times \{w_2,w_3\}]$.

Example 8.6.5:

Consider the following multistage network (Figure 8.4.)

(1) There are two stages denoted by G^0, G^1.

(2) There is one switching element E_0^0 in stage G^0 and two switching elements E_0^1, E_1^1 in stage G^1.

(3) The switching element E_0^0 has the following functionality. $V_I = \{a,b\}$, $V_O = \{c,d\}$, $C = \{C_0,C_1\}$, $C_0 = \{<a,c>, <b,d>\}$,

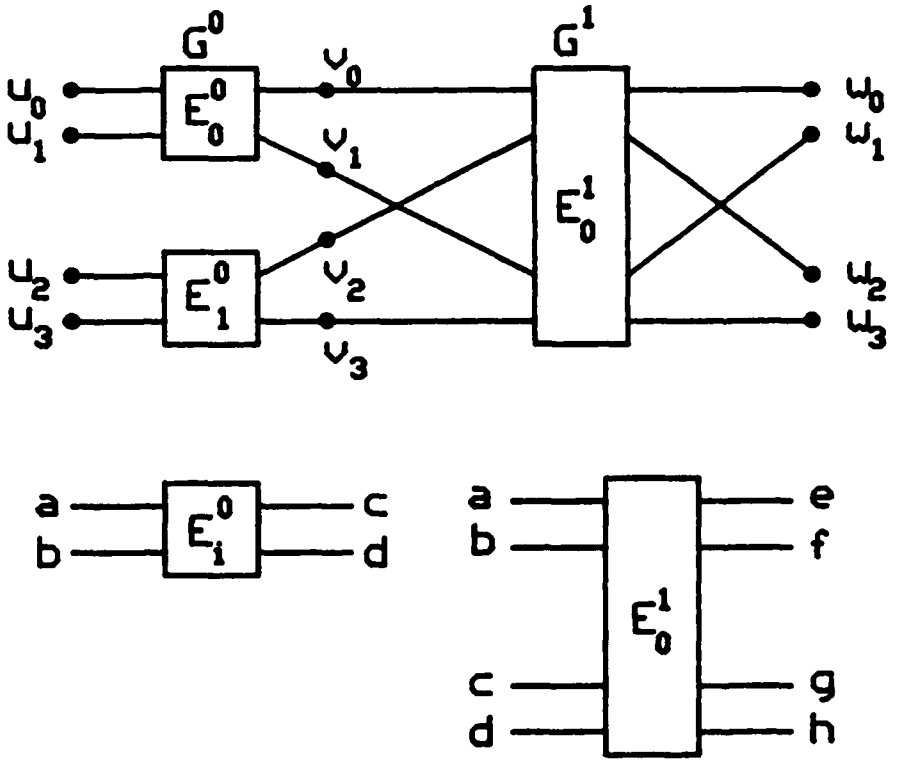


Figure 8.3:
Multistage network for Example 8.6.4.

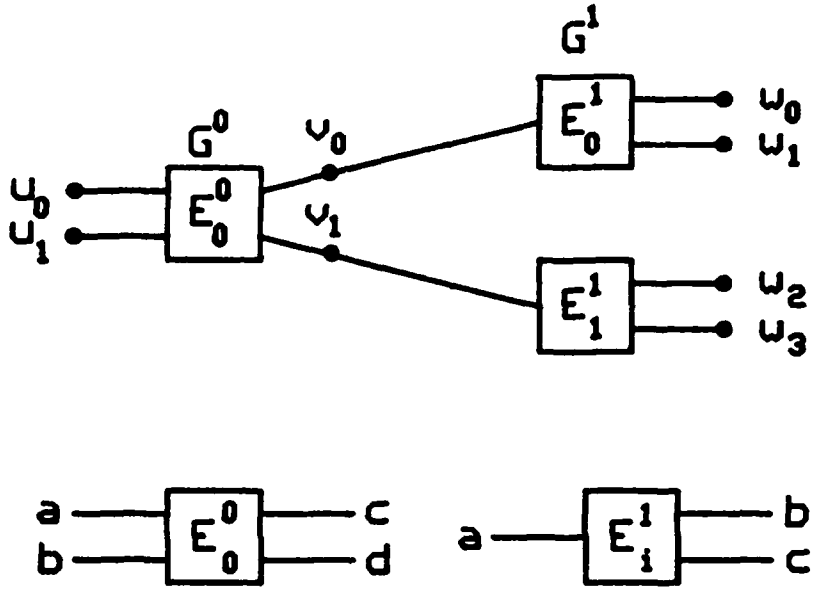


Figure 8.4:
Multistage network for Example 8.6.5.

$C_1 = \{<a,d>, <b,c>\}$.

(4) The switching elements E_0^1, E_1^1 have the following functionality. $V_I = \{a\}$, $V_O = \{b,c\}$, $C = \{C_0,C_1\}$, $C_0 = \{<a,b>\}$, $C_1 = \{<a,c>\}$.

It can be shown that if in stage G^0, the C_0 is selected, then the network can be partitioned into $K^0 \in K[\{u_0\} \times \{w_0,w_1\}]$ and $K^1 \in K[\{u_1\} \times \{w_2,w_3\}]$. Instead of fixing the G^0 setting, consider the setting of the switches in stage G^1. If in stage G^1 in switching elements E_0^1, E_1^1 either the C_0 or C_1 is selected, then the network is not partitionable due to the following. If C_0 is selected then w_1 and w_3 are not accessible and if C_1 is selected then w_0 and w_2 are not accessible in any state.

To summarize the information from the examples, the following is essential for a multistage network to be partitionable. The network must have more than one stage. There must be at least one stage such that if one state in that stage is selected, two data path independent (and possibly control independent) networks are generated. The two subnetworks must have V_I^0, V_I^1 and V_O^0, V_O^1 such that $V_I^0 \cup V_I^1 = V_I$ and $V_O^0 \cup V_O^1 = V_O$. It is not essential that the controls of the generated subnetworks are independent of each other. It is not essential that each subnetwork will be again partitionable although it is an interesting subclass.

9 DATA COMMUNICATION IN A REAL-TIME SYSTEM

9.1 Introduction

This is a study of a network design to support interprocessor data communications in a proposed real-time distributed, digital signal processing system structure [SeS84c]. The system is general nature in that it may be used in a wide variety of signal processing applications, such as Finite Impulse Response (FIR) filters, FFT and beamforming. Fault tolerance is a significant design issue for this system. In particular, the ability to reallocate distributed processing resources with minimal human intervention is important in order to maintain a functioning system, although possibly somewhat degraded in performance. The overall design of the system reflects its fault tolerance and generality.

Figure 9.1 shows the signal data transfer parameters for three iterations of the evolutionary distributed signal processing system. These parameters are based on expectations for this type of system and are used as guidelines for the design work in this study. Three phases, A, B, and C, are indicated in the figure. Phase A is the 1985 time-frame, Phase B is the 1990 time-frame, and Phase C is the 1995 time-frame. Seven functional sets of devices are shown: preprocessor, signal conditioner, signal processor, general purpose processor, tape storage, disk storage, and operator console.

The top three rows of parameters in the figure indicate the number of *PEs* (processing elements) in each functional set for each phase of development (note that for tape and disk storage the "PEs" refers to storage devices). The notation "x(@y)" means "x" devices, each with "y" times the capability of the similar device used in the previous phase (as a result of technology insertion).

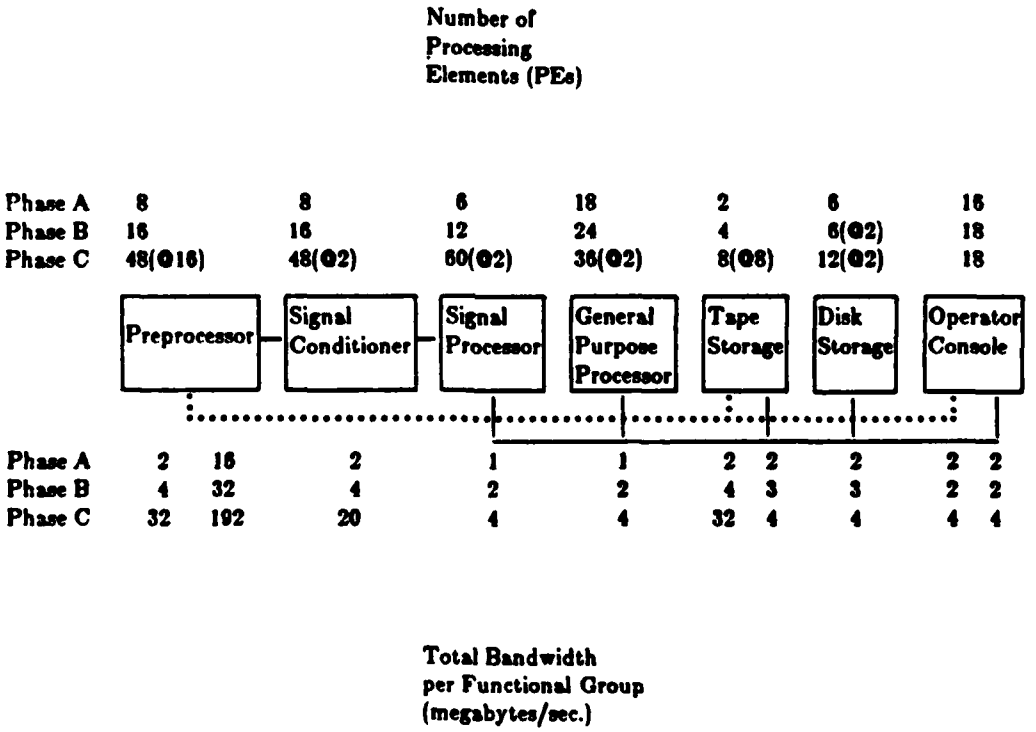


Figure 9.1:
Signal data transfer parameters for three iterations
of an evolutionary distributed processing system.

There are two classes of data communication paths. The solid lines are used for data processing. The dotted lines are used for the preprocessor to send the input data signals to: (1) tape storage to create history files for possible later off-line processing, and (2) the operator console for monitoring purposes.

The parameters in rows four through six indicate the expected average "total bandwidth per functional set" for each of the three phases. The numbers correspond to the connection above them represented by an arrow in the figure. The average bandwidth per PE in a functional set is calculated by dividing the total average bandwidth by the number of PEs. For example, the preprocessor to signal conditioner bandwidth is 2 Mbytes/sec per PE for both Phases A and B. The peak bandwidth is approximated by two times the average bandwidth. For example, the preprocessor to signal conditioner bandwidth is 4 Mbytes/sec per PE for both Phases A and B.

Our study of the data transfer network for this system will focus on the preprocessor to signal conditioner to signal processor communications. The distance between the preprocessor and signal conditioner, as well as between the signal conditioner and signal processor, is expected to be on the order of five feet. These functional sets will most likely share a single cabinet. The entire system will most likely fit in a rectangular area of approximately 40 feet by 60 feet.

The following are assumptions used in later sections about the expected data communications between the preprocessor and signal conditioner, and between the signal conditioner and signal processor. Note that for this study the network is not required to provide communications among the PEs within a functional set.

The communication patterns between the PEs in adjacent functional sets is predetermined before execution begins. Thus, each PE in a functional set knows to which PE(s) to send data in the next functional set. In case of faults in the system, once the fault is detected the system control program reallocates tasks to the PEs and modifies the associated connection requirements. Each relevant PE's program is updated, appropriate program rollback and restart procedures are performed, and execution continues.

Communications between functional sets will be from a fixed group of four PEs in the sending set to a fixed group of four PEs in the receiving set. This is demonstrated in Figure 9.2. The communication between a group of four sending PEs and a group of four receiving PEs can be one-to-one, many-to-one, or one-to-many. The one-to-one implies each sending PE is connected to only one receiving PE (and so each receiving PE is connected to only one sending PE). Note that the pairing of a sending PE to a receiving PE is arbitrary. This one-to-one pairing is expected to be the predominant mode of operation. The many-to-one connection implies that more than one PE in the sending group transmits data to the same PE in the receiving group, i.e., two-to-one, three-to-one, or four-to-one. This mode would be used in case there are one or more faulty PEs in the receiving group, or if the computational task being done by the sending PEs required the use of multiple PEs in order to prepare data for a single PE in the next functional set. This mode is expected to occur infrequently. The one-to-many connection implies that one PE in the sending group transmits data to multiple PEs in the receiving group, i.e., one-to-two, one-to-three, or one-to-four. This mode would be used in case there are one or more faulty PEs in the sending group, or if the computational task being done by the receiving PEs required the use of multiple PEs in order to process data

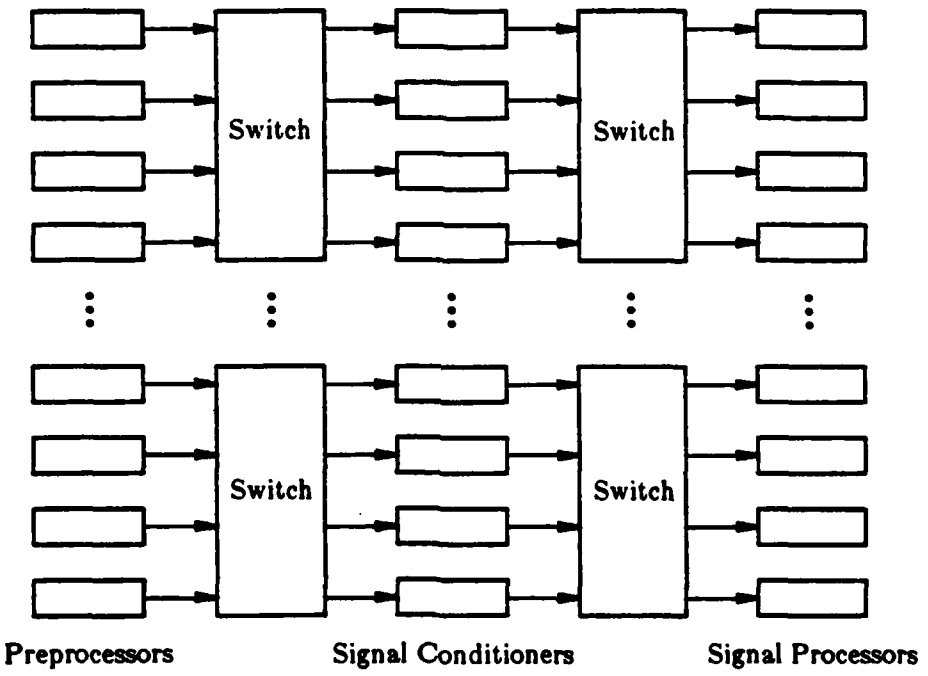


Figure 9.2:
A signal data switching configuration for front end PEs.

from a single sending PE. This mode is also expected to occur infrequently. The modes can be combined, e.g., a two-to-one connection, a one-to-two connection, and a one-to-one connection (between one pair of PEs) can be established simultaneously if no PEs are faulty. To summarize the connection patterns between functional groups: the most common mode of operation expected is the one-to-one pattern among four sending PEs and four receiving PEs (arbitrarily paired), but the network should also be capable of efficiently supporting one-to-many and many-to-one connections, as well as combinations of all three patterns.

Data transfer between PEs in different functional sets will be overlapped with computation. Consider the example shown in Figure 9.3, where each PE is connected to one PE in the next functional group. Shown below each PE is its three bank *swinging buffer* memory: one bank for data currently being operated upon, one bank for storing data previously generated by that PE (and currently being sent to the next PE), and one bank to receive data currently being sent by the previous PE (for processing after the current data set has been processed) [Dem83]. Each bank is a physically separate memory of 64K words. Thus, each PE is effectively sending a data set, processing a data set, and receiving a data set simultaneously. For example, consider the data sets in the figure using the signal conditioner PE's swinging buffers. Data set E is being sent by the preprocessor PE (which previously generated it) to the signal conditioner (which will process it after it finishes processing data set D). Data set D is currently being processed by the signal conditioner PE. Data set C is being sent from the signal conditioner PE (which previously generated it) to the signal processor PE (which will process it after it finishes processing data set B). The transmission of data sets A, C, E, and G, and the processing of

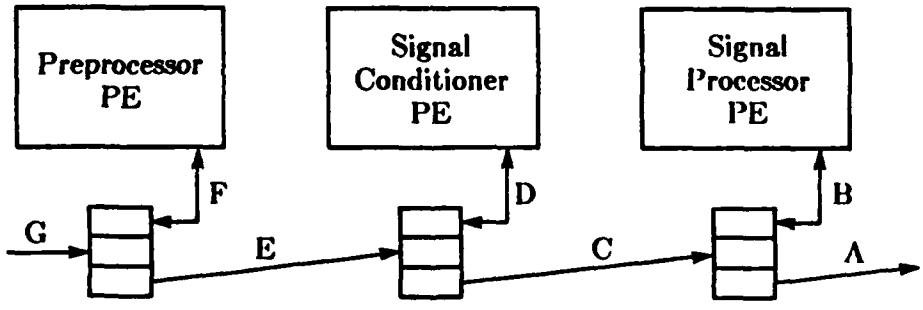


Figure 9.3:
PEs and their associated swinging buffered memories.

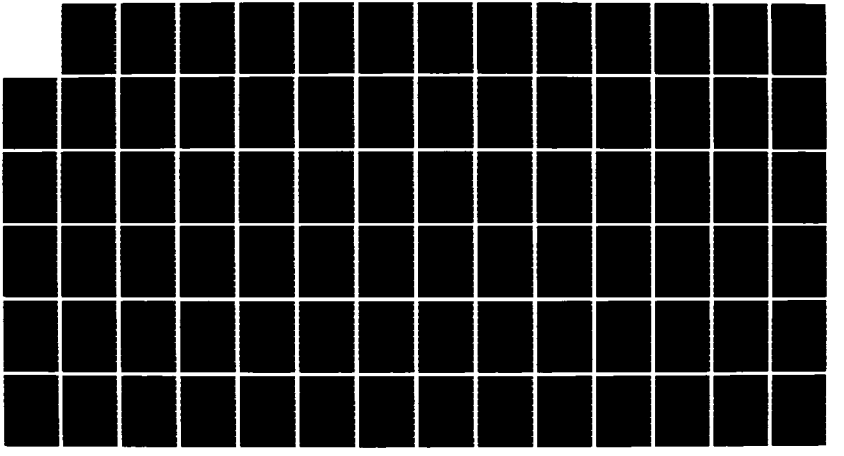
data sets F, D, and B, are all occurring simultaneously. The time to perform these simultaneous transmissions and computations is called an *interval.* In the next interval, data sets B, D, and F will be transmitted, and data sets A, C, and G will be processed. Similarly, in the interval prior to the one shown in the figure, data sets A, C, and E were processed, and data sets B, D, and F were transmitted. In summary, data sending, receiving, and processing occurs simultaneously for the PEs, as shown in Figure 9.3, through the use of three-way swinging buffers, and an interval is the time required for a PE to simultaneously receive a data set, transmit a data set, and process a data set (such as the signal conditioner PE does with data sets E, C, and D, respectively, in the figure.) It is assumed that, in general, the time to process a data set is longer than the time to transmit or receive a data set, and therefore determines the length of the interval.

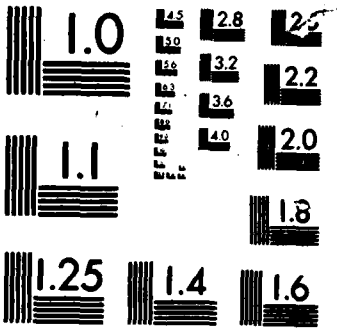
The amount of data sent by a single PE is expected to be a block of a minimum of 1K words and a maximum of 64K words. A number of source PEs can send data to any of the destination PEs (as specified by the connectivity); each destination PE, however, receives data from at most one source at any given time. Multiple sources send data to a common destination in a multiplexed fashion (in a predetermined static way) so that each source can send its data without contention.

In summary, the data communications will be between the "swinging memory buffers" associated with the PEs in the system. The requirements are that communications will be among groups of four source PEs and four destination PEs. Four approaches were considered: multistage based networks, ring based networks, shared bus, and crossbar based networks. As a result of the analyses presented in [SiM84], crossbar based networks were selected as the

AD-A167 336 DISTRIBUTED COMPUTING FOR SIGNAL PROCESSING: TOPOLOGICAL PROPERTIES OF IN..(U) PURDUE UNIV LAFAYETTE IN R R SEBAN DEC 85 ARO-18790.17-EL-APP-E 3/3

UNCLASSIFIED DAAG29-82-K-0101 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

method of choice for data communications in the system. Multistage ($\log_2$N stage) networks, such as the cube [AdS82b, SiM81], were not chosen because to establish connections between just four (or eight) source PEs and four (or eight) destination PEs the crossbar is more flexible, and, given current technology, cost-effective. Ring based networks were eliminated because if most communications are 1:1 (one source PE to one destination PE) and occur simultaneously, the parallel paths provided by the crossbar make it more suitable. Shared bus networks were eliminated because the PE's swinging buffers could not load data onto them fast enough for them to operate at the desired Phase B bandwidth. In this work, the characteristics of a crossbar chip, the organization of these chips for fault tolerance, and the way in which the crossbar based network can be interfaced to the processors are described. An 8-by-8 design is proposed instead of a 4-by-4 design to provide extra load balancing capabilities when faults occur.

9.2 Overview

In Section 9.3 the problem is informally defined. In Section 9.4 the basic terms are defined. In Section 9.5 the buffer to network interface is discussed. Several architectures at the chip level are analyzed in Section 9.6. Section 9.7 evaluates four network architectures. Finally, in Section 9.8, various fault detection and recovery methods (at the system level) are presented. In Section 9.9 the conclusions are presented.

9.3 Problem Definition

The network design presented is based on the preprocessor/signal conditioner communication requirements. These requirements are greater than those for the signal conditioner/signal processor, both in terms of throughput and number of processors. However, they are similar enough that it appears best to use the same network design in both cases. Since the preprocessor/signal conditioner requirements are stricter, these will be used to guide the network design. Any capabilities included for the preprocessor/signal conditioner communications but not needed for the signal conditioner/signal processor communications can be adapted to provide additional fault tolerance.

The problem is to design an interconnection network for a distributed signal processing system satisfying the following specifications. The specifications here are based on the expectations of the way in which such a system may operate.

In this section, the requirement of network extendibility to a larger number of PEs is described. The data communication is between a set of source processors and a set of destinations processors. The *set* corresponds to a common functional specifications, such as the set of processors used as preprocessors. The data movement is unidirectional from a source processor to a subset of destinations processors. The processors are addressed by distinct consecutive integers in each set separately. The communication requirements specify that a fixed *group* of four source PEs in one set be allowed to send data to a fixed *group* of four destination PEs in another set. The subset of four processors addressed by a, where

$$a = 4i + j \quad 0 \leq j \leq 3$$

will be called group i.

In Phase A (1985) the system will have two groups in the source set and two groups in the destination set. In Phase B (1990) the system will have four groups in the source set and four groups in the destination set. Phase C (1995) of the system will not be discussed here since we feel that technology will have changed so much by then that it is better to concentrate our efforts in this section on phases A and B. It is desirable for a single conceptual design to be applicable to both phases A and B.

In this paragraph the throughput requirements of the network are specified.

For phase A:

(a) Average: 2 Mbyte/sec/PE with 8 PEs active

(b) Peak: 4 Mbyte/sec/PE with 8 PEs active

For phase B:

(a) Average: 2 Mbyte/sec/PE with 16 PEs active

(b) Peak: 4 Mbyte/sec/PE with 16 PEs active

Since the physical distance between the set of sources and the set of destinations is expected to be on the order of five feet, it will be assumed that throughput of a single wire is 1 Mbyte/sec/wire. Using a crossbar based network, a 4-bit network word is sufficient to handle the peak load of each PE (4 Mbyte/sec) for both Phases A and B.

The following are the interconnection function requirements. The required data communication is only from group i of a set of source PEs to group i of a

set of destination PEs (four source PEs and four destination PEs). The functions are as follows within each group. PE k can send data to any subset of the destination group. Any number of source PEs can send data to any of the destination PEs as long as each destination PE is getting data from at most one source at the same time. (Note that when multiple sources send data to the same destination time division multiplexing is used so that each source can send its data without contention.) The amount of data sent by a single processor is expected to be a minimum 1K words and a maximum of 64K words.

The fault detection and recovery is a salient issue of the design. Soft faults are transient and temporary. An important requirement is that soft faults occurring in control messages (e.g., message header, chip control signals) will be detected. It is not as important to protect data information from soft errors, as they can normally be treated as additive noise. If desired, parity bits or error-detection/error-corrections bits could be added. Hard faults are permanent. Therefore, hard faults occurring in control and data communications must be detected as early as possible. In summary, the system should be able to recover from as many soft and hard errors as possible, perhaps with some loss functionality or throughput.

Another important requirement is that the cost of the implementation will be low. The cost categories are: (a) number of chips; (b) number of distinct types of chips; and (c) wiring complexity between the chips of the network.

9.4 Basic Concepts

In this section some terminology that is used throughout this work is presented.

Transmission dialog: the action of a processor transmitting all the data contents of its buffer to perhaps multiple destinations. A transmission dialog consists of a number of transmission blocks.

Transmission block: an uninterrupted transmission of ($\simeq$ 128 to 1K bytes), a component of a transmission dialog.

Data interconnection network: the hardware dedicated to the transmission of data from sources to destinations.

Report interconnection network: the hardware dedicated to the transmission of status and error reports from destinations to sources.

PE: processing element or processor.

DMA: direct memory access hardware – the hardware that controls the state of the network and is responsible for the details of the transmission dialog.

Source PE: processor designated so by being the source of data transmitted through the data network.

Destination PE: processor designated so by being the destination of the data transmitted through the data network.

SDMA: the DMA interfacing the source PE to the input of the data network.

DDMA: the DMA interfacing the output of the data network to the destination PE.

Network port: the input (output) pins of the network dedicated to a single processor.

Network path: the reconfigurable hardware between the source DMA of a single processor to the destination DMA of a single processor.

Network bit path: a single one-bit wide component of the network path.

Network word: the word consisting of the functioning bit paths (per network path).

System word: 16-bit word also called "word."

Input wire: is a wire connection from output of the source DMA to the I/O pin at the input of a chip of the network.

Output wire: is a wire connection from the I/O pin at the output of a chip of the network to the input of the destination DMA.

Middle wire: is a wire connection between the chips of the network.

Chip data line: is the path that data uses inside a network chip.

Chip control: is the path and logic the control uses inside a network chip.

9.5 DMA - Direct Memory Access

This section describes the specialized DMA chips or logic needed to interface the the swinging buffers to the network.

Here the source DMA functions (see Figures 9.4 and 9.5) will be described.

(1) Buffer interface: The logic that interfaces to the swinging output buffer.

(2) Data formatter: The conversion of 16 bit words into the network word. Network word width is determined by the number of nonfaulty bit paths per network port.

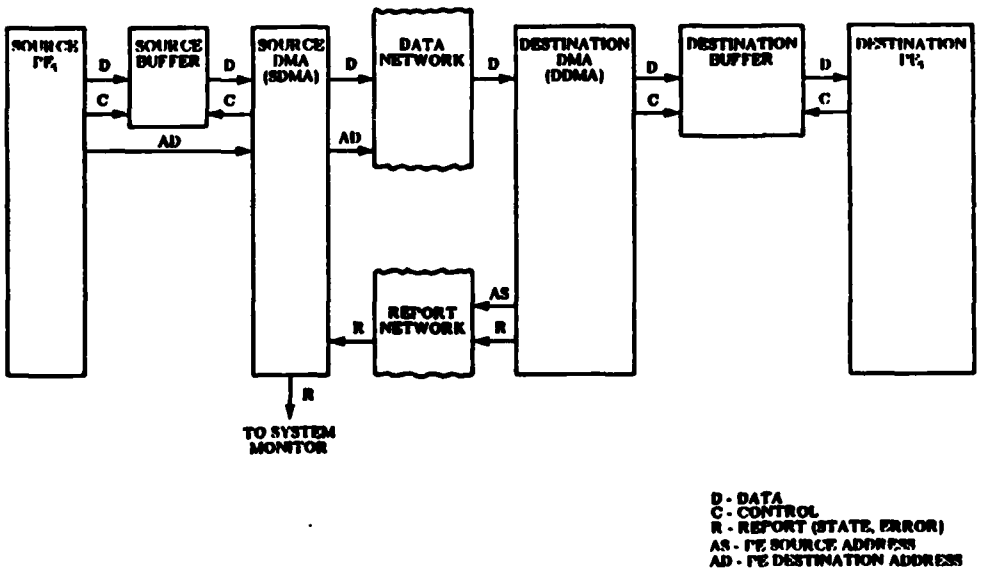


Figure 9.4:
The architecture of the communication system: D - data, C - control, R - report, AS - PE source address, AD - PE destination address.

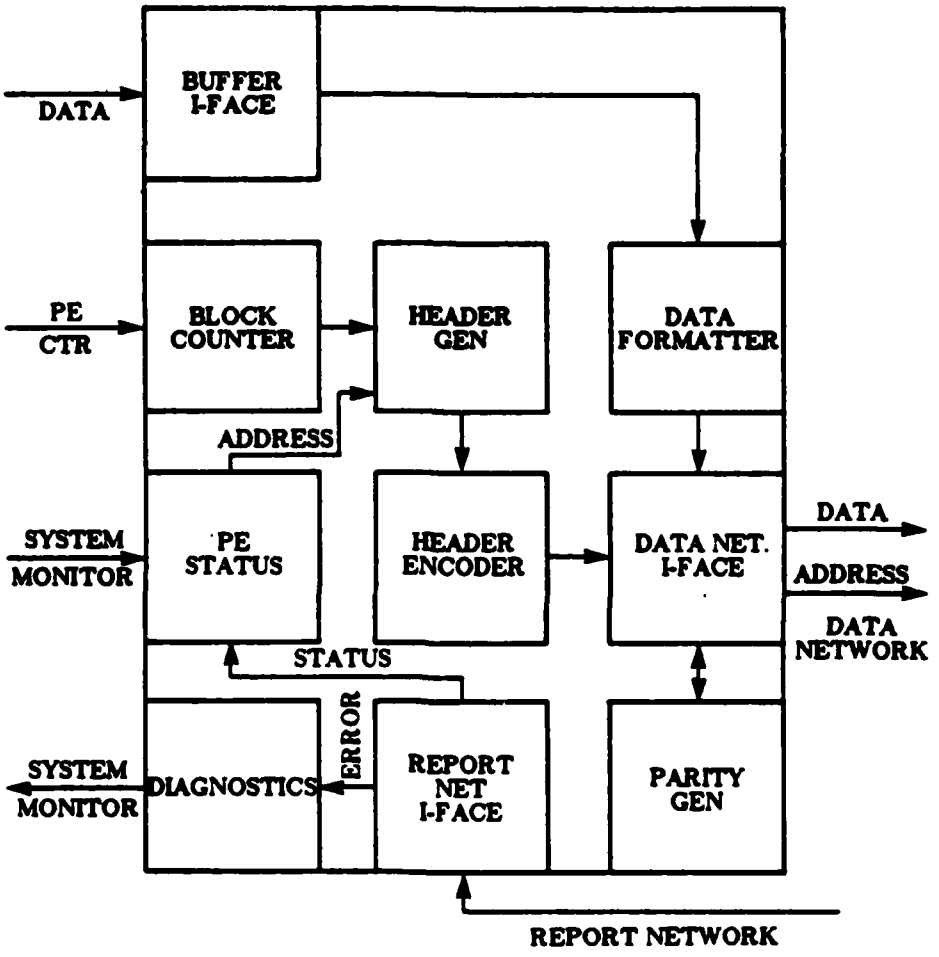


Figure 9.5:
Source DMA architecture.

(3) Block counter: The block counter is loaded by the source PE. It is normally initialized to the number of transmission blocks per dialog, and decremented each time a block is transferred. For diagnostic purposes the PE has read/write capabilities.

(4) PE status: The hardware contains a PE status table. The table consists of K registers. Register i contains the status of the destination processor i, $0 \leq i < K$ (where K is the number of processors in the group (see Section 9.3)). This table is used as follows. Every time a destination PE receives a block of data, it will return a status report. The system monitor can read the status table and monitor the correctness of the operations.

(5) Header generation: The header generation logic will construct a header as a triple (i,j,k), where i is the logical source address, j is the logical destination PE address, and k is the number of remaining blocks in the current transmission dialog. The fault tolerant extension of this minimal header is discussed in Section 9.8.

(6) Header encoder: The header encoder logic will encode the header using some error correction code (e.g., CRC) to protect against soft errors.

(7) Diagnostics: The diagnostic logic is used for diagnosis of the network. The SDMA periodically will try to test the network and all the destination processors for faults. The diagnostic logic will also report to the system monitor any destination PE that does not function properly.

(8) Parity generator: The parity generator will generate parity bits for each network word.

(9) Data network interface: The data network interface logic will send the network word (with optional parity) to the network input port.

Here the destination DMA functions (see Figures 9.4 and 9.6) will be discussed.

(1) Data network interface: This logic accepts data from the network output port.

(2) Parity check: This logic checks for correct parity of the network word.

(3) Data deformatter: This logic converts the data format from the network word format to the 16-bit word format.

(5) Header decoder: This logic decodes the header which was encoded by the error correction code at the source PE.

(6) Header check: This logic will check the source, destination, and block count fields for inconsistencies (this is discussed further in the Section 9.8).

(7) Soft/hard error: This logic will make the determination whether a soft or hard error occurred in the network. It will do so by counting parity errors and using information about header errors. If it is a hard error, the DDMA will notify the SDMA which will then reconfigure the bus or run some diagnostics to identify the exact error. The system monitor will be notified.

(9) Buffer interface: This logic sends the data which is now in the 16-bit word format to the destination buffer (see Figure 9.4.)

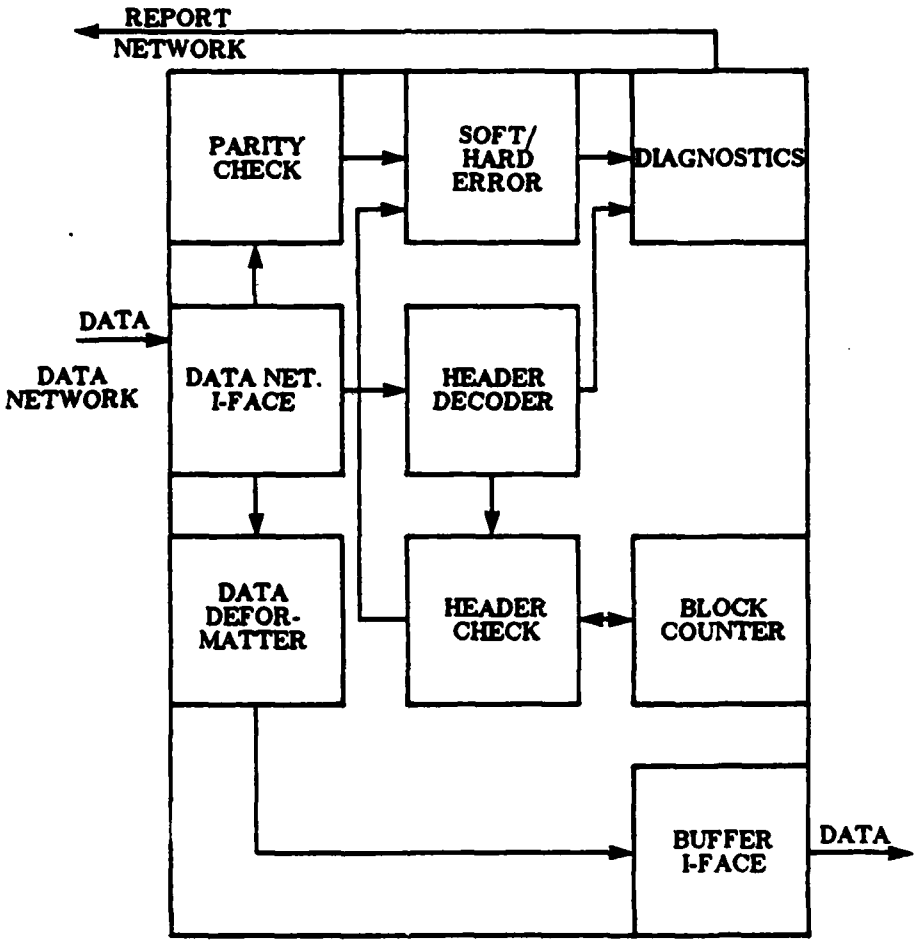


Figure 9.6:
Destination DMA architecture.

9.6 Architecture of the Fault Tolerant Crossbar

In this section two architectures of a fault tolerant crossbar will be presented, type I and type II. There are many ways to partition a n×m×k (n-input, m-output, k-bit wide) crossbar network into chips subject to available I/O pins and other constraints. One way to partition the network is to use *bit slicing*. Here the desired network is implemented using a number of network *planes*. Each plane would have the same number of interconnection ports but would have a smaller bit path. For example, a 4×4×8 crossbar can be implemented using this approach with four 4×4×2 crossbar chips as illustrated in Figure 9.7. A second approach is to build the larger network with a set of smaller networks. Here the desired network is obtained by essentially *cascading* a set of subnetworks. An example of how a 4×4×8 crossbar can be implemented using this approach with four 2×2×8 crossbar chips is illustrated in Figure 9.8.

Here the partition selected is based upon the important reliability criteria. The type I and type II chips are implemented as bit slices since that minimizes the number of chip-to-chip connections compared to the cascading approach. These connections slow down the signal and more importantly force the bit path through many soldering joints (an unreliable element).

Different chip architectures for n×m crossbars are discussed in [MaM81a, McT82]. Our design differs from these in order to support the communication requirements of this particular application. The differences include our lack of a collision detection mechanism (due to the deterministic nature of the communications), our addition of fault tolerance, and our use of serially loaded

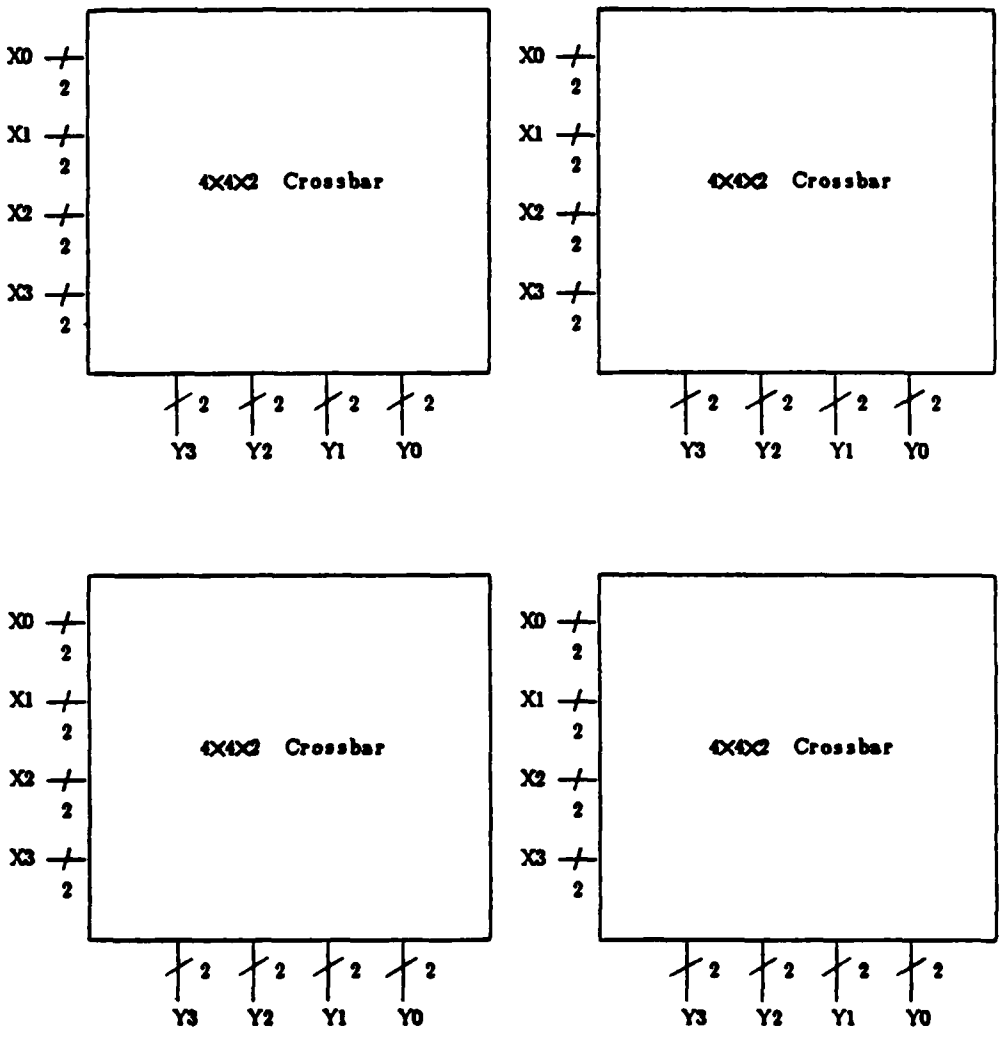


Figure 9.7:
Implementation of a 4×4×8 crossbar using bit slicing.

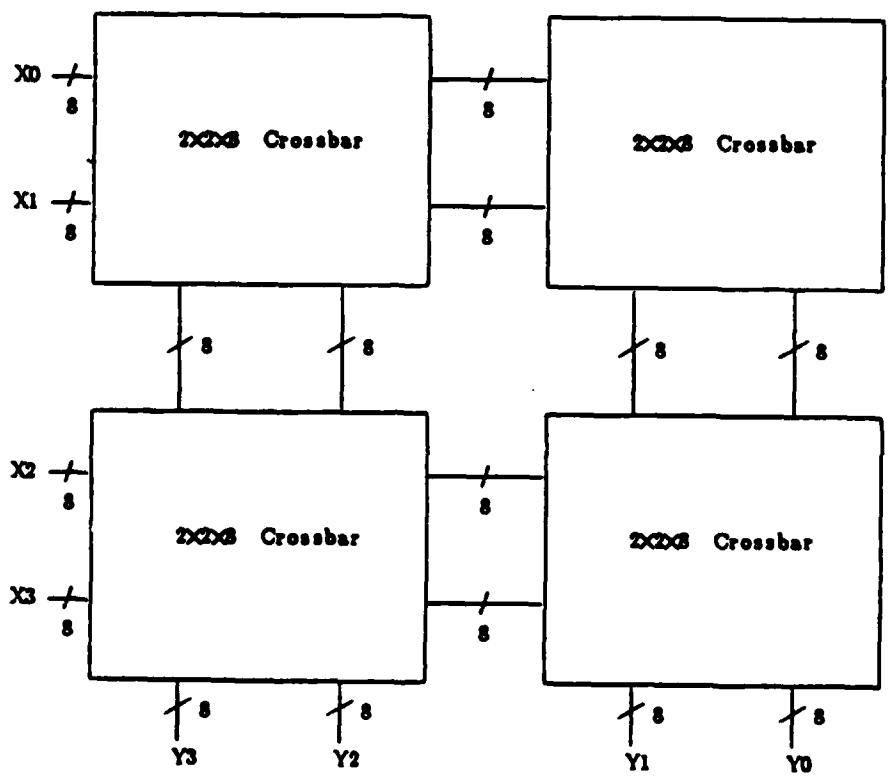


Figure 9.8:
Implementation of a 4×4×8 crossbar using cascading.

control bits (since the overhead is negligible compared to the length of the transmission block).

In this section the architecture of the $4\times4^{(1)}$ type I crossbar crossbar chip is described. The chip must satisfy the interconnection requirements described in Section 9.3. Also, it must be highly fault tolerant; for example:

(a) A faulty section can be localized and disconnected from the rest of the properly functioning chip.

(b) The pins available allow different methods of controlling the chip. It is up to the logic designer to decide which method satisfies any specific set of requirements.

(c) There are two paths for all data lines on the substrate.

In this system the interconnection functions are restricted to functions from a single group i of four source PEs to a single group i of four destination PEs, thus the pin limitation based design methodology discussed in [FrW81, FrW82] is not relevant since it applies to networks of size 64x64 or larger. Also, because here the concern is with 4x4 crossbars, the finite state automata type implementations as discussed in [WaF83] will not be applicable, especially since the fault tolerance of the implementation is the most important aspect of the design.

Figure 9.9 shows a block diagram of a type I chip. The pin functionality is as follows:

---

(1) 4x4 is described here for pedagogical reasons, however the design is applicable to rxr crossbars as well.

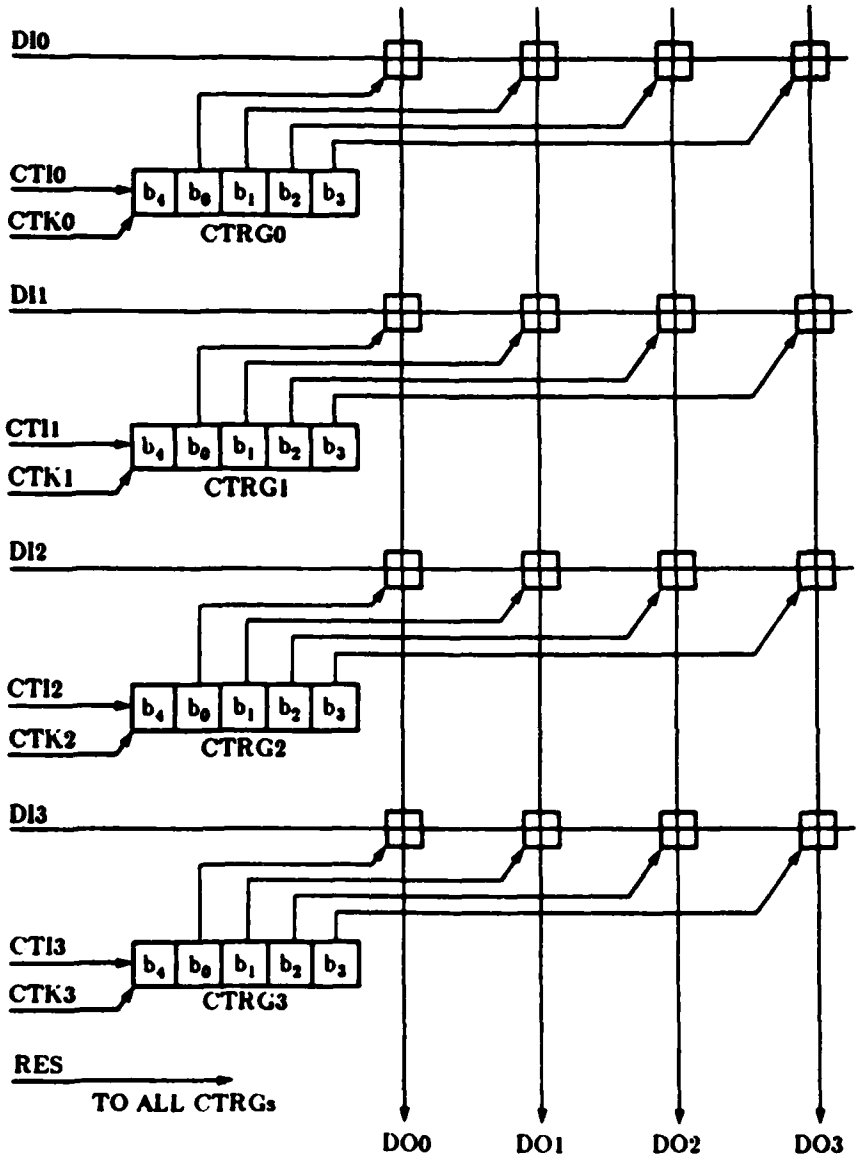


Figure 9.9:
Block diagram of a type I chip.

DI j: Data input for input port j

CTI j: Control register input for input port j

CTK j: Control clock input for input port j

DO j: Data output for output port j

RES: Reset input, will reset all CTRGs (control registers) to zero

V_{cc}: Power supply, two physically distinct pins (not shown on figure)

G: Ground, two physically distinct pins (not shown on figure)

The number of functional pins in a type I chip is as follows: input port i : three pins (DI i, CTI i, and CTK i); Output port j : one pin (DO j); and reset : one pin (RES). For a 4×4×1 crossbar the total number of signal (control and data) lines, which does not include RES, is $4\times4=16$ (4×N for an N×N×1 crossbar). For an 8×8×1 crossbar the total number of signal lines is $8\times4=32$. Assuming there can be up to 80 signal pins on a VLSI chip using VHSIC technology, four 4×4×1 crossbars, each with its own control and reset for fault tolerance purposes, can be implemented on a single chip, yielding a 4×4×4 crossbar. Similarly, an 8×8×2 crossbar chip could be constructed.

There are several methods of controlling the port.

(a) The processor that sends the data to an input port can be the same one that sets up the controls for that port (better from a reliability point of view).

(b) The chip control is given to the system control unit.

We will assume the processor sending the data controls the input port (i.e., sets the port's CTRG).

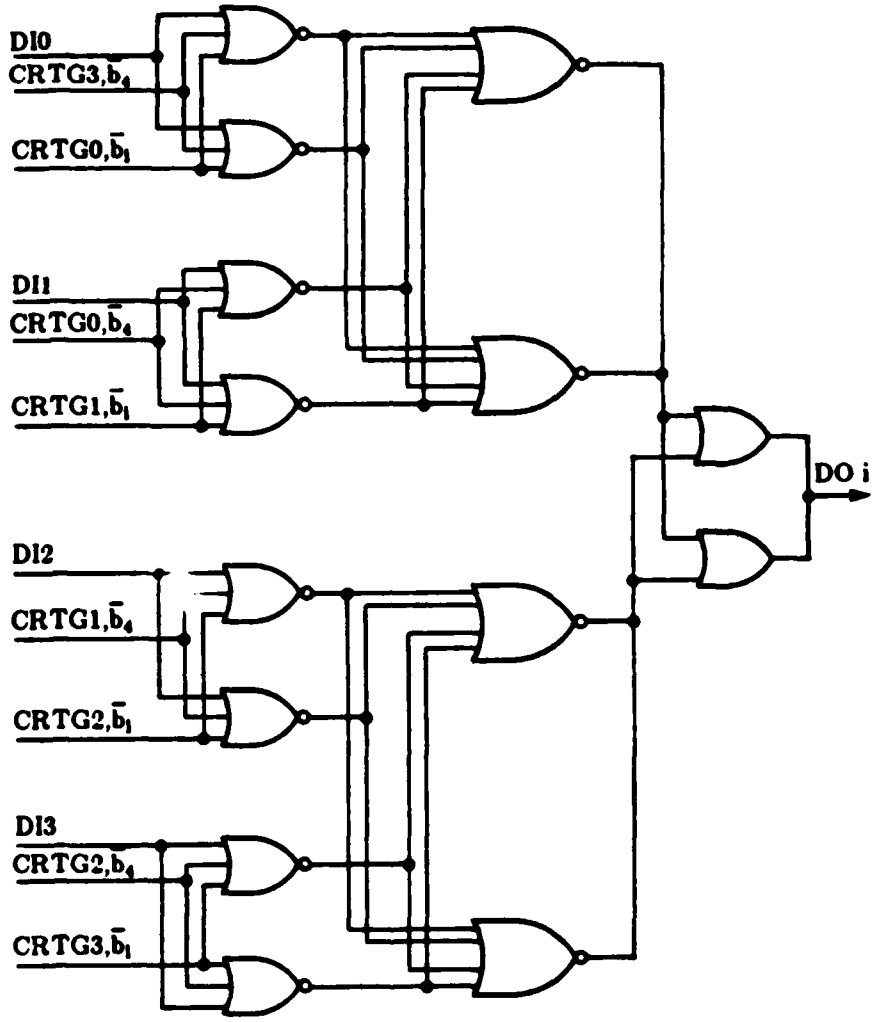
The port functionality of a type I chip is as follows. The CTRG j must be loaded with control information. Port j can be in one of the two states, the enabled state or the disabled state. If input port j is enabled, and $b_i = 1$ in CTRG j, for some fixed i, $0 \leq i \leq 3$, then the data from DI j will propagate to DO i. It is possible to have any subset of bits set in CTRG j. If input port j is disabled, then input port j data will not get propagated to any output port. A special control bit b_4 is in each CTRG for fault tolerance reasons. If $b_4 = 1$ in CTRG j then input port $j + 1$ modulo 4 is disabled. This allows a PE to "disconnect" another PE which is faulty. The usage of the b_4 bit is discussed later. There is no need for contention logic since the SDMA will know which destination processors are available (as discussed in Section 9.1).

In Figure 9.10 the data path for the output port i (DO i) is shown. For reliability reasons each gate is duplicated by a parallel gate with the same logic function. This method will protect the chip from an open gate fault (stuck low) since its parallel gate can carry the function alone. If a gate output is stuck on high it will cause loss of functionality of only part of the chip; the closer to the chip output that the gate is, the larger the part of the chip that will lose its functionality.

Although the possibilities to recover from faults are many, only a few will be discussed here to illustrate the main strong points of the design.

(1) Suppose a single gate in the crossbar chip is stuck at low in the data path, then the error will not exhibit itself because of the gate parallel to the faulty one.

(2) Suppose it is known that the input data path (external to the chip) is stuck on high, then the control of that port will load CTRG appropriately



Note: $\bar{b}_i$ means complement of bit i

Figure 9.10:
The data path for output port i (DO i).

and disconnect the data path. (Stuck on high means that the path is stuck in such way that the DO i connected to this input would be forced high.)

(3) Suppose it is known that input data path of input port j is stuck on high and also the control logic of that port is not functioning. Then the processor attached to input port $j-1$ modulo 4 can use its disable logic (b_4) to disable the faulty port j.

(4) If the input path is stuck at low, the functionality of the rest of the chip will not be impaired.

If the combined delay from the output of the SDMA to the input of the DDMA (see Figure 9.4) exceeds the desired clock cycle time, then the path has to be broken by a set of registers, one per port, allowing data to be pipelined through with shorter delays. When the crossbar chip is located physically near the source processors, then buffers should be placed at the output of the crossbar chip (on the chip itself). The decision to place the buffers at the outputs of the crossbar is based on the assumption that the delay from an output of the SDMA to an output of the crossbar chip is one half of the combined delay from the output of the SDMA to the input of the DDMA. In this system the combined delay is short therefore there is no need to break the path.

In this section the architecture of type II crossbar will be described. The type II crossbar (see Figure 9.11) is very similar to the type I implementation with exception of the following. The CTI i and DI i inputs are merged into a single pin. This results in a savings of $N \times b$ pins for an $N \times N \times b$ crossbar. The reliability has been compromised somewhat, however, because if DI i is stuck

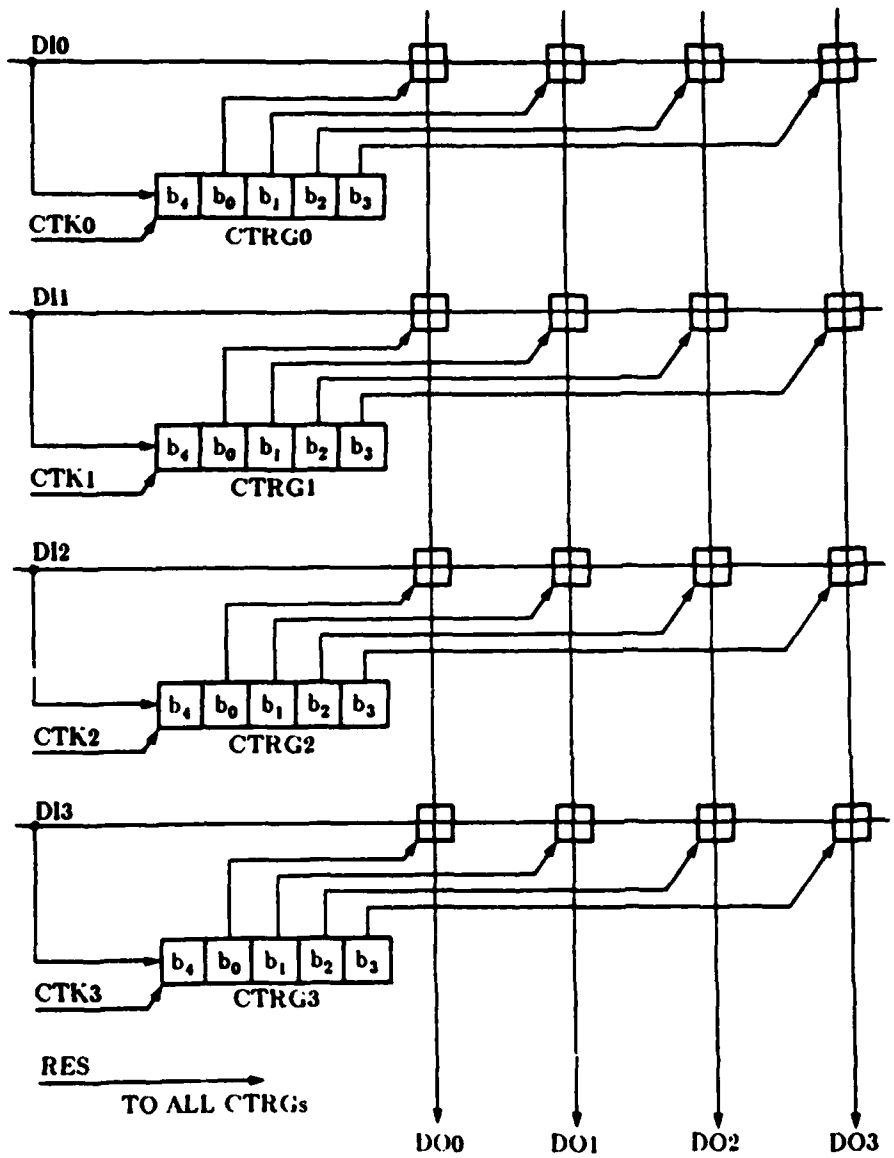


Figure 9.11:
Block diagram of a type II chip.

on one, the CTRG i cannot be loaded to get the DI i off the output bus DO j (if DI i is connected to DO j). However, it is still possible to get DI i off the output bus DO j by using the b_4 bit of CTRG i−1 modulo 4.

The number of functional pins in a type II chip is as follows. Input port i: two pins (DI i and CTK i). Output port j: one pin (DO j). Reset one pin (RES). For a 4×4×1 crossbar the total number of signal (control and data) lines (not including RES) is 3×4 = 12 (3×N for an N×N×1 crossbar). For an 8×8×1 crossbar the total number of signal lines is 3×8 = 24. Similar to the analysis for a type I crossbar design, assuming there can be up to 80 signal pins on a chip, a 4×4×6 or 8×8×3 crossbar can be constructed.

9.7 Network Architectures

Several different network architectures and their implementations using type I or type II crossbar chips will be presented in this section. Each scheme has sufficient throughput.

Each scheme will be evaluated using the following criteria.

(1) Types of interconnection functions admissible.

(2) Number of chips.

(3) Cost of connections between the chips of the network.

(4) Fault detection (hard faults).

(5) Fault recovery.

(6) Extendibility to a larger number of processors.

(7) Extendibility to larger bandwidth.

Although the required interconnection functions demand only a 4×4 crossbar, for the following reliability reasons an 8×8 crossbar will be used. Suppose two PEs fail or the paths to them fail in a single group j (of size four). This would cause the load on the two remaining PEs to double. Using an 8×8 crossbar it is possible to allocate one PE from group j + 1 and thereby balance the load over two groups (and their associated PEs).

The DMA network port consists of four bits which provides sufficient bandwidth (4 Mbyte/sec/PE) to meet the specifications in Section 9.3. This can be calculated as follows. Each PE has four-bit wide bus. Based upon the longest distance of the connections between source and destination PEs ($\simeq$ 5-10 ft.) a single wire can transfer approximately 1 Mbyte/sec. A bus width of four bits allows 4 Mbytes/sec. Now, consider the swinging buffer memory bandwidth. Since the output memories are capable of reading 2 bytes/100 ns ($\simeq$ 20 Mbyte/sec.) the memories, too, have sufficient bandwidth. The above calculation shows that each PE has available a network (and memory) bandwidth of up to 4 Mbyte/sec/PE, which satisfies the requirements for both Phases A and B.

Consider scheme 1 shown in Figure 9.12.

(1) Interconnection functions admissible: The functions admissible are the full crossbar functions.

(2) Number of chips required: Using chip type I or type II two chips are required.

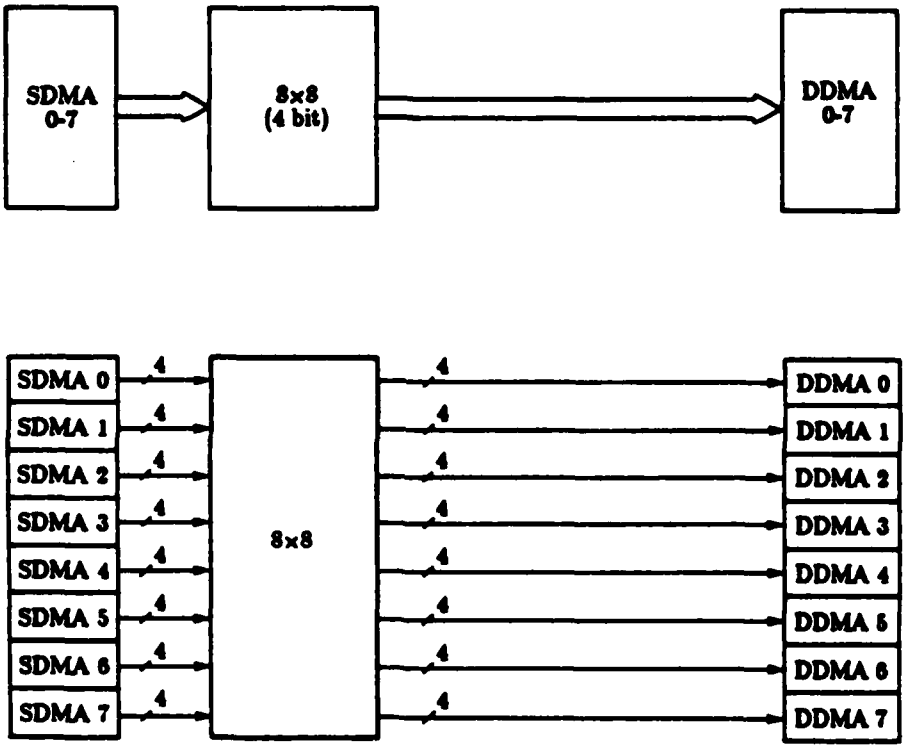


Figure 9.12:
Network architecture scheme 1.

(3) Cost of connections between chips of the network: Not applicable.

(4) Fault detection: The header method and diagnostics will detect multiple faults of the data path and also faults in the control, e.g., routing to an incorrect destination processor. (For more details see Section 9.8 on fault detection and recovery.)

(5) Fault recovery:

(a) If a bit path is broken either in the wires or on the chip, then the SDMA will reformat the network word and send it over the other correctly working bit paths. The DDMA will then deformat the network word into the system 16-bit word.

(b) If the control of a single bit path is not functioning, the fault will be handled as if the bit path is broken.

(6) Extendibility to a larger number of processors: Since the required interconnection functions can be partitioned (restricted) to groups of four processors, the scheme is easily extendible. Extension of the network can be accomplished by adding a complete interconnection network for each additional two source and destination groups (eight source processors and eight destination processors).

(7) Extendibility to a larger bandwidth: Since the bandwidth is limited by the number of wires per port, the extension simply involves increasing the number of wires per port and also the number of bit slices of the network. (This can be done up to the limit imposed by the swinging buffer bandwidth.)

Consider scheme 2, shown in Figure 9.13. This system consists of two complete networks in parallel. If there are no faults, only one of these

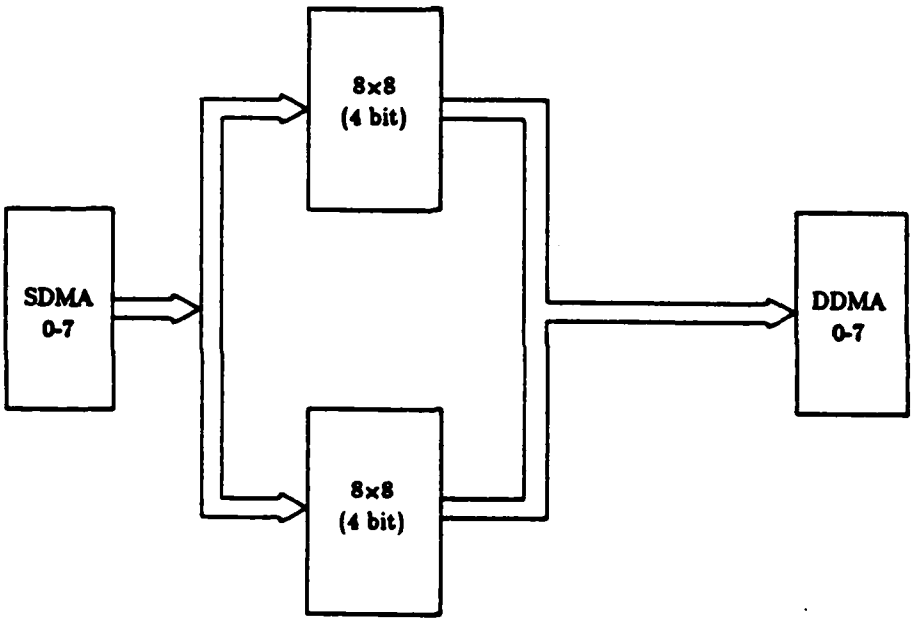


Figure 9.13:
Network architecture scheme 2.

networks is used. The outputs from the two networks are either selected by a multiplexer (with each bit path controlled independently) or by the tri-state logic inside the chips themselves. It is assumed that faults in either chip can be contained and will not affect the other chip.

(1) Types of interconnection functions admissible: Same as scheme 1.

(2) Number of chips required: Using chip type I or type II four chips are required.

(3) Cost of connections between the chips of the network: Connections are simple.

(4) Fault detection: Same as scheme 1.

(5) Fault recovery:

  (a) Same as 5(a) for scheme I.

  (b) If a bit path is broken inside one of the chips, then using the multiplexer (or tri-state control) the corresponding functioning bit path from the other network will be substituted.

  (c) If the control for a single bit path is not functioning, use the substitution as in (b).

(6) Extendibility to a larger number of processors: Same as scheme 1.

(7) Extendibility to larger bandwidth: Within a single network the same arguments as for scheme 1 hold.

Consider scheme 3, shown in Figure 9.14. The first (closest to the SDMA) part of the total network will be referred to as the front network. The second (closest to the DDMA) part of the network will be referred to as the rear network. The output port i of the front network and input port i of the rear network will be referred to as intermediate port i. If the assumption that long

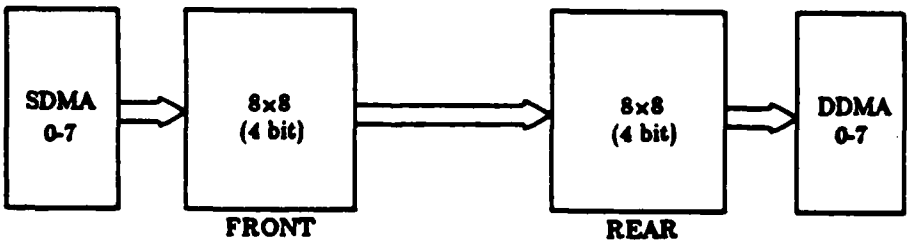


Figure 9.14:
Network architecture scheme 3.

wires are more susceptible to faults than short wires holds, then this scheme has some advantages.

(1) Types of interconnection functions admissible: Same as scheme 1.

(2) Number of chips: Using chip type I or type II four chips are required.

(3) Cost of connections between the chips of the network: Connections are simple.

(4) Fault detection: Same as scheme 1.

(5) Fault recovery: All techniques presented for scheme 1 can be used in addition to the following. Suppose source PE i wants to transmit to destination PE j. If a bit path is broken in the middle wire of port j it is possible to send data over the middle wires of intermediate port $k \neq j$ and then use the rear network crossbar to move the data from port k to output port j. Depending on the percent utilization of the paths, this may make system degradation negligible.

In this paragraph the scheme 4 will be described. It is possible to combine schemes 2 and 3 and get the benefits of both schemes. It will however involve four times more hardware than absolutely necessary from a connectivity and throughput point of view.

In this paragraph the network architecture for phase B will be presented. To construct the network for a system consisting of 16 source PEs and 16 destination PEs, the schemes 1 through 4 can be used as follows. For each set of eight source PEs together with eight destination PEs construct an independent network. That means that for phase B (16 source PEs, 16 destination PEs), there is one 8×8 network for source PEs 0-7 communicating with destination PEs 0-7 and there is another independent 8×8 network for

source PEs 8-15 communicating with destination PEs 8-15.

9.8 Fault Detection and Recovery

Three techniques for fault detection can be used: (1) parity generation and checking, (2) system run diagnostics, and (3) block header generation and checking during the normal mode of operation. Consider the latter two in more detail.

For system run diagnostics, the SDMA of processor i will either generate *(or use prestored) test patterns to test all the bit paths of the network.* It will send the patterns to all the destinations within the group and thereby test the data paths and controls of the network. The message will have the following format. At the beginning and the end of the block there will be a header containing the source field, destination field, opcode field, and block count. Some header formats and dialog techniques are discussed in [ThC83]. The scheme presented here is an augmented version of these formats for increased ease of fault detection. The opcode will say which diagnostic is being run. That will notify the DDMA for what it should specifically test. Some test patterns may follow the header, depending upon the particular diagnostic. The DDMA will analyze the header's destination field to check the control of the network. The DDMA will then send the error report to the SDMA. This is done through the report network (see Figure 9.4.)

The report network is an independent network used by the destination PEs to return status and error reports, or any information that the diagnostic routine requests. While the necessary bandwidth is low, for reliability reasons it should consist of at least four one-bit slices. Architecturally it is identical to the data network (that is, an 8×8 crossbar). It is important that the SDMA originating the diagnostic gets the error report even if the report network is not completely operational. This will be accomplished by trading throughput for redundancy in the information. Basically the error report will be sent serially over each of the bit paths belonging to the particular port being tested. For the error report to get back to the testing SDMA it is then sufficient if only one bit path in the report network is non-faulty. (The SDMA will analyze the header of the report message sent by the DDMA and check it for correctness in a way similar to that used by the DDMA to check the header of the data message.) The error report itself should be encoded by multiple error correcting code, because soft errors in the error report could have catastrophic consequences. The reason why it is important for the testing SDMA to receive the error report is that it can then make the best decision about which hardware is faulty and should not be used. The more information that is available to the testing SDMA the less, but sufficient, amount of hardware will have to be reconfigured. The major philosophy here is that the detection of faults in the network as well as subsequent reconfiguration (discussed in the next section) is done locally, independent of the system monitor. The exact description of the error will be assembled and broadcasted to all the source PEs by the DDMA. For example, if destination j has bit path k broken, all the source PEs when sending data to the destination j will format their data in such a way as not to use bit path k. This describes only the flavor of possible

diagnostics and many more are possible. Further research is required in this area.

Block header generation and checking during normal mode of operation can be implemented as follows. Each block during a normal transmission dialog will contain a header of form (i,j,k,l,m), where

i is the source PE address,

j is the destination PE address,

k is the number of this block within the current transmission dialog,

l is the operation to be performed by the destination PE on the data,

m is the multiple error correction code on the header.

First, the SDMA sets up the path in the network to the proper destination. Then the header will be sent on each of the bit paths at the source port to the DDMA. The DDMA will receive the header (actually multiple headers, one on each bit path). Trivially, the DDMA will discover any broken bit path. It will also discover any faulty network controls by examining the destination field. If the network is implemented as independent slices, it is possible that only some of the bit paths have bad control which will be discovered by the destination field. The block number can be used as follows. The DDMA maintains the last received block count in a register. By comparing the register with the incoming block number, it will discover faults such as lost blocks. The headers have to be soft error protected since they carry important information. The headers will be resent at the end of the block. If received correctly then, it will be assumed that data was transmitted correctly with the exception of soft errors on the data which will be ignored and treated as additive noise.

In this section several fault recovery techniques will be discussed. Some of the techniques may be applicable to only some network architectures and/or implementations. The possible hard faults can be classified as follows.

(1) Bit path in an input or output wire breaks.

(2) Bit paths inside the network chip breaks.

(3) Bit path in a middle wire breaks

(4) Bit path inside the network chip is stuck on high or low.

(5) The control of some but not all bit lines (of a single path) are faulty and the destination port is not receiving all of its bits.

(6) The control of all the bit lines (of a single path) are faulty and the destination port is (a) not receiving any data or (b) receiving data destined for another processor.

(7) PE fault.

It can be seen in the section on fault detection (Section 9.8) that any of these faults are detectable by the header and status report during normal operation. The question of how to reconfigure the network will depend upon the network architecture. For more details, see the section on network architectures (Section 9.7).

When a fault occurs, it will be discovered by the DDMA at the next block transmission. The DDMA will then send an error report to the SDMA. The SDMA will start diagnostic routines to evaluate the exact nature of the fault (for example, a faulty bit path). The source and DDMA will then reconfigure their hardware (for example, format the network word to skip the faulty bit path). At this time the SDMA will also notify the system monitor about the new reconfiguration. The system monitor does not have to be involved in the

network reconfiguration, it will just notify the operator that it occurred.

9.9 Conclusions

For this application, and given current and near future technology, a crossbar based interconnection network is very well-suited to the task under consideration. Two different fault tolerant chip architectures were presented. Four network architectures were designed and their characteristics described. Several fault detection and recovery techniques on the system level were shown, since the fault tolerance is a salient issue of this system.

10 SHUFFLING WITH THE ILLIAC AND PM2I SIMD NETWORKS

10.1 Introduction

Parallel computation is one way to take advantage of the low-cost processing power made possible by VLSI technology. The SIMD mode of parallelism has been successfully exploited in a number of problem domains. A critical architectural feature of a large-scale SIMD system is the interconnection network. A variety of networks have been proposed and analyzed [Sie79a]. The choice of which network to implement in a system is a function of factors such as the intended computational environment (i.e., task domain) for the system, construction time and cost constraints for building the system, and the capabilities of the interconnection networks. One of the ways in which to measure the capabilities of a network is to examine its ability to do different data permutations. Here, the abilities of two single stage networks to perform the "shuffle" data movement are evaluated.

This paper extends SIMD interconnection network studies presented in [Sie77, Sie79b]. In particular, the ability of the PM2I and Illiac single stage SIMD machine interconnection networks to perform the shuffle interconnection is examined. Two algorithms for an SIMD or multiple-SIMD machine with the PM2I network to perform the shuffle are given. One algorithm is used in the event that the SIMD machine is of the same size (in terms of number of processors) as the shuffle to be emulated. The other algorithm is used when the shuffle to be performed is of smaller size than the given machine with the PM2I network. It is proven that both algorithms require only one more network transfer than the previously published lower bound (which is $\log_2 S$ for a shuffle on S elements [Sie77]). The PM2I algorithm is used as basis for an

algorithm to do the shuffle with the Illiac network in $(2\sqrt{N})-1$ transfers. A lower bound of $2\sqrt{N}-4$ on the emulation of the shuffle using the Illiac network (and a different algorithm to perform the emulation) is presented in [NaS80].

10.2 Overview

In Section 10.3 the basic concepts are presented. In Section 10.4 an overview of the interconnection networks Illiac, PM2I and, Shuffle-Exchange is given. In Section 10.5 two algorithms of PM2I performing the shuffle are developed as well as proven correct. This is used as a basis for the algorithm for performing the shuffle with the Illiac network which is presented in Section 10.6. In Section 10.7 the conclusions are presented.

10.3 SIMD Machines

Typically, an *SIMD (single instruction stream - multiple data stream) machine* [Fly66] is a computer system consisting of a control unit, N processors, N memory modules, and an interconnection network (e.g. Illiac IV [BoD72]). The control unit broadcasts instructions to the processors, and all active

processors execute the same instruction at the same time. Each active processor executes the instruction on data in its own memory module. The interconnection network provides for communications among the processors and memory modules. A *multiple SIMD system* is a parallel processing system which can be structured as one or more independent SIMD machines, each with its own control unit (e.g. MAP [Nut77]).

One way to configure an SIMD machine is as a set of N processing elements (PEs) interconnected by a network, where each *PE* consists of a processor with its own memory. This is shown in Figure 10.1 and is called the PE-to-PE organization. An alternative organization is to position the network between the processors and the memories. The PE-to-PE paradigm will be assumed, however, the results presented will be applicable to the other organization also.

The model of an SIMD machine presented in [Sie79b] is used here. The assumptions made about the SIMD machine to be used as the model are intentionally minimal so that the material presented is applicable to a wide range of machines.

There are *N* PEs, addressed (numbered) from 0 to N−1, where $N = 2^m$. It is assumed that the processor contains a fast access general purpose register A and a *data transfer register (DTR)*. When data transfers among PEs occur, it is the DTR contents of each PE that are transferred. The notation "A ← DTR" means the contents of the DTR are copied into the A register. The notation "A ⟷ DTR" means the two registers exchange their contents.

The *PE address masking* scheme uses an m-position mask to specify which PEs are to be activated [Sie77]. Each position of the mask will contain either a 0, 1, or X ("don't care"). The only PEs that will be active are those that

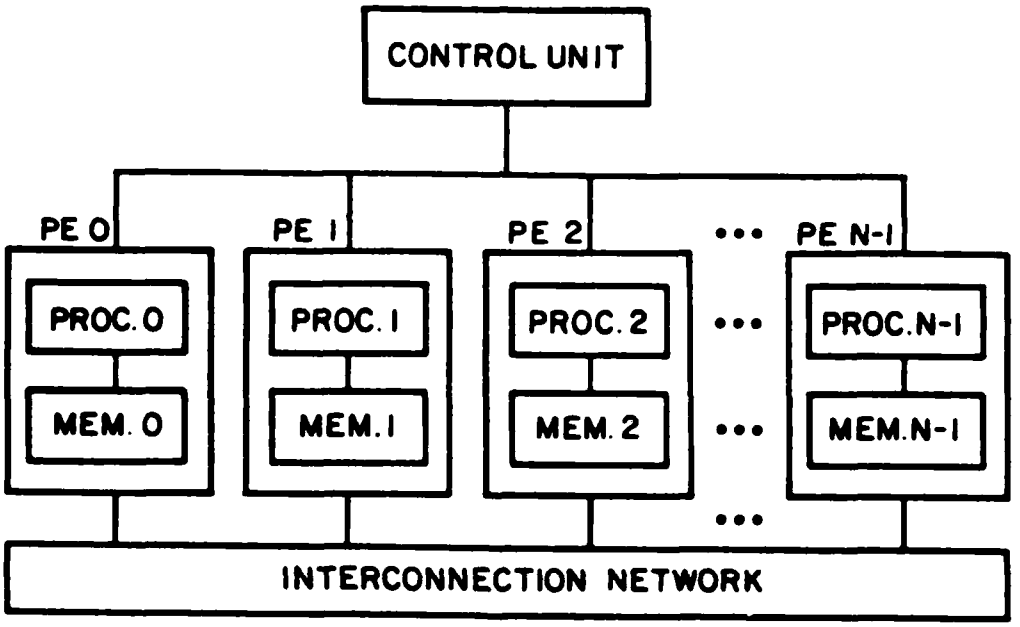


Figure 10.1:
PE-to-PE SIMD machine configuration, with N PEs.

match the mask in each position: 0 matches 0, 1 matches 1, and X matches 0 or 1. For example, if N = 8 and the mask is 1X0, then only PEs 6 = 110 and 4 = 100 are active. Superscripts are used as repetition factors, e.g., $X^3 01^2$ is XXX011. Square brackets will be used to denote a mask. Each PE instruction and interconnection function (defined below) will be accompanied by a mask specifying which PEs will execute that command.

An *interconnection network* can be described by a set of interconnection functions, where each *interconnection function* is a bijection (permutation) on the set of PE addresses [Sie77]. When an interconnection function f is applied, PE i sends the contents of its DTR to the DTR of PE f(i). This occurs for all i simultaneously, for $0 \leq i < N$ and PE i active. Saying that an interconnection function is a bijection means that every PE sends data to exactly one PE, and every PE receives data from exactly one PE (assuming all PEs are active). In this model, it is assumed that an inactive PE can receive data, but cannot send data. To pass data from one PE to another PE a programmed sequence of one or more interconnection functions must be executed, moving the data by a single transfer or by passing the data through intermediary PEs. Since there is a single instruction stream in an SIMD machine, all active PEs must use the same interconnection function (connection) at the same time.

10.4 The Interconnection Networks

The following notation will be used: let $N = 2^m$, let the binary representation of an arbitrary PE address P be $p_{m-1}p_{m-2}\cdots p_1p_0$, and let $\bar{p}_i$ be the complement of p_i. It is assumed that $-j \bmod N = N-j \bmod N$, for $j > 0$.

The *Illiac* network consists of the four interconnection functions:

$$\begin{aligned}
\text{Illiac}_{+1}(P) &= P+1 \bmod N \\
\text{Illiac}_{-1}(P) &= P-1 \bmod N \\
\text{Illiac}_{+n}(P) &= P+n \bmod N \\
\text{Illiac}_{-n}(P) &= P-n \bmod N
\end{aligned}$$

where $n = \sqrt{N}$ is assumed to be an integer. For example, if $N = 16$, $\text{Illiac}_{+n}(0) = 4$. The network is shown for $N = 16$ in Figure 10.2. This network was implemented in the Illiac IV SIMD machine [BoD72], and is included in the MPP [Bat80] and DAP [Hun81] SIMD systems. Illiac network capabilities are discussed in [BoD72, Orc76, Sie77, Sie79b, Sie80].

The *Plus-Minus* 2^i *(PM2I)* network consists the 2m interconnection functions:

$$\begin{aligned}
\text{PM2}_{+i}(P) &= P+2^i \bmod N \\
\text{PM2}_{-i}(P) &= P-2^i \bmod N
\end{aligned}$$

for $0 \leq i < m$. For example, $\text{PM2}_{+1}(2) = 4$ if $N > 4$. Figure 10.3 shows the PM2_{+i} interconnections for $N = 8$. Diagrammatically, PM2_{-i} is the same as PM2_{+i} except the direction is reversed. A network similar to the PM2I is used in the "Novel Multiprocessor Array" [OkT82] and is included in the network of the Omen computer [Hig72]. The PM2I connection pattern forms the basis for the data manipulator [Fen74], ADM [AdS82a, McS82], and gamma [PaR82]

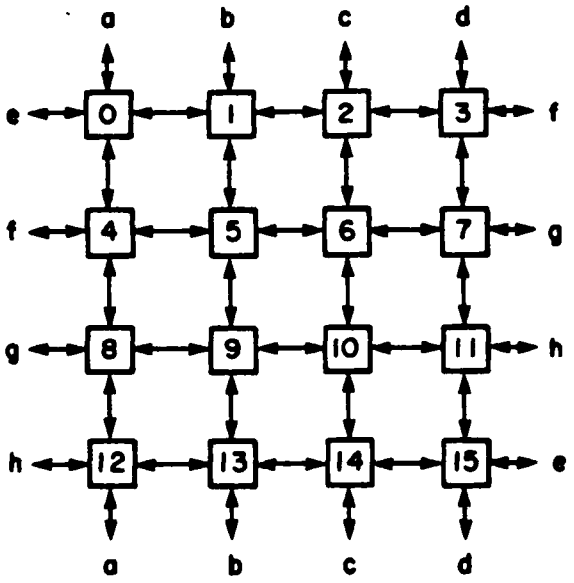


Figure 10.2:
Illiac network for N = 16. (The actual Illiac IV SIMD machine had N = 64). Vertical lines are $+\sqrt{N}$ and $-\sqrt{N}$. Horizontal lines are +1 and −1.

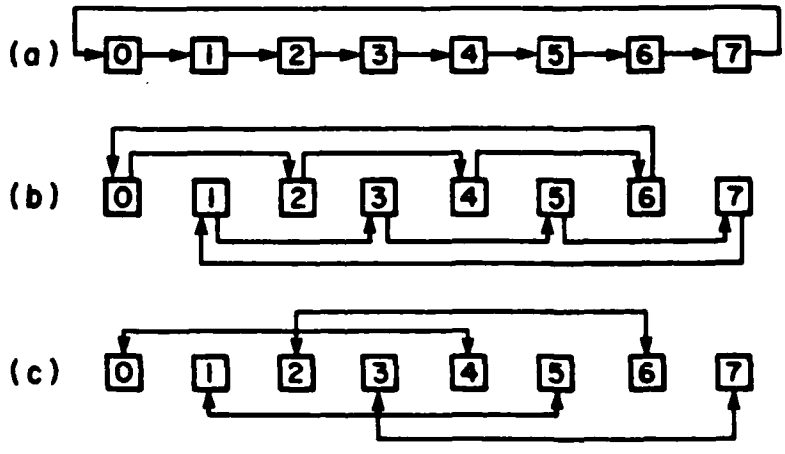


Figure 10.3:
PM2I network for N = 8. (a) $PM2_{+0}$ connections.
(b) $PM2_{+1}$ connections. (c) $PM2_{+2}$ connections.
For the $PM2_{-i}$ connections, $0 \leq i \leq 2$,
reverse the direction of the arrows.

multistage networks. Various properties of the PM2I are discussed in [FiF82, PrK80, Sie77, Sie79b, Sie80].

The *Shuffle-Exchange* network consists of the *shuffle* interconnection function and the *exchange* interconnection function:

$$\text{shuffle}(p_{m-1}p_{m-2}\cdots p_1p_0) = p_{m-2}p_{m-3}\cdots p_1p_0p_{m-1}$$

$$\text{exchange}(p_{m-1}p_{m-2}\cdots p_1p_0) = p_{m-1}p_{m-2}\cdots p_1\bar{p}_0.$$

For example, shuffle(3) = 6 and exchange(6) = 7, for $N \geq 8$. This network is shown in Figure 10.4 for N = 8. The shuffle is also included in the networks of the Omen [Hig72] and RAP [CoG74] systems. The multistage omega network is a series of m Shuffle-Exchanges [Law75]. Features of the Shuffle-Exchange are discussed in [ChL81, FiF82, Lan76, LaS76, NaS81, NaS82, PrK80, Sie77, Sie79b, Sie80, Sto71, WuF81].

The ability of each of the PM2I and Illiac networks to perform the exchange function in just two transfers was presented in [Sie79b]. Thus, the algorithms given here for performing the shuffle can be used to allow either the PM2I or Illiac network to emulate the Shuffle-Exchange network.

10.5 Shuffling with the PM2I Network

In this section the use of the PM2I network to perform the shuffle will be examined. Two algorithms for an SIMD or multiple-SIMD machine with the PM2I network to perform the shuffle are given. One algorithm, presented in this section, is used in the event that the SIMD machine is of the same size (in

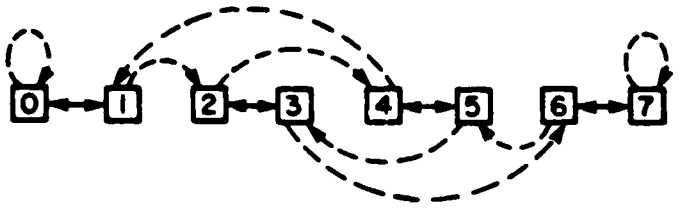


Figure 10.4:
Shuffle-Exchange network for N = 8.
Dashed line is shuffle, solid line is exchange.

terms of number of PEs) as the shuffle to be emulated. The other algorithm described is used when the shuffle to be performed is of smaller size than the given machine with the PM2I network. If the shuffle is of size S (in terms of number of PEs) then it was shown previously in [Sie77] that the lower bound of the algorithm for the PM2I to emulate the shuffle requires $\log_2 S$ network transfers. It is proven here that both algorithms require only $\log_2 S + 1$ network transfers.

In this section an algorithm to perform the shuffle with a PM2I of the same size will be developed. This algorithm applies to the case where the machine with the PM2I network is of the same size in terms of the number of processors as the shuffle to be emulated. The following ground rules will be used in the design and analysis of the algorithm.

(1) The model and definitions presented in Sections 10.3 and 10.4 will be the formal basis for the results.

(2) When simulating the shuffle, the data that is originally the DTR of PE P must be transferred to the DTR of PE shuffle(P), for all P, $0 \leq P < N$.

(3) The time for each algorithm is in terms of the number of executions of interconnection functions required to perform the simulation.

The reason for (3) can be seen by considering the way in which various instructions can be implemented. The instructions in the algorithm can be divided into three categories: control unit operations (in C), register to register operations (in I), and inter-PE data transfers (in F). Control unit operations, such as incrementing a count register in the control unit for a "for loop," can, in general, be done in parallel (overlapped) with the previously broadcast PE instruction, thus taking no additional time. Register to register operations

within a PE will probably involve a single chip or, at worst, physically adjacent chips. The inter-PE data transfers will involve setting the controls of the interconnection network and passing data among the PEs, involving board to board, and probably rack to rack, distances. Thus, unless the number of register to register operations is much greater than the number of inter-PE data transfers, the time for the inter-PE transfers will be the dominating factor in determining the execution time of the algorithm.

In the algorithm below ":" indicates a comment. When discussing the algorithm, "Li" is used as an abbreviation for "statement i of the algorithm." For $j = 0$, $X^j = X^0$ where "X^0" is the null string, i.e., no "X"s.

To understand the concept underlying the algorithm to perform the shuffle, consider the "distance" the shuffle moves a data item. The data item originally in the DTR of PE P, $0 \leq P < N/2$, is moved to shuffle(P) = 2P, a distance of shuffle(P) − P = P. The data item originally in the DTR of PE P, $N/2 \leq P < N$, is moved to shuffle(P) = 2P+1 mod N, a distance of shuffle(P) − P = P+1. This is shown in Table 10.1 for N = 8.
Specifically, data originally in PE P, $0 \leq P < N$, with $p_i = 1$ is moved by $PM2_{+i}$ ($i = 0, 1, ..., m - 1$) to PE 2P mod N. If $N/2 \leq P < N$ then in addition to the previous move the data will be moved +1 by $PM2_{+0}$ to 2P + 1. This is also shown for N = 8 in Table 10.1.

The difficulty in designing a parallel algorithm for this task arises from the need to keep track of the flow of N data items among the N PEs. Note that Table 10.1 does not show the intermediate PEs through which the data is passed. For example, for N = 8 after executing $PM2_{+0}$ the data originally in PEs 4 and 5 will both be in PE 5.

Table 10.1:
The idea underlying the algorithm for the PM2I to perform the shuffle, shown for N = 8.

origin PE number	distance moved by shuffle	distance moved by PM2I				
		$PM2_{+0}$	$PM2_{+1}$	$PM2_{+2}$	$PM2_{+0}$	Total
0 = 000	+0	–	–	–	–	+0
1 = 001	+1	+1	–	–	–	+1
2 = 010	+2	–	+2	–	–	+2
3 = 011	+3	+1	+2	–	–	+3
4 = 100	+5	–	–	+4	+1	+5
5 = 101	+6	+1	–	+4	+1	+6
6 = 110	+7	–	+2	+4	+1	+7
7 = 111	+0	+1	+2	+4	+1	+0

In the algorithm below, during steps L3 to L5, for $1 \leq j < m-1$, all of the data of interest are in even numbered PEs. After L5 has been executed for $j = m-1$, the data from PE P, $0 \leq P < N$, has been moved to PE 2P mod N by using a subset of $PM2_{+0}$, $PM2_{+1}$, ..., $PM2_{+m-1}$, in that order. For $N/2 \leq P < N$, L6 executes $PM2_{+0}$ to move data from PE 2P to 2P+1.

Algorithm to perform the shuffle with a PM2I network of the same size:

(L1) $A \leftarrow DTR\ [X^{m-1}0]$

:even PEs save DTR contents in A register

(L2) $PM2_{+0}\ [X^{m-1}1]$

:odd PEs send DTR data "+1" to even PEs

(L3) for $j = 1$ *until* $m-1$ *do*

begin

(L4) $A \longleftrightarrow DTR\ [X^{m-j-1}1X^{j-1}0]$

:even PEs, j-th bit=1, switch A and DTR

(L5) $PM2_{+j}\ [X^{m-1}0]$

:even PEs send DTR data "$+2^j$"

end

(L6) $PM2_{+0}\ [X^{m-1}0]$

:half of data sent from even PEs to odd PEs

(L7) $DTR \leftarrow A\ [X^{m-1}0]$

:reload DTR from A register in even PEs

This algorithm used m+1 inter-PE data transfers and m+1 register to register moves. The operation of this algorithm for N = 8 (m = 3) is shown in Table 10.2.

For example, consider the data item initially in the DTR of PE 5 (= 101). PE 5 does not match the mask in L1 ([XX0]). PE 5 does match the mask in L2 ([XX1]) and the data is moved to PE $PM2_{+0}(5) = 6$ (= 110). PE 6 does match the mask in L4 when j = 1 ([X10]) and the data is moved to the A register of PE 6. The data is unaffected by L5 when j = 1 (since it is not in the DTR). PE 6 does match the mask in L4 when j = 2 ([1X0]) and the data is moved to the DTR of PE 6. PE 6 does match the mask in L5 when j = 2 ([XX0]) and the data is moved to the DTR of PE $PM2_{+2}(6) = 2$. PE 2 does match the mask in L6 ([XX0]) and the data is moved to the DTR of PE $PM2_{+0}(2) = 3$. PE 3 does not match the mask in L7 ([XX0]). Thus, the data originally from PE 5 is moved to PE 3 = shuffle(5). This is shown by the dotted line in Table 10.2.

Proof that the algorithm is correct:

Assume all arithmetic is mod N.

The induction hypothesis (proven correct below) is that after executing $PM2_{+j}$ in L1 (for j = 0) or L5 (for $1 \leq j < m$) the data originally in the DTR of PE $Q = q_{m-1} \cdots q_1 q_0$ will currently be in PE $P = p_{m-1} \cdots p_1 p_0 = (q_{m-1} \cdots q_{j+2} q_{j+1}) * 2^{j+1} + (q_j \cdots q_1 q_0) * 2$. (When j = 0, $P = (q_{m-1} \cdots q_2 q_1) * 2 + (q_0) * 2$.) The data will be in the A register if $q_j = 0$ and in the DTR if $q_j = 1$.

Thus, when j = m−1, the data originally from PE Q is in PE $(q_{m-1} \cdots q_1 q_0) * 2$. The data item in the DTR of PE $(q_{m-1} \cdots q_1 q_0) * 2$ is moved to PE $(q_{m-1} \cdots q_1 q_0) * 2 + 1$ by L6; which is correct since this data item is from a

Table 10.2:
Example of the algorithm for performing the shuffle
using the PM2I when N = 8.
It is assumed that initially the DTR of PE P
contains the integer P, $0 \leq P < 8$.
The dotted line shows the movement of the data originally
in the DTR of PE 5 (= 101).

PE	Initial DTR Contents	L1 A Contents	L2 DTR Contents	L4, j=1 A Contents	L4, j=1 DTR Contents	L5, j=1 DTR Contents
000	000	000	111	000	111	110
001	001	-	-	-	-	-
010	010	010	001	001	010	111
011	011	-	-	-	-	-
100	100	100	011	100	011	010
101	101	-	-	-	-	-
110	110	110	101	101	110	011
111	111	-	-	-	-	-

PE	L4, j=2 A Contents	L4, j=2 DTR Contents	L5, j=2 DTR Contents	L6 DTR Contents	L7 DTR Contents
000	000	110	100	100	000
001	-	-	-	100	100
010	001	111	101	101	001
011	-	-	-	101	101
100	010	100	110	110	010
101	-	-	-	110	110
110	011	101	111	111	011
111	-	-	-	111	111

PE where $q_j = q_{m-1} = 1$, so shuffle(Q) = 2 * Q + 1. The data item in the A register of PE $(q_{m-1}\cdots q_1q_0) * 2$ is moved to the DTR of that PE by L7; which is correct since this data item is from a PE where $q_j = q_{m-1} = 0$, so shuffle(Q) = 2 * Q.

To complete the correctness proof it must be shown that the induction hypothesis is true. Basis: j = 0.

Case 1: Consider the data item originally in the DTR of PE $Q = q_{m-1}\cdots q_2q_10$. This data item is moved to the A register of that PE by L1. Since $q_0 = 0$, $Q = (q_{m-1}\cdots q_2q_1) * 2 + (q_0) * 2 = P$. This data is not moved by L2. It remains in the A register and $q_0 = 0$. Thus, the induction hypothesis is true for j = 0 for this case.

Case 2: Consider the data item originally in the DTR of PE $Q = q_{m-1}\cdots q_2q_11$. This data item is not moved by L1. It is moved to the DTR of PE P = Q + 1 by $PM2_{+0}$ in L2. Since $q_0 = 1$, $Q + 1 = q_{m-1}\cdots q_2q_11 + 1 = (q_{m-1}\cdots q_2q_1) * 2 + 2 = (q_{m-1}\cdots q_2q_1) * 2 + (q_0) * 2 = P$. The data item is in the DTR and $q_0 = 1$. Thus, the induction hypothesis is true for j = 0 for this case.

Induction Step: Assume true for j = k − 1 and show true for j = k.

Case 1: Consider the data item originally in the DTR of PE $Q = q_{m-1}\cdots q_2q_1q_0$, where $q_{k-1} = 0$.

From the induction hypothesis when j = k−1, this data item is in the A register of PE $P = p_{m-1}\cdots p_1p_0 = (q_{m-1}\cdots q_{k+1}q_k) * 2^k + (q_{k-1}\cdots q_1q_0) * 2$.

Subcase 1a: $p_k = 1$. The A register data is moved to the DTR of PE P by L4 and then to the DTR of PE $P + 2^k$ by L5. Recall $P = p_{m-1}\cdots p_1 p_0 = (q_{m-1}\cdots q_{k+1}q_k) * 2^k + (q_{k-1}\cdots q_1 q_0) * 2$. Since $q_{k-1} = 0$, $(0q_{k-2}\cdots q_1 q_0) * 2 < 2^k$. Thus, if $p_k = 1$, it must be that $q_k = 1$. Since $q_k = 1$, $P + 2^k = (q_{m-1}\cdots q_{k+1}1) * 2^k + (q_{k-1}\cdots q_1 q_0) * 2 + 2^k = (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + 2^k + (q_{k-1}\cdots q_1 q_0) * 2 + 2^k = (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + (1q_{k-1}\cdots q_1 q_0) * 2 = (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + (q_k q_{k-1}\cdots q_1 q_0) * 2$.

Furthermore, the data is in the DTR and $q_k = 1$. Thus, the induction hypothesis is true for $j = k$ for this subcase.

Subcase 1b: $p_k = 0$. The A register data is kept in the A register of PE P and not moved by L4 or L5. As in Subcase 1a, since $q_{k-1} = 0$, $(0q_{k-2}\cdots q_1 q_0) * 2 < 2^k$. Thus, if $p_k = 0$, it must be that $q_k = 0$. Since $q_k = 0$, $P = (q_{m-1}\cdots q_{k+1}0) * 2^k + (q_{k-1}\cdots q_1 q_0) * 2 = (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + (q_k\cdots q_1 q_0) * 2$.

Furthermore, the data is in the A register and $q_k = 0$. Thus, the induction hypothesis is true for $j = k$ for this subcase.

Case 2: Consider the data item originally in the DTR of PE $Q = q_{m-1}\cdots q_1 q_0$, where $q_{k-1} = 1$.

From the induction hypothesis when $j = k-1$, this data item is in the DTR of PE $P = p_{m-1}\cdots p_1 p_0 = (q_{m-1}\cdots q_{k+1}q_k) * 2^k + (q_{k-1}\cdots q_1 q_0) * 2$.

Subcase 2a: $p_k = 1$. The DTR data is moved to the A register of PE P by L4 and is not moved by L5. Recall $p_{m-1}\cdots p_1p_0 = (q_{m-1}\cdots q_{k+1}q_k) * 2^k + (q_{k-1}\cdots q_1q_0) * 2$. Since $q_{k-1} = 1$, $(q_{k-1}\cdots q_1q_0) * 2 = 2^k + (q_{k-2}\cdots q_1q_0) * 2$. Thus, if $p_k = 1$, it must be that $q_k = 0$. Since $q_k = 0$, $P = (q_{m-1}\cdots q_{k+1}0) * 2^k + (q_{k-1}\cdots q_1q_0) * 2$

$= (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + (q_k\cdots q_1q_0) * 2$.

Furthermore, the data is in the A register and $q_k = 0$. Thus, the induction hypothesis is true for $j = k$ for this subcase.

Subcase 2b: $p_k = 0$. The DTR data is kept in the DTR of PE P (not moved by L4). It is then moved to the DTR of PE $P + 2^k$ by L5. Since $q_{k-1} = 1$, $(q_{k-1}\cdots q_1q_0) * 2 = 2^k + (q_{k-2}\cdots q_1q_0) * 2$. Thus, if $p_k = 0$, it must be that $q_k = 1$. Since $q_k = 1$, $P + 2^k = (q_{m-1}\cdots q_{k+1}) * 2^{k+1} + (q_kq_{k-1}\cdots q_1q_0) * 2$ as in Subcase 1a. Furthermore, the data is in the DTR and $q_k = 1$. Thus the induction hypothesis is true for $j = k$ for this subcase.

This completes the proof that the induction hypothesis is true.

No data of interest is destroyed by the inter-PE data transfers. The transfer in L2 overwrites no relevant data since such data is saved in the A registers in L1. The transfers in L5, for $1 \leq j < m$, move data among the even numbered PEs (i.e., all even numbered PEs transfer data simultaneously) so no data is overwritten. Finally, the transfer in L6 overwrites data in the DTRs of the odd numbered PEs, however, all data of interest are in even

numbered PEs at that point.

This completes the correctness proof. All the data items have been moved as the shuffle would have moved them.

In this section an algorithm for PM2I emulating shuffle of smaller size will be developed. This algorithm is applicable when the machine with the PM2I network is larger (in terms of number of PEs) then the shuffle to be emulated. To solve this problem it will be decomposed into several subproblems.

It was shown in [Sie80] that the PM2I network can be partitioned into independent subnetworks. There are some constraints on how this can be done. Suppose there is a PM2I network of size $N = 2^m$ and it is desired to partition the network into groups of size 2^r ($0 \leq r \leq m$). Recall that the PEs are addressed as $p_{m-1}p_{m-2}...p_0$. To form a group of size 2^r all PEs in the group must have the same m−r least significant bits. That means that for each group the value of address bit positions $p_{m-r-1}p_{m-r-2}...p_0$ is fixed and unique. Denote $p_{m-r-1}p_{m-r-2}...p_0$ by B.

This group (identified uniquely by its value of B) then constitutes a logical PM2I network of size 2^r, with the PEs logically numbered from 0 to 2^r-1 by the r high order bits of their physical address. Each logical function $PM2_{+j}$ will be executed by the physical function $PM2_{+j+(m-r)}$.

The previous algorithm for the PM2I network to emulate the shuffle will be mapped into the logical PM2I network of size 2^r. This can be implemented as follows.

(a) Let the logical PE addresses in a set of size 2^r be denoted as

$$\{Q\} = \{q_{r-1}q_{r-2}...q_0 \mid q_i = 0,1\}.$$

Let the physical PE addresses in a set of size 2^m be denoted as

$$\{P\} = \{p_{m-1}p_{m-2}...p_0 \mid p_i = 0,1\}.$$

Define a map from the logical PE address set {Q} into the group B of the physical PE address set {P} as follows:

$$\phi: \{Q\} \rightarrow \{P\}$$

$$\phi(q_{r-1}q_{r-2}...q_0) \rightarrow q_{r-1}q_{r-2}...q_0B\ .$$

(b) Map the logical function set $PM2_{+j}$ into the physical function set as follows:

$$\psi : \{PM2_{+j}\} \rightarrow \{PM2_{+k}\}$$

$$\psi(PM2_{+j}) \rightarrow PM2_{+j+(m-r)} \quad \text{where } 0 \leq j \leq r\ .$$

Algorithm for a PM2I of size 2^m to emulate a shuffle of size 2^r ($1 \leq r \leq m$):

(L1) $A \leftarrow DTR\ [X^{r-1}0B]$

:logical even PEs save DTR data in A register.

(L2) $PM2_{+(m-r)}\ [X^{r-1}1B]$

:logical odd PEs send DTR data logical "+1" to logical even PEs

(L3) *for* $j = 1$ *until* $r-1$ *do*

begin

(L4) $A \leftrightarrow DTR\ [X^{r-j-1}1X^{j-1}0B]$

:logical even PEs, logical j-th bit = 1, switch A and DTR

(L5) $PM2_{+j+(m-r)}\ [X^{r-1}0B]$

:logical even PEs send DTR data logical "+ 2^j"

end

(L6) $PM2_{+(m-r)}$ [$X^{r-1}0B$]

:logical even PEs send data to logical odd PEs

(L7) DTR ← A [$X^{r-1}0B$]

:logical even PEs reload DTR from A register

The proof of correctness of this algorithm is directly based upon the theory of partitionability [Sie80] and the algorithm for performing the shuffle with PM2I of the same size. Performance evaluation of this algorithm follows. The general lower bound result in [Sie77] is applicable with r replaced by m, yielding lower bound of r transfers. Thus, this algorithm with a performance of r + 1 transfers compares favorably with the lower bound. This algorithm is applicable in the following situations. Suppose there is an SIMD machine with a PM2I network, then it is possible to select a group of PEs (with certain constraints) and let the group perform a shuffle (while the other PEs are disabled). Alternatively it is possible to "partition" the network into equal size groups and let any or all of the groups perform the shuffle concurrently, using appropriate masking. The groups which will perform the shuffle will be determined by the value of "B" in the algorithm. Suppose there is an multiple-SIMD machine with a PM2I network, then the algorithm can be used so that each SIMD submachine can emulate shuffle independently. Since the submachines are independent, they can be of different sizes.

10.6 Shuffling with the Illiac Network

In this section the use of the Illiac network to perform the shuffle will be examined. A lower bound of 2(n−2) transfers can be derived from [NaS80]. In [NaS80] there is also a procedure for constructing an algorithm for a mesh-connected computer to perform the shuffle in 2n transfers (a mesh network is the same as an Illiac network without the "wrap around" edge links). In this section, an explicit algorithm for the Illiac to perform the shuffle in 2n−1 transfers is given. It is based upon the algorithm for the PM2I to perform the shuffle. Since Illiac cannot be partitioned into independent subnetworks [Sie80], consideration of performing the shuffle on a subset of PEs is inappropriate.

In this section an algorithm to perform the shuffle with the Illiac will be developed. Consider an algorithm for performing a size N shuffle interconnection function on a size N Illiac network, where $r = \sqrt{N} = 2^{m/2}$ is an integer (i.e. m is even). The three ground rules listed in Section 10.5 are also used in this section.

The algorithm to perform the shuffle using the Illiac network will be constructed by replacing each PM2I interconnection function in the above algorithm with Illiac interconnection functions. For L2, use "Illiac_{+1} $[X^{m-1}1]$," since $\text{Illiac}_{+1} \equiv \text{PM2}_{+0}$ by definition. Similarly, for L6, use "Illiac_{+1} $[X^{m-1}0]$." To do L5, first recall that only the even numbered PEs contain the data of concern (after L2 is executed and before L6 is executed). Therefore, it is acceptable to use "PM2_{+j} $[X^m]$" in L5, since any data movement among the odd numbered PEs is ignored (and overwritten by L6). To perform

"$PM2_{+j}[X^m]$," for $1 \leq j < m$, with the Illiac network the algorithms presented in [Sie79b] can be used. Specifically, to perform "$PM2_{+j}[X^m]$" for $1 \leq j < m/2$ use:

$$for\ i = 1\ until\ 2^j\ do\ Illiac_{+1}[X^m]$$

since 2^j execution of $Illiac_{+1}$ is equivalent to $+2^j = PM2_{+j}$. Analogously, to perform "$PM2_{+j}[X^m]$" for $m/2 \leq j < m$ use:

$$for\ i = 1\ until\ 2^j/n\ do\ Illiac_{+n}[X^m]$$

since $2^j/n$ executions of $Illiac_{+n}$ is equivalent to $+2^j = PM2_{+j}$. The total number of Illiac transfers needed is:

for L2: 1

for L6: 1

for L5, $1 \leq j < m/2$: $\sum_{j=1}^{(m/2)-1} 2^j = 2^{m/2} - 2 = n-2$

for L5, $m/2 \leq j < m$: $\sum_{j=m/2}^{m-1} 2^j/n = \sum_{j=m/2}^{m-1} 2^{j-(m/2)} = \sum_{j=0}^{(m/2)-1} 2^j = n-1$

Thus, the grand total is $2n-1$ transfers. The number of register to register moves is still $m+1$.

In summary, an algorithm to perform the shuffle data permutation using the Illiac interconnection network has been constructed based on the algorithm to perform the shuffle using the PM2I network. The algorithm developed for the Illiac required $2n-1$ inter-PE transfers.

10.7 Conclusions

The ability of the PM2I and Illiac type single stage SIMD machine interconnection networks to perform the shuffle interconnection was examined. It was previously shown that a lower bound on the number of transfers needed for the PM2I network to perform the shuffle is $\log_2 N$. The algorithm described here and proven correct required only $(\log_2 N)+1$ transfers. Also, an algorithm for the case where there is a machine with a PM2I network and it is desired to emulate a shuffle that is of smaller size than the host network was presented. Using the PM2I algorithm as a basis, an algorithm for the Illiac to emulate the shuffle is given. Its performance is $2\sqrt{N}-1$ transfers, which is only three transfers more than the previously shown lower bound of $2\sqrt{N}-4$.

These results are of both theoretical and practical value. Theoretically, they add to the body of knowledge about the properties of the PM2I and Illiac networks. Practically, the algorithms presented could actually be used to perform the shuffle interconnection on a system that has implemented the PM2I or Illiac type of interconnection network.

11 CONCLUSIONS

The two major methods used to speed up the execution phase of a computational task are (a) utilization of new materials to construct faster devices and (b) exploitation of parallel execution of subtasks of the task. This research was concerned with the second method of speeding up the execution phase of a task. The exploitation of time parallel execution of subtasks of the task requires parallel computer architectures. In general a parallel computer system consists of a set of devices such as processors, memories, and I/O devices that communicate through one or more interconnection networks.

Different computer systems use their networks differently, as can be seen in the following few examples. Some systems use networks dedicated to the communication between particular subsystems, some other systems use a single network multiplexed for communication among different parts of the system. In an ensemble parallel system the network is used by the control unit to broadcast instructions and data. In a pipelined system the interconnection network is used to provide data communication among the computational units (segments) of the pipeline. In vector and array parallel system one interconnection network is used for interprocessor communication and a usually separate network is used by the control unit to broadcast data, instructions, and control information to the processors. In a systolic system the interconnection network is used to propagate the wave of the partial results from a set of processors to the next set of processors. In an associative system the control unit uses the interconnection network to broadcast the selected data fields to the processors for comparison, and in some cases another network is used for interprocessor communications. Reconfigurable systems have a network that allows the system to be statically or dynamically restructured into multiple machines of different sizes in terms of processors. Data flow

system consisting of multiple rings needs a communication network to move data among rings.

The computer system designer is faced with two basic tasks: the analysis and evaluation of existing interconnection networks and the synthesis of desired interconnection networks. Much research has been done on several topologically regular interconnection networks. Amongst the best known networks are Illiac [BoD72], Shuffle [LaS76], Omega [Law75], multistage Cube [AdS82b], STARAN [Bat76], ADM [McS82], k-connected mesh [NaS80], and PM2I [SeS84b]. The researcher usually proceeded as follows: he selected a network of interest, devised a model for that network and derived analytical results based on that network. This approach has the drawback that the results are network specific since the model is network specific and sometimes implementation dependent. In addition, most work was concentrating on the analysis of properties of networks and only a little has been done on the synthesis of networks with desired properties.

Our research differs from the past work in following aspects. First, a unified approach to the analysis of interconnection networks that is valid for large of classes interconnection networks was developed. The approach is unified in the sense that it does not assume a particular network or an implementation but considers networks as a set. Second, two algorithms that allow systematic design of networks with the desired property of partitionability were developed. In more detail, the following related topics of topological properties of parallel computer systems were studied.

In Chapter 4 a general, implementation independent model for single stage interconnection networks was developed. The model can be used to analyze both topologically regular and irregular single stage interconnection networks.

The network model was extended to the modeling of parallel computer systems. A system informally consists of a set of devices, an interconnection network, and the method of use of the network by the devices. Three different types of systems were defined, based upon the method of use of the network, and several relationships between systems were analyzed. The systems types recognized are recirculating, nonrecirculating, and partially recirculating. In a recirculating system each device d_i has its logical output port connected to an input label of the interconnection network and its input port connected to an output label of the interconnection network. One result of this configuration is that the system can generate different connection patterns using multiple passes through the network. Also, for a recirculating system $|V_I| = |V_O|$. In a nonrecirculating system, each device is connected only to network input or (exclusively) to a network output. This type of configuration appears frequently in real time digital signal processing systems. The result of this configuration is that no new connection patterns can be achieved by multiple passes, since it is not possible to move the data from the output of the network back to its input. A partially recirculating system contains some, but not all, devices each of which has its output port connected to the network input label and its input port connected to an output label of the network. If $|V_I| \neq |V_O|$ than the system cannot be recirculating and can be only either partially recirculating or nonrecirculating.

The previous method of classification of the relationship between two networks K^1 and K^2 used only two categories: (a) K^1 and K^2 are isomorphic and (b) K^1 and K^2 are not isomorphic. Our method refined the classification of the relationship between two networks into the following categories presented in the order of decreasing similarity: (a) K^1 is isomorphic to K^2, (b) K^1 is

subnetwork of type c of K^2, (c) K^1 is subnetwork of type b of K^2, (d) none of the above. The method was expanded to classify relationship between two systems S^1 and S^2. The categories are similar except in addition to above, another category is possible after "type b" and that is "type a."

In Chapter 5 the measure of similarity between two systems was expanded to include arbitrary labeling. Recall that in Chapter 4 the comparison between system S^1 over $V_I^1 \times V_O^1$ and system S^2 over $V_I^2 \times V_O^2$ assumed the labeling was such that $V_I^1 \times V_O^1 \subseteq V_I^2 \times V_O^2$. If this condition does not hold that does not mean that the two systems are necessarily dissimilar. It could be that the labeling of of the two systems is different. To handle this problem, the concept of quasimorphism of systems was developed. Quasimorphism allows comparison of randomly labeled systems with arbitrary topologies. The problem of comparison of systems can be formulated as finding relationships between two S-sets. The problem is very complex and therefore was broken down into two major steps. First the T-set over $V_I \times V_O$ was defined. The T-set has less constraints than the S-set over the same $V_I \times V_O$ and therefore it is easier to analyze relationships between T-sets than between S-sets. The quasimorphism, denoted by $\bar{\psi}$, is uniquely determined by two maps ϕ_I and ϕ_O. Some behavior of ϕ_I, ϕ_O was inherited by $\bar{\psi}$-correspondence $\bar{\psi}$. For example if ϕ_I-map ϕ_I, and ϕ_O-map ϕ_O are 1:1 maps then $\bar{\psi}$-correspondence $\bar{\psi}$ is a 1:1 correspondence. Conversely if a $\bar{\psi}$-correspondence $\bar{\psi}$ is 1:1 then the ϕ_I-map ϕ_I and the ϕ_O-map ϕ_O are 1:1 maps. Properties of the $\bar{\psi}$-correspondence $\bar{\psi}$ similar to the reflexive, symmetric, and transitive properties of relations were discussed, in particular the following were shown. Let S^1, S^2, and S^3 be three systems. A quasimorphism has the following properties.

(1) $\exists\ \bar{\psi}$ such that $\bar{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2 \not\rightarrow \exists\ \bar{\psi}^2$ such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$ and $\bar{\psi}^2(S^2) = S^3 \rightarrow \exists\ \bar{\psi}$, $\bar{\psi}(S^1) = S^3$.

Let S^1, S^2, and S^3 be three systems. A 1:1 quasimorphism has the following properties.

(1) $\exists\ \bar{\psi}$, 1:1 such that $\bar{\psi}(S^1) = S^1$.

(2) $\bar{\psi}^1(S^1) = S^2$, 1:1 $\rightarrow \exists\ \bar{\psi}^2$, 1:1 such that $\bar{\psi}^2(S^2) = S^1$.

(3) $\bar{\psi}^1(S^1) = S^2$, 1:1 and $\bar{\psi}^2(S^2) = S^3$, 1:1 $\rightarrow \exists\ \bar{\psi}$, 1:1, $\bar{\psi}(S^1) = S^3$.

The quasimorphism measure provides a theoretical background for studying the following problems of parallel processing.

(a) Emulation of system S^1 by system S^2.

(b) Fault tolerance method achieved by concurrent execution of multiple copies of the same problem.

(c) Partitioning of a system.

Three types of emulation were defined based upon the subsystem relationship between the image of the emulated system and the host system. Several measures of efficiency of the emulation based upon the preservation of the computational loading and other factors were defined and the emulation types were evaluated on that basis. For example in a system where the input nodes are connected to processors and the output nodes to memories, the factors have the following physical meaning. If the input node factor = 1, then the computational load of each processor in the emulated system is same as the computational load in the image of the emulated system in the host system. If the output node factor = 1, then the amount of data stored in each of

memories of the emulated system is same as the amount of the data stored in the image of the emulated system in the host system. Factors greater than 1 imply a heavier load in terms of computation or amount of data stored per memory unit than in the host system.

In Chapter 6 operations on single stage networks such as composition and decomposition were defined. Using these primitives, the partitionability of single stage networks was studied. Partitionability informally means that the network can be divided into several parts with certain amount of independence. The partitionability property is important for the reasons detailed in the chapter.

Three types of partitionability were recognized and an algorithm was developed which will output one of the following:

(1) The network is not partitionable.

(2) The network is partitionable into subnetworks with common control signals and the combination of the of the subnetworks will exactly generate all interconnection patterns of the original network.

(3) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will exactly generate all interconnection patterns of the original network.

(4) The network is partitionable into subnetworks with separate control signals and the combination of the subnetworks will generate a superset of interconnection patterns of the original network.

The algorithm is network topology independent and can be used to analyze topologically regular and irregular interconnection networks.

In Chapter 7 two techniques of synthesis of single stage partitionable networks were developed. Each of these techniques allow the design of a large class of partitionable single stage interconnection networks. The specification of the construction is given in terms of properties of the individual I/O correspondences.

In Chapter 8 multistage networks were studied. A composition of single stage networks was defined and its properties studied. Using the model of single stage network and composition above, the multistage network was defined. The model is very general since each stage of the multistage network is topologically general single stage network. Several examples of the application of the model were presented.

In Chapter 9 a network and network interfaces were designed for a real-time, distributed digital signal processing system. The design was subject to number of system constraints such as very high throughput, system extendibility, and fault tolerance requirements. For this application, and given the current and near future technology, a crossbar based interconnection network was very well-suited to the task under consideration. Two different fault tolerant chip architectures were presented. Four network architectures were designed and their characteristics described. Several fault detection and recovery techniques on the system level were shown, since the fault tolerance is a salient issue of this system.

In Chapter 10 the ability of the PM2I and Illiac type single stage SIMD machine interconnection networks to perform the shuffle interconnection was examined. It was previously shown that a lower bound on the number of transfers needed for the PM2I network to perform the shuffle is $\log_2 N$. The algorithm described here and proven correct required only $(\log_2 N)+1$ transfers.

Also, an algorithm for the case where there is a machine with a PM2I network and it is desired to emulate a shuffle that is of smaller size than the host network was presented. Using the PM2I algorithm as a basis, an algorithm for the Illiac to emulate the shuffle was developed. Its performance is $2\sqrt{N} - 1$ transfers, which is only three transfers more than the previously shown lower bound of $2\sqrt{N} - 4$.

LIST OF REFERENCES

LIST OF REFERENCES

[AdS82a] G. B. Adams III and H. J. Siegel, "On the number of permutations performable by the augmented data manipulator network," *IEEE Trans. Comput.*, Vol. C-31, Apr. 1982, pp. 270-277.

[AdS82b] G. B. Adams III and H. J. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, Vol. C-31, May 1982, pp. 443-454.

[BaB68] G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick, and R. A. Stokes, "The Illiac IV computer," *IEEE Trans. Comput.*, Vol. C-17, Aug. 1968, pp. 746-757.

[Bae80] J. L. Baer, *Computer Systems Architecture*, Computer Science Press, Potomac, MD, 1980.

[Bat74] K. E. Batcher, "STARAN Parallel Processor System Hardware," *AFIPS Conf. Proc., 1974 Nat'l. Computer Conf.*, May 1974, pp. 405-410.

[Bat76] K. E. Batcher, "The flip network in STARAN," *1976 Int'l. Conf. Parallel Processing, Aug. 1976*, pp. 65-71.

[Bat77] K. E. Batcher, "STARAN series E," *1977 Int'l. Conf. Parallel Processing*, Aug. 1977, pp. 140-143.

[Bat80] K. E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. Comput.*, Vol. C-29, Sept. 1980, pp. 836-840.

[BeK79] J. L. Bentley and H. T. Kung, "A Tree Machine for Searching Problems," *1979 Int'l. Conf. Parallel Processing*, Aug. 1979, pp. 257-266.

[Ben65] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, NY, 1965.

[Ben74] V. E. Benes, "Applications of group theory to connecting networks," *The Bell System Technical Journal*, Vol. 54, Feb. 1975, pp. 407-420.

[BoD72] W. J. Bouknight, S. A. Denenberg, D. E. McIntyre, J. M. Randall, A. H. Sameh, and D. L. Slotnick, "The Illiac IV System," *Proc. IEEE*, Vol. 60, Apr. 1972, pp. 369-388.

[BoM76] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North Holland Publishing, 1976.

[BuK71] P. Budnick and D. J. Kuck, "The organization and use of parallel memories," *IEEE Trans. Comput.*, Vol. C-20, Dec. 1971, pp. 1566-69.

[ChL81] P-Y. Chen, D. H. Lawrie, P-C. Yew, and D. A. Padua, "Interconnection networks using shuffles," *Computer*, Vol. 14, Dec. 1981, pp. 55-64.

[CoG74] G. R. Couranz, M. S. Gerhardt, and C. J. Young, "Programmable RADAR signal processing using the RAP," *1974 Sagamore Computer Conf. Parallel Processing*, Aug. 1974, pp. 37-52.

[Dem83] G. L. DeMuth, "A distributed signal processor incorporating VLSI and high order language programming," *1983 Int'l. Conf. Acoustics, Speech, and Signal Processing*, Apr. 1983, pp. 439-442.

[FeF74] J. D. Feldman and L. C. Fulmer, "RADCAP - An operational parallel processing facility," *1974 Nat'l. Computer Conf.*, May 1974, pp. 7-15.

[Fen74] T. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comput.*, Vol. C-23, Mar. 1974, pp. 309-318.

[Fen81] T. Feng, "A survey of interconnection networks," *Computer*, Vol. 14, Dec. 1981, pp. 12-27.

[FiF82] J. P. Fishburn and R. A. Finkel, "Quotient networks," *IEEE Trans. Comput.*, Vol. C-31, Apr. 1982, pp. 288-295.

[Fly66] M. J. Flynn, "Very high-speed computing systems," *Proc. of the IEEE*, Vol. 54, Dec. 1966, pp. 1901-1909.

[FrW81] M. A. Franklin and D. F. Wann, "Pin limitations and VLSI interconnection networks," *1981 Int'l. Conf. Parallel Processing*, Aug. 1981, pp. 253-258.

[FrW82] M. A. Franklin, D. F. Wann, and W. J. Thomas, "Pin limitations and partitioning of VLSI interconnection networks," *IEEE Trans. Computers*, Vol. C-31, Nov. 1982, pp. 1109-1116.

[Gok76] L. R. Goke, *"Banyan networks for partitioning multiprocessors systems,"* Ph.D. Thesis, University of Florida, June 1976.

[GoL73] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessing systems," *1st Annual Symp. Computer Architecture*, Dec. 1973, pp. 21-28.

[HaL82] L. S. Haynes, R. L. Lau, Siewiorek, and D. W. Mizell, "A survey of highly parallel computing," *Computer*, Vol. 15, Jan. 1982, pp. 9-24.

[Han68] C. B. Hanneken, *Introduction to Abstract Algebra*, Dickenson Publishing, 1968.

[Har69] F. Harary, *Graph Theory*, Addison Wesley Publisher, 1969.

[Her75] I. N. Herstein, *Topics in Algebra*, Xerox College Publishing, 1975.

[Hig72] L. C. Higbie, "The Omen computer: associative array processor," *IEEE Computer Society Compcon 72*, Sept. 1972, pp. 287-290.

[Hun81] D. J. Hunt, "The ICL DAP and its application to image processing," in *Languages and Architectures for Image Processing*, M. J. B. Duff and S. Levialdi, eds., Academic Press, London, England, 1981, pp. 275-282.

[KaP80] R. N. Kapur, U. V. Premkumar, and G. J. Lipovski, "Organization of the TRAC processor-memory subsystem," *1980 Nat'l. Computer Conf.*, May 1980, pp. 623-629.

[KoT80] E. W. Kozdrowicki and D. J. Theis, "Second generation of vector supercomputers," *Computer*, Vol. 13, Nov. 1980, pp. 71-83.

[KuL78] H. T. Kung and C. E. Leiserson, "Systolic arrays (for VLSI)," *Proc. Symp. Sparse Matrix Computations and Their Applications*, Nov. 1978, pp. 256-282.

[Kun82] H. T. Kung, "Why Systolic Architectures?," *Computer*, Vol. 15, Jan. 1982, pp. 37-46.

[Lan76] T. Lang, "Interconnections between processors and memory modules using the shuffle-exchange network," *IEEE Trans. Comput.*, Vol. C-25, May 1976, pp. 496-503.

[LaS76] T. Lang and H. S. Stone, "A shuffle-exchange network with simplified control," *IEEE Trans. Comput.*, Vol. C-25, Jan. 1976, pp. 55-66.

[Law75] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, Vol. C-24, Dec. 1975, pp. 1145-1155.

[LiM82] G. J. Lipovski and M. Malek, *"A Theory for Interconnection Networks,"* Electrical Engineering Department, University of Texas at Austin, TRAC Report 41, Oct. 1982.

[MaM81a] B. A. Makrucki and T. N. Mudge, "VLSI design of a crossbar switch," *Technical Report SEL-TR-149*, Dept. of Electrical and Computer Engineering, The University of Michigan, Ann Arbor, MI, Jan. 1981.

[MaM81b] M. Malek and W. W. Myre, "A description method of interconnection network," *Distributed Processing,* Vol. 1, No. 1, Feb. 1981, pp. 1-6.

[McS82] R. J. McMillen and H. J. Siegel, "Routing schemes for the augmented data manipulator network in an MIMD system," *IEEE Trans. Comput.,* Vol. C-31, Dec. 1982, pp. 1202-1214.

[McT82] S. McFarling, J. Turney, and T. N. Mudge, "VLSI design version two," *Technical Report CRL-TR-8-82,* Computing Research Laboratory, The University of Michigan, Ann Arbor, MI, Feb. 1982.

[NaS80] D. Nassimi and S. Sahni, "An optimal routing algorithm for mesh-connected parallel computers," *J. Ass. Comput. Mach.,* Vol. 27, Jan. 1980, pp. 6-29.

[NaS81] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Trans. Comput.,* Vol. C-30, Feb. 1981, pp. 101-107.

[NaS82] D. Nassimi and S. Sahni, "Parallel permutation and sorting algorithms and a new generalized connection network," *Journal of the ACM,* Vol. 29, July 1982, pp. 642-667.

[Nut77] G. J. Nutt, "Microprocessor implementation of a parallel processor," *4th Symp. Computer Architecture,* Mar. 1977, pp. 147-152.

[OkT82] Y. Okada, H. Tajima, and R. Mori, "A reconfigurable parallel processor with microprogram control," *IEEE Micro,* Vol. 2, Nov. 1982, pp. 48-60.

[Orc76] S. E. Orcutt, "Implementation of permutation functions in Illiac IV-type computers," *IEEE Trans. Comput.,* Vol. C-25, Sept. 1976, pp. 929-936.

[PaR82] D. S. Parker and C. S. Raghavendra, "The gamma network: a multiprocessor interconnection network with redundant paths," *9th Annual Symp. Computer Architecture,* Apr. 1982, pp. 73-80.

[Pea77] M. C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, Vol. C-26, May 1977, pp. 458-473.

[PrK80] D. K. Pradhan and K. L. Kodandapani, "A uniform representation of single- and multistage interconnection networks used in SIMD machines," *IEEE Trans. Comput.*, Vol. C-29, Sept. 1980, pp. 777-791.

[RaF83] L. V. Ramakrishnan, D. S. Fussell, and A. Silberschatz, "On mapping homogeneous graphs on a linear array-processor model," *1983 Int'l. Conf. Parallel Processing*, Aug. 1983, pp. 440-447.

[ReS82] A. P. Reeves and R. R. Seban, "The moment computer," *15th Hawaii Int'l. Conf. System Sciences*, Jan. 1982, pp. 388-396.

[RoP77] D. Rohrbacher and J. L. Potter, "Image processing with the STARAN parallel computer," *Computer*, Vol. 10, June 1977, pp. 54-59.

[Seb82] R. R. Seban, *"Parallel Computer Architecture Parallel Algorithms and the Theory of Image Analysis Using Cartesian Moments,"* Master's Thesis, Purdue University, Aug. 1982.

[SeS84a] R. R. Seban and H. J. Siegel, "Theoretical modeling and analysis of special purpose interconnection networks," *4th Int'l. Conf. Distributed Computing Systems*, May 1984, pp. 256-265.

[SeS84b] R. R. Seban and H. J. Siegel, "Shuffling with Illiac and PM2I SIMD networks," *IEEE Trans. Comput.*, Vol. C-33, July 1984, pp. 619-625.

[SeS84c] R. R. Seban, H. J. Siegel, and D. G. Meyer, "Data communications in a Real-Time distributed signal processing system: A case study," *1984 Real-Time Systems Symp.*, Dec. 1984.

[SeS85] R. R. Seban and H. J. Siegel, "Analysis of partitionability properties of topologically arbitrary interconnection networks," *5th Int'l. Conf. Distributed Computing Systems*, May 1985, pp. 173-181.

[SeU80] M. C. Sejnowski, E. T. Upchurch, R. N. Kapur, D. P. S. Charlu, and G. J. Lipovski, "An overview of the Texas Reconfigurable Array Computer," *1980 Nat'l. Computer Conf.*, May 1980, pp. 631-641.

[Sie77] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks," *IEEE Trans. Comput.*, Vol. C-26, Feb. 1977, pp. 153-161.

[Sie79a] H. J. Siegel, "Interconnection networks for SIMD machines," *Computer*, Vol. 12, June 1979, pp. 57-65.

[Sie79b] H. J. Siegel, "A model of SIMD machines and a comparison of various interconnection networks," *IEEE Trans. Comput.*, Vol. C-28, Dec. 1979, pp. 907-917.

[Sie80] H. J. Siegel, "The theory underlying the partitioning of permutation networks," *IEEE Trans. Comput.*, Vol. C-29, Sept. 1980, pp. 791-801.

[Sie85] H. J. Siegel, *Interconnection Networks for Large Scale Parallel Processing: Theory and Case Studies*, Lexington Books, Lexington, MA, 1985.

[SiM81] H. J. Siegel and R. J. McMillen, "The multistage cube: A versatile interconnection network," *Computer*, Vol. 14, Dec. 1981, pp. 65-76.

[SiM84] H. J. Siegel, D. G. Meyer, E. J. Coyle, R. R. Seban, S. Hutchinson and B. Young, *"Interconnection networks for a distributed signal processing system,"* Report for IBM Federal Systems Division, Manassas, VA, Dec. 1983.

[SiS79] H. J. Siegel, L. J. Siegel, R. J. McMillen, P. T. Mueller, Jr., S. D. Smith, "An SIMD/MIMD multimicroprocessor system for image processing and pattern recognition," *1979 IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing*, Aug. 1979, pp. 214-224.

[SiS81] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, and S. D. Smith, "PASM: a partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Comput.*, Vol. C-30, Dec. 1981, pp. 934-947.

[SiS84] H. J. Siegel, T. Schwederski, N. J. Davis IV, and J. T Kuehn, "PASM: A reconfigurable parallel system for image processing," *Computer Architecture News*, Vol. 12, No. 4, Sep. 1984, pp. 7-19.

[Sny82] L. Snyder, "Introduction to the Configurable Highly Parallel Computer," *Computer*, Vol. 15, Jan. 1982, pp. 47-56.

[Sto71] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, Vol. C-20, Feb. 1971, pp. 153-161.

[Sto80] H. S. Stone, *Introduction to Computer Architecture*, Science Research Associates, Chicago, IL, 1980.

[ThC83] J. E. Thornton and G. S. Christensen, "Hyperchannel network links," *Computer*, Vol. 16, Sept. 1983, pp. 50-54.

[The74] D. J. Theis, "Vector supercomputers," *Computer*, Vol. 7, April 1974, pp. 52-61.

[ThN81] S. Thanawastien and V. P. Nelson, "Interference analysis of shuffle/exchange networks," *IEEE Trans. Computers*, Vol. C-30, Aug. 1981, pp. 545-556.

[ThW75] K. J. Thurber and L. D. Wald, "Associative and Parallel Processors," *Computing Surveys*, Vol. 7, Dec. 1975, pp. 215-255.

[TuS83] D. L. Tuomenoksa and H. J. Siegel, "Analysis of multiple-queue task scheduling algorithms for multiple-SIMD machines," *3rd Int'l. Conf. Distributed Computing Systems*, Oct. 1982, pp. 114-121.

[Upp81] P. V. Uppaluru, *"A theoretical basis for analysis and partitioning of regular SW banyans,"* Ph.D. Thesis, The University of Texas at Austin, 1981.

[ViC78] C. R. Vick and J. A. Cornell, "PEPE architecture - Present and future," *1978 Nat'l. Computer Conf.*, June 1978, pp. 981-992.

[WaF83] D. F. Wann and M. A. Franklin, "Asynchronous and clocked control structures for VLSI based interconnection networks," *IEEE Trans. Computers*, Vol. C-32, Mar. 1983, pp. 284-293.

[WaG82] I. Watson and J. Gurd, "A practical data flow computer," *Computer,* Vol. 15, Feb. 1982, pp. 51-57.

[WuF80] C. Wu and T. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.,* Vol. C-29, Aug 1980, pp. 694-702.

[WuF81] C. Wu and T. Feng, "The universality of the shuffle-exchange network," *IEEE Trans. Comput.,* Vol. C-30, May 1981, pp. 324-332.

[WuF84] C. Wu and T. Feng, *Tutorial: Interconnection Networks for Parallel and Distributed Processing,* IEEE, Computer Society Press, Silver Spring, MD, 1984.

[YaF77] S. S. Yau and H. S. Fung, "Associative processor architecture - A survey," *Computing Surveys,* Vol. 9, March 1977, pp. 3-27.

VITA

VITA

Robert R. Seban received the B.E. degree in electrical engineering with Magna Cum Laude from the City College of New York in 1976. From 1976 to 1980 he worked in industry for Sperry Univac, IBM, and Advanced Technology Systems. In industry he held the positions of logic and system designer and project supervisor. In 1980 he returned to graduate school at Purdue University. He received M.S. in 1982 and Ph.D. in 1985, both in electrical engineering from Purdue University. While at Purdue he was a recipient of David Ross fellowship.

Dr. Seban has written one journal publication, five conference papers, and several technical reports. His current research interests include computer architectures for parallel processing and interconnection networks.

He is a member of Eta Kappa Nu Honorary Society, Sigma Xi Scientific Research Society, and IEEE Computer Society. His hobbies include swimming, flying, and racquetball.

END

DTIC

8-86