1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**Hyperstructures as Computer
Architectures for High Performance
Multiprocessor Systems,**

by

S. Boucouris

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of**

Doctor of Philosophy

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

September 25, 1989

copyright

1989. Spiros Boucouris

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-58975-2

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

**Hyperstructures as Computer
Architectures for High Performance
Multiprocessor Systems**

submitted by

Spiros Boucouris, B.Eng., M.Eng.

in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Chair, Department of Systems and Computer Engineering

Thesis Supervisor

External Examiner

Carleton University

March 30, 1990

Abstract

In this work we address the problem of analytically evaluating multiprocessor configurations, targeted towards hypercube-type architectures. Previous work on hypercube derivatives has produced a variety of architectures but no generalized framework for their analysis and comparison. In this work a theoretical framework (along with a set of closed form expressions) for the evaluation of hypercube-like architectures (Hyperstructures) is presented. The evaluation criteria include the diameter, average message path length, average message density and fault tolerance. It is demonstrated that almost all configurations proposed in the literature are analyzable as special cases of the proposed scheme. A number of containment and partitioning properties of the binary hypercube are generalized to a large class of hypercube derivatives. Algorithms are presented that make possible the generation of near-optimal layouts for any hypercube derivative. Results are presented that match or exceed the performance of configuration specific algorithms previously reported. The reduction of the interconnection cost of a Hyperstructure is also addressed. An approach is presented that makes possible the reduction of the interconnection cost in a controlled fashion.

Acknowledgements

I would like to gratefully acknowledge all the people whose assistance and support made this thesis possible.

Special thanks are due to my thesis advisor, Dr. B.A. Bowen, for his support, guidance, patience and especially his friendship throughout this work. Thanks are also due to Dr. S.A. Mahmoud for his advise and assistance during the early stages of this work. I would like to thank Dr. N. Santoro for many fruitful discussions that proved instrumental to the direction of this work. I would also like to thank the member of the examining committee for their many insights and suggestions. Finally I would like to thank my colleagues at BNR and especially Dr. D.G. Agnew for his support and Dr. S.J. Curry for many useful discussions and suggestions during the later stages of this work.

Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
1 Introduction	1
1.1 Subject of the Thesis	2
1.2 Organization of the Thesis	4
2 Literature Review	5
2.1 The Binary Hypercube	6
2.2 Other Hypercube Structures	8
2.3 Modified Hypercube Structures	10
2.4 Other structures with W^D Processors	18
2.5 Discussion	20

3	Hyperstructures	22
3.1	Definition	23
3.1.1	Multiprocessor Structures	23
3.1.2	A Mixed Radix System	33
3.1.3	Cartesian Product of Structures	34
3.1.4	Multiprocessor Hyperstructure	37
3.2	Properties of Hyperstructures	40
3.2.1	Diameter	41
3.2.2	Interprocessor Communication Requirements	42
3.2.3	Average Message Path Length	43
3.2.4	Network Message Load	45
3.2.5	Processor Message Load	47
3.2.6	Fault Tolerance	48
3.2.7	Examples	51
3.3	Hyperstructures with Point to Point Connections	55
3.3.1	Binary Hypercube	59
3.3.2	Generalized Hypercubes	60
3.3.3	Embedded Cycles	60

4	Hyperstructure Layout	65
4.1	Linear Structure Layout	66
4.2	Layout of Structure Product	71
4.2.1	Co-Linear Layout	71
4.2.2	Orthogonal Layout	78
4.3	Estimation of Hyperstructure Layout Area	81
4.4	Minimization of Hyperstructure Layout Area	86
5	Reduction of Hyperstructure Interconnection	91
5.1	Overview of Present Techniques	92
5.2	Interconnection Adjacency Matrix (IAM)	93
5.3	Interconnection Pruning	99
5.4	Layout of Reduced Hyperstructures	102
5.4.1	Comparison to Previous Results	107
6	Summary and Conclusions	113
6.1	Recommendations for Further Work	115
	References	116
A	List of Symbols	121

Chapter 1

Introduction

Multiprocessing has always been the obvious, perhaps even the only, available approach to solving the ever increasing computing requirements. Throughout the history of computers, a variety of special and semi-special purpose multiprocessor systems have been proposed and built for a wide range of applications.

Almost invariably today's mainframes and supercomputers employ more than one CPU. Shared memory systems represent the simplest approach to multiprocessing with two or more processors accessing some common storage and sharing the processing load among themselves. Typically, for straightforward designs, memory bandwidth severely limits the number of processors to less than 20. However, some approaches have been proposed and employed that make it possible to build shared memory systems with a few hundred processors. It is expected that the average number of processors will increase over the next decade as the techniques for implementing and managing coarse grain parallelism come to maturity and are put into commercial machines.

Several structures have also been proposed in the literature for interconnect-

ing a large network of computers in parallel and distributed environments. Although there are many interconnection structures that can be used for constructing large multiprocessor systems, advantages and disadvantages of different structures are always the subject of debate. The difficulty arises not only from the lack of tools to make it easy to model different structures, but also a lack of knowledge about how structures affect system performance.

An interconnection structure in general should have a low interconnection cost, small communication latencies and a large number of alternate paths between any pair of nodes for fault tolerance. Most prominent among the interconnection schemes employed is the Binary Hypercube, with several machines presently on the market. Considering its excellent interconnection characteristics, and the availability of well understood algorithms for mapping several important classes of algorithms to a Binary Hypercube, its popularity is to be expected. Several derivatives of the Binary Hypercube have also been proposed in the literature. Some have also been employed in actual machines and used (sometimes quite ingeniously) in a variety of applications with varying degrees of success.

1.1 Subject of the Thesis

A number of hypercube structures have been defined and discussed in the literature. Hypercube derivatives have been shown to be capable of supporting a very large number of processors while keeping the parameters of the system within reasonable bounds. However, these structures have been developed in an ad-hoc way. No unified methodology has been developed for the definition and analysis of hypercube structures. This is precisely the primary contribution of this work.

A generalized hypercube structure (Hyperstructure) is presented. A Hyperstructure H is constructed from simpler Structures $S_0, S_1, \dots, S_{D-1}$ (factor Structures) through a set of operations. These operations are defined and their implications considered in detail. A Hyperstructure can be thought of as a collection of multiple copies of its factor Structures. Closed form expressions are derived that calculate the parameters and characteristics of a Hyperstructure as a function of the parameters of its factor structures. The parameters examined and derived include diameter, average message path length, average message traffic and fault tolerance. Most of these parameters depend heavily on the communication pattern between processors in the Hyperstructure. A number of different assumptions are used to model the communication requirements between processors.

It will be demonstrated that many of the hypercubes discussed in the literature are in fact special cases of Hyperstructures. As such they can be easily analyzed and important conclusions about them can be reached with minimum effort. Furthermore, the defined operations suggest a number of variations of existing hypercubes resulting in structures with improved characteristics. New configurations will be presented to demonstrate this capability.

The VLSI implementation of Hyperstructures will be considered. Algorithms will be presented that allow the creation of near-optimal layouts with minimal effort. This approach is a by-product of the definition of Hyperstructures and constitutes the first minor contribution of this work. Also the problem of limiting the interconnection costs of large multiprocessor systems in a controlled fashion and with minimum impact on the functionality of the system will be addressed. This will be the second minor contribution of this work. A general approach will be developed that facilitates the customization of Hyperstructures for different applications. The optimal layout of

Hyperstructures with reduced interconnections will also be addressed.

1.2 Organization of the Thesis

In chapter 2 a review of the relevant literature will be presented. Different Hypercube-like architectures proposed in the literature will be reviewed and their relative merits summarized and analyzed. A simple classification of the different architectures is presented and this work placed in perspective.

Chapter 3 will present a unified theoretical frame for the constructive development of high performance multiprocessor systems. A set of operations will be defined that allow the manipulation of multiprocessor configurations in a mathematical way. Closed form expressions will be derived for the basic characteristics of Hyperstructures based on the characteristics of their factor Structures.

In chapter 4, the layout of Hyperstructures for VLSI implementation will be investigated. Algorithms will be presented that allow the creation of near-optimal layouts with minimal effort. Results will be presented for different architectures and compared with known layouts from the literature.

Chapter 5 will discuss the problem of limiting the interconnection costs of large multiprocessor systems in a controlled fashion and with minimum impact on the functionality of the system. A general approach will be developed that facilitates the customization of Hyperstructures for different applications. The optimal layout of Hyperstructures with reduced interconnections will also be discussed. Finally, chapter 6 will present the conclusions along with some recommendations for further work.

Chapter 2

Literature Review

The single processor bottleneck has always been one of the most controversial subjects of computer architecture. Parallel processing, in a variety of forms, has always been accepted as the answer. This took the form of micro-parallelism (where a number of processing elements are connected together to form a computer) and macro-parallelism (where a number of processors are connected into a system).

Although most interconnection structures were originally proposed for the interconnection of processing elements (such as registers and processing units), the rapid advancement of VLSI technology which led to the production of today's high-performance microprocessors and high density memories, turned the research toward multiprocessing.

Most prominent among the approaches to the interconnection problem is the Perfect Shuffle [STON 71] and its derivatives. Originally proposed for the interconnection of components, it has been used since in a variety of forms for the implementation of a large number of Multistage Interconnection Networks (MINs).

Simpler structures have been used extensively both in processor and multiprocessor designs. Among these the most widely used is the shared bus where a number of processors or processing elements are connected to and communicate through a single bus. The shared bus interconnection has been employed in the implementation of most present day mainframes. Recently [MUDG 85, DAS 85, BHUY 84, IRAN 84, MARS 82, LANG 82], the Multiple Bus has been proposed and analyzed as a method of removing the bottleneck imposed by a single bus while maintaining the advantages of a shared bus.

Other alternatives concentrated on the design of loosely coupled processors where each processor is autonomous and cooperates with other processors in performing a certain task. Networks such as the Ring and its derivatives, the Star, the Mesh and full interconnection have been used. As systems with thousands of processors became possible the hypercube interconnection and its derivatives have gained in popularity. These will be reviewed in detail in the following sections.

2.1 The Binary Hypercube

The n -dimensional cube (usually referred to as binary hypercube or simply hypercube) has been used in a number of studies and implementations. In fact, several machines are presently available based on this interconnection and more are under development. As examples we might sight the Caltech Cosmic Cube [FOX 85] and the Connection Machine [HILL 85]. Although radically different in their implementation details (MIMD vs SIMD), both machines derive many of their advantages from the hypercube interconnection topology. Figure 2.1 shows 3 and 4 dimensional binary cubes.

Figure 2.1.a 3-dimensional Cube

Figure 2.1.b 4-dimensional Cube

The popularity of the hypercube interconnection is not accidental. An excellent summary of its properties is given in [SAAD 88]. The diameter of a hypercube grows only as $\log_2(N)$ with the number of processors. Message routing is extremely simple while the large number of alternative paths connecting any two nodes provide excellent reliability and tolerance to failures. A number of other structures can be embedded in a hypercube. These include the ring, the torus, the grid and the mesh. A binary tree of size $n - 1$ can be embedded in a hypercube of size 2^n . An elegant analysis of embedding binary trees in hypercubes is given in [WAGN 85].

2.2 Other Hypercube Structures

Besides the binary hypercube, a number of similar structures have been proposed in the literature. The fact that in a binary hypercube only two processors are available along each dimension is advantageous for some applications but limiting in others. These structures generally allow for more than two processors along each dimension of the cube. The single edge along a dimension of the binary hypercube is replaced by a more complex structure allowing the interconnection of multiple processors.

In [WITT 81] the Spanning Bus Hypercube (SBH) is defined. A Spanning Bus Hypercube can have any number of processors along each dimension, interconnected by a shared bus. In general a SBH has W^D processors interconnected by DW^{D-1} buses where W is the number of processors along a dimension and D is the number of dimensions. Figure 2.2 shows a 3^3 Spanning Bus Hypercube. Compared to the binary hypercube, the SBH has a lower interconnection cost, and a smaller diameter for a given number of processors. A SBH can be expanded by adding more dimensions or

Figure 2.2 3^3 Spanning Bus Hypercube

by increasing the number of processors along each dimension.

In [WULI 81] the Spanning Bus Hypercube is analyzed along with two other strictly hierarchical structures. One important difference of this analysis is that inter-processor communication requirements are allowed to vary along different dimensions. This makes the results more realistic, especially in the case of jobs that decompose naturally in a hierarchical fashion.

In [BHUY 84a], a more general topology is discussed where the number of processors along each dimension is allowed to vary. This topology is called the Generalized Hypercube. In a Generalized Hypercube the total number of processors is

$$N = n_0 \times n_1 \times \dots \times n_{D-1}$$

but homogeneity of the structure is maintained. Two structures were presented and analyzed: The Generalized Hypercube (GHC), where processors along a dimension are fully connected via point to point links, and the Generalized Mesh Hypercube (GMHC) where processors along a dimension are connected in a loop. Figure 2.3 shows a $3 \times 2 \times 4$ GHC, while Figure 2.4 illustrated a $3 \times 2 \times 4$ GMHC. Clearly the GMHC is a generalization of the D -dimensional torus [MART 81]. The presence of an imbedded loop in both structures was also demonstrated.

2.3 Modified Hypercube Structures

Despite its many advantages, the hypercube interconnection and its derivatives also have their drawbacks. The main drawbacks cited in the literature are the increase of the number of interconnections as $N \log(N)$ with the number of processors N , and the fact that it does not lend itself for VLSI implementation. These problems have

Figure 2.3 $3 \times 2 \times 4$ Generalized Hypercube

Figure 2.4 $3 \times 2 \times 4$ Generalized Mesh Hypercube

been addressed in the literature and different solutions have been proposed.

In [PREP 81] both problems are addressed and the topology known as the Cube Connected Cycles (CCC) is proposed. In the D -dimensional CCC each processor of a D -dimensional binary hypercube is replaced by a loop of S processors. In general $S \geq D$ but usually it is assumed that $S = D$. Each processor of the loop has exactly three ports. Two of these are used by the loop connections, while the third is connected to one of the D links that were connected to the original node. Thus, each processor of the loop has a connection along one of the dimensions of the original hypercube. An example of a 3-dimensional CCC is shown in Figure 2.5. In the case that $S > D$, the extra ports are not utilized.

The diameter of the CCC is $(5D - 2)/2$ and its average message path length is $(7D - 12)/4$. The diameter and average message path length of the original binary hypercube are D and $D/2$ respectively. However this is not a fair comparison since the CCC has D times as many processors. The number of connections per node of a CCC is limited to 3 as opposed to D for the binary hypercube. This reduces the number of independent paths in the structure and correspondingly its tolerance to failures.

Besides limiting the number of ports per node, the resulting CCC topology is very suitable for VLSI implementation. Indeed it has been shown [PREP 81] that a CCC of any size can be laid-out with no more than two levels of interconnection. Figure 2.6 demonstrates this fact for a 3-dimensional CCC and illustrates the general solution.

In [WITT 81] a number of topologies were compared for the purpose of interconnecting a large number of microprocessors. The networks considered varied from

Figure 2.5 3 dimensional Cube Connected Cycles

Figure 2.6 Layout of 3-dimensional CCC

simple networks like the ring and common bus to highly complex ones like the CCC and SBH. One important aspect of the comparison is that topologies based on buses are treated similarly to topologies employing point to point connections. Many different parameters of the topologies were considered including average message path length, message density, interconnection cost and number of ports per node. Also some reliability aspects of the topologies were addressed.

The problem of the varying number of ports per node for different size structures was addressed and a new topology, the Dual Bus Hypercube (DBH), was proposed which aimed at limiting the interconnection cost while maintaining the overall flexibility and power of a hypercube interconnection. Unlike the CCC where a node is replaced by a network, the DBH limits the number of ports per node by pruning buses off an existing structure (the Spanning Bus Hypercube).

In order to obtain a D -dimensional DBH, we start with a D -dimensional SBH. All buses spanning dimension 0 are maintained. From the buses spanning dimension i (where $i = 1 \dots D - 1$) only those connected to processors for which the relation $(x_0 \bmod D) = i$ holds true are maintained. Thus each node is connected to a bus spanning dimension 0 and a bus spanning one of the other dimensions. Figure 2.7 shows a 3-dimensional Dual Bus Hypercube with three processors along each dimension.

The pruning results in a structure whose diameter and average message path length are about double those of the original Spanning Bus Hypercube. Also the number of disjoint paths between any two nodes has been reduced to 2 with a corresponding decrease in fault tolerance. However, the number of ports per node and the interconnection costs of both the CCC and the DBH are constant. This makes it

Figure 2.7 $3 \times 3 \times 3$ Dual Bus Hypercube

possible to build CCCs and DBHs of arbitrary size from standardized components.

2.4 Other structures with W^D Processors

Besides the different hypercubes, a number of other multiprocessor configurations have been proposed in the literature where the total number of processors is W^D . These were mainly targeted at fault tolerance and were based on extremal graph theory and cyclic coding [PRAD 82, PRAD 85]. The configuration proposed in [PRAD 85] is called an $FG(W,D)$ network. Also a number of topologies based on group operations have been recently presented and analyzed [AKER 89].

An FG network consists of clusters of W processors each. The processors of each cluster are fully connected via point to point connections. Depending on the values of W and D each processor has one or two extra ports that connect it to processors in other clusters. Intercluster connections of a processor are made between processors whose radix W numbers are rotated by one position. Some extra links are then added to connect processors that were not connected during the previous step. Figure 2.8 shows $FG(4,2)$.

The main advantage of the FG structures is that they allow W^D processors to be connected with an interconnection cost equal to that of a D -dimensional binary hypercube with 2^D processors. The fault tolerance of the resulting network and the number of connections per node are proportional to W rather than D as is the case for hypercubes.

On the negative side, the diameter of the network is nearly double that of a hypercube with the same number of dimensions. Also the number of redundant paths

Figure 2.8 $FG(1,2)$ network

between any pair of nodes is drastically reduced. Consequently, failures occurring in an FG network are likely to disrupt the balance of message traffic in a large section of the system, while the damage caused by several failures is not localized as is the case in hypercubes.

A special class of networks, called **symmetric interconnection networks** is addressed in [AKER 89]. By exploiting the symmetries of these graphs, a group theoretic model for interconnection networks is developed. Two types of networks are examined, the Star Graph and the Pancake Graphs. Both graphs are generated as permutations of finite sets. The implementation of certain fundamental algorithms on the Pancake Graphs is also discussed. Of particular interest to this work is the discussion on Products of Groups and Graphs, and the generation of the Cube Connected Cycles as the wreath product of S_2 and Z_n .

2.5 Discussion

A closer examination of the different structures proposed in the literature reveals two large classes of architectures. In the first class we find all the hypercube derivatives which are arrived at by substituting nodes and/or edges of the Binary Hypercube with other, larger multiprocessor structures. These structures retain the simple routing algorithm of the Binary Hypercube (adjusted for the structure used), and provide improved behaviour in one or more of the other parameters (diameter, message density, fault tolerance). Invariably these structures have been produced in an ad-hoc way.

In the other category we find architectures arrived at from a mathematical

origin. These structures are derived from various group or coding operations. Such structures will typically be optimal with respect to certain parameters, depending on the operations used for their generation. It is safe to say that since these configurations are direct mappings of certain operations, they will be ideal for machines primarily intended for such operations. On the other hand the Hypercube Derivatives attempt to combine the advantages of the Binary Hypercube (which itself belongs to the second category), with the advantages of certain (simpler) configurations known to provide good characteristics in general purpose systems.

Despite the large number of structures proposed in the literature and the rich mathematical background available, the problem of generating interconnection networks has only been addressed recently [AKER 89] for structures in the second category. The corresponding problem for hypercube derivatives has not been addressed at all. The following chapters will present a mathematical model that will not only abstract the structure and properties of Hypercube Derivatives, but will also offer a fertile source of other similar topologies.

Chapter 3

Hyperstructures

This chapter deals with the definition of the Multiprocessor Hyperstructures and their properties. Hyperstructures are a generalization of Hypercubes and will be shown to include all hypercubes discussed in the literature while allowing the rapid evaluation of other novel (hypercube) configurations. In section 3.1 a definition of the Hyperstructures is given. A general Multiprocessor Structure is defined along with a set of operations on it which produce Hyperstructures. Section 3.2 deals with the properties of Hyperstructures concentrating on interprocessor communications and fault tolerance. In section 3.3 Hyperstructures where all the processors are connected through point to point connections are examined in detail. Such Hyperstructures warrant special attention because of their wide use in the form of the binary hypercube and because they can be analyzed to greater detail than their more general counterparts.

3.1 Definition

This section defines the basic operations and constructs that will be used throughout this work. Section 3.1.1 describes a general Multiprocessor Structure. A number of parameters are defined that will be used to characterize different structures. In all cases it is assumed that the processors of a structure are homogeneous in terms of processing capacity although they may differ in terms of the number of communication ports they have.

Section 3.1.2 describes a mixed radix numbering scheme which will be used to uniquely identify nodes and interconnections within a Hyperstructure. This numbering scheme is conceptually similar to the one introduced in [BHUY 84a], extended to include the numbering of interconnections. Section 3.1.3 defines the Cartesian product of Multiprocessor Structures which is the "generating function" of Multiprocessor Hyperstructures, along with its mathematical properties. Finally, in section 3.1.4 the Multiprocessor Hyperstructure is defined.

3.1.1 Multiprocessor Structures

In general, a Multiprocessor structure can be thought of as a set of nodes (processors) N communicating through some medium (interconnection network) I as shown in Figure 3.1. The Structure as a whole is characterized by:

W the number of nodes.

$DIAM$ the maximum number of nodes a message may have to go through to reach its destination.

Figure 3.1 Multiprocessor Structure

MPL The average number of hops a message has to do to reach its destination, assuming message destinations are uniformly distributed among all processors in the Structure, including the originator.

MPL* The average number of hops a message has to do to reach its destination, assuming each processor uniformly communicates with all other processors in the Structure.

$D(X_1, X_2)$ The maximum number of hops a message has to do to reach node X_2 starting from node X_1 .

Lemma 3.1 Assuming that a processor sends messages to all processors in the Structure, excluding itself, with equal probability (**Assumption 1***), the average

message path length is:

$$MPL^* = \frac{W}{W-1} MPL \quad (3.1)$$

Proof 3.1 Taking the average of the distances between processors, and accounting for the fact that a processor's distance from itself is zero, we have:

$$\begin{aligned} MPL^* &= \frac{1}{W(W-1)} \sum_{X,Z, X \neq Z} D(X,Z) \\ &= \frac{W}{W-1} \frac{1}{W} \sum_{X,Z} D(X,Z) \\ &= \frac{W}{W-1} MPL \end{aligned}$$

Q.E.D.

Nodes are connected to the interconnection network through ports. Depending on the type of interconnection network used, different types of ports may be employed. A port is characterized by:

PMD_k The maximum message density (number of messages per time unit) a port of type k can handle.

Node ports are connected to interconnection network ports of the same type. Note that this implies a change of direction for unidirectional ports. Thus if ports of type i are inputs of the nodes, they are outputs of the interconnection network and vice-versa.

For the purpose of this work we assume that only the nodes are capable of performing computations. The interconnection network is strictly passive and is mainly used for communication between the processors. Optionally the interconnection network may be capable of storing information. In general, information stored can be considered as part of the interprocessor communication activity.

Each node is a combination of a computation oriented main processor and a communications oriented front-end processor as shown in Figure 3.2. The main processor handles all computations and local (internal) communications whereas the front-end processor handles all communications with other processors including any message traffic that has to go through the node. Typically the front-end processor would be a special purpose processor optimized for message handling [WITT 81]. The front-end processor of a node is characterized by:

NNP_k^j the number of ports of type k on node j .

NMD the maximum message density a node can handle, excluding internal messages.

$NiMD$ the maximum message density that can be passed between the main and front-end processor of a node.

Theorem 3.1 For any node of a Multiprocessor Structure it holds that:

$$NMD_j \leq \frac{1}{2}(NiMD + \sum_k (NNP_k^j \times PMD_k)) \quad (3.2)$$

Proof 3.1 Clearly a node's message throughput cannot exceed the sum of the throughputs of its ports. However, if different types of unidirectional ports are used, it is possible that the overall throughput will be limited by the throughput of a particular type of ports. The factor of $1/2$ comes from the fact that, any message has to go through two ports (one entering the front-end processor and one leaving it).

In practice it is unlikely that all communication channels will be 100% utilized. Consequently, the front-end processor would be designed to only handle the average message throughput so that the inequality in 3.2 will hold.

Figure 3.2 Block diagram of Typical Node

The exact form of the interconnection network is of limited importance in this work. Instead we are interested in the capacity of the interconnection network to support interprocessor communication. An interconnection network will be characterized by:

NIP_k the number of ports of type k .

IMD the maximum message density the interconnection network can handle.

IMD_k the maximum message density the interconnection network can handle through ports of type k .

It should be noted that depending on the type of interconnection the number of ports may vary arbitrarily. For example, in the case of a shared common bus, the

number of processors connected to the bus may vary within certain bounds. Thus NIP_k could be a variable for some k . In such cases, the number of ports required for the implementation of the Multiprocessor Structure under consideration will be assumed.

Theorem 3.2 For any interconnection network of any Multiprocessor Structure, the following relations hold:

$$IMD_k \leq (NIP_k \times PMD_k) \quad (3.3)$$

$$IMD \leq \frac{1}{2} \sum_k IMD_k \quad (3.4)$$

Proof 3.2 Clearly an interconnection network cannot handle more messages than the throughput of its ports will allow. However, for certain designs it is possible that message throughput will be limited by the internal characteristics of the interconnection network.

If messages enter and leave the interconnection network through ports of the same type, then the equality in equation 3.4 is obvious. However, it is possible that two or more types of unidirectional ports (say i and j) will be employed in the interconnection network. In this case the message throughput contributed by these two types of ports will be limited by:

$$\min(IMD_i, IMD_j) < \frac{1}{2}(IMD_i + IMD_j)$$

and 3.4 follows.

Since each port of a node is connected to a port (of the same type) of the interconnection network it follows that:

$$NIP_k = \sum_k NNP_k^j \quad (3.5)$$

$$NP = \sum_k NIP_k \quad (3.6)$$

where NP is the total number of ports of all types. We define

$$\eta_N = \eta_I = \{1, 2, \dots, NP\}$$

to be the numbered sets of ports of the nodes and the interconnection network respectively. Now we can define the connection of the nodes to the interconnection network as a monomorphism $C : \eta_N \rightarrow \eta_I$. Clearly C^{-1} exists and describes the connection of the interconnection network to the nodes. Thus a Multiprocessor Structure can be expressed as a triad $\{N, I, C\}$.

Example 3.1.1.1 Consider the Multiprocessor Structure shown in Figure 3.3 where a set of 8 processors is connected in a loop via bidirectional links. For this structure we have:

$$W = 8$$

$$NNP_0^0 = NNP_0^1 = NNP_0^2 = NNP_0^3 =$$

$$NNP_0^4 = NNP_0^5 = NNP_0^6 = NNP_0^7 = 2$$

$$NIP_0 = 16$$

$$DIAM = 4$$

$$MPL = 2$$

$$MPL^* = 2\frac{2}{7}$$

$$D(x_1, x_2) = \min(|x_1 - x_2|, 8 - |x_1 - x_2|)$$

while 3.2 and 3.3 hold with equality.

Example 3.1.1.2 Consider the Multiprocessor Structure shown in Figure 3.4 where a set of 4 processors is connected by a common bus. For this structure we have:

Figure 3.3 Multiprocessor Structure with loosely coupled processors

$$W = 4$$

$$NNP_0^0 = NNP_0^1 = NNP_0^2 = NNP_0^3 = 1$$

$$NIP_0 = 4$$

$$DIAM = 1$$

$$MPL = 0.75$$

$$MPL^* = 1.0$$

$$D(x_1, x_2) = 1$$

while the maximum message density through the bus is:

$$IMD = \min(MD_{bus}, 2PMD_0)$$

Example 3.1.1.3 Consider the Multiprocessor Structure shown in Figure 3.5 where a set of 4 processors is connected through a shared memory accessible via a

Figure 3.4 Multiprocessor Structure with Shared Bus

common bus. For this structure we have:

$$W = 4$$

$$NNP_0^0 = NNP_0^1 = NNP_0^2 = NNP_0^3 = 1$$

$$NIP_0 = 4$$

$$DIAM = 1$$

$$MPL = 0.75$$

$$MPL^* = 1.0$$

$$D(x_1, x_2) = 1$$

while the maximum message density through the bus (memory) is:

$$IMD = \min(MD_{bus}, MD_{memory}, 2PMD_0)$$

Example 3.1.1.4 Figure 3.6 shows a Multiprocessor Structure consisting of 4 processors connected via a multiport memory. For this structure we have:

Figure 3.5 Multiprocessor Structure with Shared Memory

$$W = 4$$

$$NNP_0^0 = NNP_0^1 = NNP_0^2 = NNP_0^3 = 1$$

$$NIP_0 = 4$$

$$DIAM = 1$$

$$MPL = 0.75$$

$$MPL^* = 1.0$$

$$D(x_1, x_2) = 1$$

while the maximum message density through the memory is:

$$IMD = \min(MD_{memory}, 2PM D_0)$$

It should be noted that the exact form of the connection function C is important only for the Multiprocessor structure in example 3.1.1.1.

Figure 3.6 Multiprocessor Structure with Multiport Memory

3.1.2 A Mixed Radix System

The numbering scheme described in this section is similar to one introduced in [BHUY 84a], extended to include the numbering of interconnections as well as the numbering of processors.

Let W , the total number of processor in a system be expressible as:

$$W = \prod_{i=0}^{D-1} W_i$$

Then each processor can be denoted as a D -tuple

$$x_{D-1}x_{D-2}x_{D-3} \dots x_1x_0$$

where $0 \leq x_i < W_i$ for $i = 0, 1, \dots, D-1$. In order to convert between this mixed radix system and decimal numbering we define a set of weights w_i , $i = 0, 1, \dots, D-1$,

such that for each processor X , where $0 \leq X < W$, we have

$$X = \sum_{i=0}^{D-1} x_i w_i$$

and $w_i = W_{i-1} w_{i-1}$ for all $0 < i < D$, and $w_0 = 1$.

Example 3.1.2.1 Let $W_0 = 5$, $W_1 = 4$, $W_2 = 3$. Then $W = 60 = 5 \times 4 \times 3$ and $w_0 = 1$, $w_1 = 5$, $w_2 = 20$. Therefore $X = x_2 x_1 x_0$ such that $x_0 = 0, \dots, 4$, $x_1 = 0, \dots, 3$, $x_2 = 0, \dots, 2$ for any $X = 0, \dots, 59$. Then $0_{10} = (000)$, $26_{10} = (111)$ and $59_{10} = (234)$ in this mixed radix system.

Let the interconnection network of a Multiprocessor Structure S include WI_i disjoint copies of the interconnection network of a Structure S_i (interconnection subnetworks) and

$$WI_i = \prod_{j=0, j \neq i}^{D-1} W_j$$

Then each copy of the interconnection network (interconnection subnetwork) Y_i , where $Y_i \in \{0 \dots WI_i - 1\}$ can be expressed as a D -tuple

$$\forall D-1 y_{D-2} \dots y_{i+1} \psi y_{i-1} \dots y_1 y_0$$

$0 \leq y_i < W_i$ for $i = 0, 1, \dots, D-1$. Note the place holder ψ used to make the numbering of interconnection subnetworks compatible with the numbering of processors. Using the weights w_i we have

$$Y_i = \sum_{j=0, j \neq i}^{D-1} y_j w_j$$

3.1.3 Cartesian Product of Structures

Definition 1 Let $S_0, S_1, \dots, S_{D-1}$ be Multiprocessor Structures consisting of

$$W_0, W_1, \dots, W_{D-1}$$

processors, connected through interconnection networks

$$IN_0, IN_1, \dots, IN_{D-1}$$

respectively. The Cartesian Product of $S_0, S_1, \dots, S_{D-1}$ is defined to be a Multiprocessor Structure S consisting of $W = W_0 \times W_1 \times \dots \times W_{D-1}$ processors connected through an interconnection network consisting of WI_i copies (interconnection sub-networks) of IN_i for $i = 0, 1, \dots, D-1$. The Cartesian Product of Structures will be denoted as:

$$S_0 \times S_1 \times \dots \times S_{D-1}$$

Let also the processors and interconnection networks of S be numbered according to the mixed radix system defined in section 3.1.2. Then processor $(x_{D-1}x_{D-2} \dots x_1x_0)$ is connected to networks $(y_{D-1}y_{D-2} \dots y_{i+1}\psi y_{i-1} \dots y_1y_0)$ where $x_j = y_j$ for $j = 0, 1, \dots, D-1$; $j \neq i$ and $i = 0, 1, \dots, D-1$. Figure 3.7 illustrates the Cartesian Product of two simple Multiprocessor Structures.

The connection of processor $(x_{D-1}x_{D-2} \dots x_1x_0)$ to interconnection network $(y_{D-1}y_{D-2} \dots y_{i+1}\psi y_{i-1} \dots y_1y_0)$ is identical to the connection of processor x_i of S_i to interconnection network IN_i for $i = 0, 1, \dots, D-1$. Processor x_i of S_i is called the **projection** of $(x_{D-1}x_{D-2} \dots x_1x_0)$ on S_i for $i = 0, 1, \dots, D-1$. The Multiprocessor Structures $S_0, S_1, \dots, S_{D-1}$ will be called the **factor Structures** or simply **factors** of S .

From the definition it is easy to see that the Cartesian Product of Multiprocessor Structures is commutative and associative. With the nodes of the resulting Multiprocessor Structure numbered according to the mixed radix system described in section 3.1.2, a change in the order the Structures are multiplied results in a simple

Figure 3.7 Structure Multiplication

rearrangement of the indexes. Figure 3.8 graphically demonstrates the commutativity of the Cartesian Product of Structures.

If we accept the trivial Structure U consisting of a single node ($W = 1$) and no interconnection network ($I = \emptyset$), then U is the unit of the Cartesian Product of Structures. Thus the set of Multiprocessor Structures with the Cartesian multiplication of Structures form an Abelian Monoid.

A Structure will be called **prime** (or **indecomposable**) with respect to Cartesian multiplication if $S = S_0 \times S_1$ implies either $S_0 \cong U$ or $S_1 \cong U$. It should be noted that in the special case where all the interconnection networks of the factor structures consist only of point to point connections, the Cartesian multiplication of Structures becomes equivalent to the Cartesian multiplication of graphs as defined in [SABI 60].

A Structure will be called **uniform** if the distribution of nodes with respect to distance from a node X is invariant across all the nodes of the Structure.

3.1.4 Multiprocessor Hyperstructure

Definition 2 A (Multiprocessor) Hyperstructure H is a Structure obtained as the Cartesian Product of Multiprocessor Structures. However, the various interconnection subnetworks are not merged but are retained as separate, disjoint, copies of the interconnection networks of the factor Structures. Thus a Hyperstructure is a 4-tuple $\{N, W, I, C\}$ where:

- N • the set of nodes of H .

Figure 3.8 Commutativity of Structure Multiplication

- W • an ordered set of the numbers of nodes W_i of the factor Structures of H .
- I • an ordered set of the interconnection networks of the factor Structures.
- C • an ordered set of the connection functions of the factor Structures.

For clarity, the parameters of the factor Structures will be denoted with one (extra) index denoting the Structure to which they belong. The parameters of a Hyperstructure generated as

$$S_0 \times S_1 \times \dots \times S_{D-1}$$

can be calculated from the parameters of its factor Structures. The number of ports of a node of H is the sum of the numbers of ports of the projections of the node on all the factor Structures

$$NNP_k^{s_{D-1}s_{D-2}\dots s_1s_0} = \sum_{i=0}^{D-1} NNP_{i,k}^{s_i}$$

Similarly the number of ports of the interconnection network of H can be calculated as

$$NIP_k = \sum_{i=0}^{D-1} WI_i \times NIP_{i,k}$$

Theorem 3.3 For any two nodes X, Z of a Hyperstructure H , their distance is:

$$D(X, Z) = \sum_{i=0}^{D-1} D_i(x_i, z_i) \quad (3.7)$$

Proof 3.3 In order to prove the existence of a path of this length we define X_j to be the node $x_{D-1} \dots x_{j+1} z_j \dots z_1 z_0$. From the definition of a Hyperstructure it follows that there exists a path between X and Z of the form

$$X \rightarrow X_0 \rightarrow X_1 \rightarrow \dots \rightarrow X_{D-2} \rightarrow Z$$

with length

$$L_P = D(X, X_0) + D(X_0, X_1) + \dots + D(X_{D-2}, Z)$$

To prove that this is the shortest path between X and Z we assume that there exists a path P' such that $L_{P'} < L_P$. This would mean that for some $j \in (0 \dots D-1)$ there exists a path between X_{j-1} and X_j with length less than $D(X_{j-1}, X_j)$, where $X_{-1} = X$ and $X_{D-1} = Z$. Since this is impossible it follows that P is a shortest path between X and Z .

Q.E.D

3.2 Properties of Hyperstructures

In this section various properties of Hyperstructures will be discussed in detail. Closed form expressions will be derived that allow the calculation of the properties of any Hyperstructure as a function of the properties of its factor Structures. In all cases examples will be given utilizing hypercubes proposed in the literature.

In section 3.2.1 the diameter of Hyperstructures will be analyzed. Section 3.2.2 will present an analysis of the communication requirements of Hyperstructures under different assumptions. The analysis will deal with bandwidth requirements for processors and interconnection networks. The effect of communication locality will be analyzed. The Average Message Path Length, Network Message load and Processor

Message load will then be analyzed under different assumptions in sections 3.2.3, 3.2.4 and 3.2.5 respectively.

In section 3.2.6 the fault tolerance of Hyperstructures will be analyzed. The effects of failures to the operation of a Hyperstructure will be examined in detail. Section 3.2.7 will present examples of the use of the results of this chapter in the analysis of a Hyperstructure. A number of hypercubes described in the literature will be examined and the results validated against those in the literature.

3.2.1 Diameter

Theorem 3.4 The diameter ($DIAM$) of a Hyperstructure H is:

$$DIAM = \sum_{i=0}^{D-1} DIAM_i \quad (3.8)$$

Proof 3.4 Taking the maximum of both sides of equation 3.4 we have:

$$\max(D(X, Z)) = \max\left(\sum_{i=0}^{D-1} D_i(x_i, z_i)\right)$$

Since the projections of a path on each factor Structure are independent of each other we can distribute the maximum over the summation:

$$\max(D(X, Z)) = \sum_{i=0}^{D-1} \max(D_i(x_i, z_i))$$

Q.E.D.

Example 3.2.1.1 Consider the $4 \times 3 \times 2$ GMHC shown in Figure 2.4. For that Hyperstructure we have:

$$DIAM_0 = 2$$

$$DIAM_1 = 1$$

$$DIAM_2 = 1$$

Applying 3.8 we obtain:

$$DIAM = DIAM_0 + DIAM_1 + DIAM_2 = 2 + 1 + 1 = 4$$

which agrees with the diameter given in [BHUY 84].

3.2.2 Interprocessor Communication Requirements

The communication requirements of a Hyperstructure are strongly dependent on the communication pattern between processors. The simplest model of interprocessor communication is that a message generated on a processor needs to go to any processor in the system (including the originator) with equal probability (Assumption 1). While simple to analyze this assumption leads to overly pessimistic results. However it is useful as a first approximation and has been used extensively in the literature.

In order to take into consideration the locality that is likely to exist in any algorithm it has been suggested to assume that the probability that a processor is the destination of a message decreases with the distance from the originator according to some function (Assumption 2).

Both of the above assumptions presume an existing structure and examine the effects of mapping an algorithm onto it. Since the major objective of this work is the constructive development of Hyperstructures as computer architectures for classes of problems we introduce an assumption that presumes some knowledge of the algorithm(s).

A message generated by a processor $x_{D-1}x_{D-2}\dots x_1x_0$ has as destination processor $x_{D-1}x_{D-2}\dots x_1x_1y$ (where $y = 0, 1, \dots, W_0 - 1$) with probability α and any other processor in the system with probability $1 - \alpha$. The probability that any particular processor within either group is the destination of the message is uniform (Assumption 3).

In other words we assume that communication is localized inside clusters but not otherwise restricted. Such behaviour is typical of electronic CAD algorithms where the communication requirements are directly proportional to the number of signal lines running between groups of devices. This number increases in proportion to the logarithm of the number of gates in a cluster until a self contained unit is built (e.g. an ASIC chip) at which point it drops dramatically restricted by I/O pin limitations.

3.2.3 Average Message Path Length

Theorem 3.5 Under assumption 1 the average message path length can be calculated as:

$$MPL = \sum_{i=0}^{D-1} MPL_i \quad (3.9)$$

Proof 3.5 With uniform communications the average message path length is equal to the average distance between processors. Thus we have:

$$\begin{aligned}
MPL &= \frac{1}{W^2} \sum_{X,Z} D(X,Z) \\
&= \frac{1}{W^2} \sum_{X,Z} \sum_{i=0}^{D-1} D_i(x_i, z_i) \\
&= \frac{1}{W^2} \left(\sum_{i=0}^{D-1} W I_i^2 \times \sum_{x_i, z_i} D_i(x_i, z_i) \right) \\
&= \sum_{i=0}^{D-1} \left(\frac{1}{W^2} \sum_{x_i, z_i} D_i(x_i, z_i) \right) \\
&= \sum_{i=0}^{D-1} MPL_i
\end{aligned}$$

Q.E.D.

Applied to binary hypercubes, assumption 2 resulted in a uniform decrease in message traffic. This result can be generalized according to the following theorem.

Theorem 3.6 Communication requirements varying only with distance yield a uniform message load on a uniform Hyperstructure.

Proof 3.6 Consider the amount of message traffic that has to be processed by a node X of a Hyperstructure. This is proportional to the number of messages that have to make i hops and are generated by nodes whose distance from X is at most i . Thus each message going through k hops is seen by k nodes. Since the structure is uniform, each processor will see a number of messages equal to the weighted sum of all the messages it generates.

Q.E.D.

Clearly it is undesirable to map an algorithm with uniform communication requirements to a non-uniform Hyperstructure. Since for uniform Hyperstructures the results obtained under assumption 2 are identical to the results under assumption 1

(within a constant factor), assumption 2 will not be analyzed in detail in the remainder of this work.

Theorem 3.7 Under assumption 3 the average message path length on a Hyperstructure is:

$$MPL = MPL_0 + (1 - \alpha) \frac{WI_0}{WI_0 - 1} \sum_{i=1}^{D-1} MPL_i \quad (3.10)$$

Proof 3.7 Let $H' = S_1 \times S_2 \times \dots \times S_{D-1}$. Then it holds that:

$$H = S_0 \times H'$$

and

$$MPL = \alpha MPL_0 + (1 - \alpha)(MPL_0 + MPL^*)$$

Substituting MPL^* from Lemma 3.1 and taking advantage of the fact that $W' = WI_0$ we have:

$$\begin{aligned} MPL &= \alpha MPL_0 + (1 - \alpha)(MPL_0 + \frac{W'}{W'-1} \sum_{i=1}^{D-1} MPL_i) \\ &= MPL_0 + (1 - \alpha) \frac{WI_0}{WI_0 - 1} \sum_{i=1}^{D-1} MPL_i \end{aligned}$$

Q.E.D.

3.2.4 Network Message Load

In this section the message load of the interconnection subnetworks of a Hyperstructure will be examined. Since transmission of messages across different subsystems (along different dimensions) is independent, we can consider each message hop as a separate message transmission. Thus the total number of messages that must be handled by the interconnection network of a Hyperstructure is:

$$\overline{IMD} = W \times MPL + \overline{PMD}$$

although the actual number of (different) messages generated is only:

$$\overline{DMD} = W \times \overline{PMD}$$

where $\overline{PMD}$ is the number of messages generated by each processor per time unit. Since the network message load and subsequently the required network bandwidth depend directly on the message path length, they will also be examined under the assumptions outlined in section 3.2.2.

Theorem 3.8 Under assumption 1 the Average Message Density that must be handled by interconnection subnetwork i (the interconnection network of S_i) is:

$$\overline{IMD}_i = (W_i - 1)\overline{PMD} \quad (3.11)$$

Proof 3.8 The probability that a given message must travel along dimension i is $(W_i - 1)/W_i$. Since there are W_i copies of interconnection subnetwork i it follows that:

$$\begin{aligned} \overline{IMD}_i &= \frac{W_i - 1}{W_i} \overline{DMD} \\ &= \frac{W_i - 1}{W} W \overline{PMD} \\ &= (W_i - 1)\overline{PMD} \end{aligned}$$

Q.E.D.

Theorem 3.9 Under assumption 3 the Average Message Density that must be handled by interconnection subnetwork i is:

$$\overline{IMD}_0 = (W_0 - 1)\overline{PMD} \quad (3.12)$$

$$\overline{IMD}_i = (1 - \alpha)(W_i - 1)\overline{PMD} \quad (3.13)$$

Proof 3.9 The probability that a given message must travel along dimension 0 is independent of α . Therefore the relation for IN_0 is directly derived from theorem 3.8. Given that a message must travel along any other dimension (probability $1 - \alpha$), the probability that it must travel along dimension i is

$$\frac{W_i - 1}{W_i}$$

and 3.13 follows.

Q.E.D.

3.2.5 Processor Message Load

In this section the Message load of a (front-end processor) of a node will be examined. If we examine any one of the factor structures (say S_i) of a Hyperstructure H alone we find that the total number of messages that need to travel through its interconnection network is:

$$(W_i - 1)\overline{PMD}$$

Theorem 3.10 The Average Message Load that must be handled by the front-end processor of node $x_{D-1}x_{D-2} \dots x_1x_0$ under Assumption 1 is:

$$\overline{NMD}^{x_{D-1}x_{D-2} \dots x_1x_0} = \sum_{i=0}^{D-1} \overline{NMD}_i^{x_i} \quad (3.14)$$

Proof 3.10 The total number of messages that must travel along dimension (interconnection subnetwork) i of H is:

$$IMD_i = (W_i - 1)\overline{PMD}$$

which is the number of messages that would travel through the interconnection network of S_i if factor Structure S_i was alone. Therefore the average number of messages that must be handled by the front-end processor of a node through its connection to interconnection subnetwork i is the same as that of its projection on S_i . Since traffic along different dimensions is independent, their contribution to the message load of a node is additive.

Q.E.D.

Theorem 3.11 The Average Message Load that must be handled by the front-end processor of node $x_{D-1}x_{D-2} \dots x_1x_0$ under Assumption 3 is:

$$\overline{NMD}^{x_{D-1}x_{D-2} \dots x_1x_0} = \overline{NMD}_0^{x_0} + (1 - \alpha) \sum_{i=1}^{D-1} \overline{NMD}_i^{x_i} \quad (3.15)$$

Proof 3.11 By reasoning along the same lines as for Theorem 3.10 and using Theorem 3.9 3.15 follows.

Q.E.D.

3.2.6 Fault Tolerance

Traditionally the Fault Tolerance of multiprocessor systems has been measured as the minimum number of nodes or interconnections that could cause the disconnection of one or more nodes. This definition can be trivially extended to Hyperstructures.

Theorem 3.12 The node (interconnect) Fault Tolerance of a Hyperstructure $S_0 \times S_1 \times \dots \times S_{D-1}$ is the sum of the node (interconnect) Fault Tolerances of its

factor Structures.

$$FT^{node} = \sum_{i=0}^{D-1} FT_i^{node}$$

$$FT^{interconnect} = \sum_{i=0}^{D-1} FT_i^{interconnect}$$

Proof 3.12 The node Fault Tolerance of a Hyperstructure is directly proportional to the number of nodes that can be reached with a single hop. The interconnect Fault Tolerance of a Hyperstructure is directly proportional to the number of interconnection network each node is connected to. Therefore 3.15 follows from the construction of a Hyperstructure.

Although useful for an initial estimate this approach produces overly pessimistic results in the case of Hyperstructures. The failure of D nodes in an D -dimensional hypercube will only cause a disconnection if all D failed nodes are neighbors of a single node. Even if this is the case a total of $D + 1$ nodes will become unavailable. Clearly it is necessary to include both the probability of a disconnection and the maximum damage that may be caused in order to obtain a more realistic measurement.

Given the potentially large number of processors in a Hyperstructure it is important to consider the effects of failures on the operation of the system along with the minimum number of failures that could make the system unusable. To accomplish this a new metric is needed that will provide a quantitative measure of the impact of failures when they occur in such a manner that they do not interact with each other. Assuming that the failure of a node does not result in the loss of critical information, in the long term the failure will affect only the communication of the remaining nodes. To accomplish this the Fault Tolerance of a Hyperstructure is defined as:

Definition 3 The Fault Tolerance of a Structure (Hyperstructure) is the average of the normalized number of different (not necessarily disjoint) shortest paths that exist between any two of its nodes.

The normalized number of different shortest paths between a pair of nodes X, Z is defined to be the number of paths of length $D(X, Z)$ divided by $2^{D(X, Z)-1}$. If this quotient is greater than one then one is used instead. The following table gives the different measures of Fault Tolerance for a number of Hyperstructures proposed in the literature. It also demonstrates that the metric proposed here provides a good indication of the capability of a configuration to operate without degradation in the presence of failures.

Structure	Fault Tolerance	FT^{node}	$FT^{interconnect}$
GMHC 8×8	0.66	4	4
GMHC $4 \times 4 \times 4$	1	6	6
GRID 8×8	0.39	2	2
SBH 6×8	1	14	2
SBH $4 \times 4 \times 4$	1	9	3
GHC 8×8	1	14	14
GHC $4 \times 4 \times 4$	1	9	9
Q_6	1	6	6
Q_7	1	7	7
Q_8	1	8	8

3.2.7 Examples

In this section a number of hypercube structures presented in the literature will be extensively analyzed based on the results of the previous sections. Also a new structure will be presented that will be shown to have superior characteristics for applications where the message traffic follows assumption 3.

Example 3.2.7.1 Consider the **SBH** as defined in [WITT 79]. For a **SBH** we have:

$$W_{D-1} = W_{D-2} = \dots = W_1 = W_0 = W$$

$$NNP = D$$

$$MPL = D(W - 1)/W$$

$$\overline{IMD} = (W - 1)\overline{PMD}$$

$$WI = DW^D$$

$$FT^{node} = D$$

which agree with the analysis of the **SBH** presented in [WITT 79].

Example 3.2.7.2 Consider the **GHC** as defined in [BHUY 84]. For a **GHC** we have:

$$NNP = \sum_{i=0}^{D-1} (W_i - 1)$$

$$MPL = \sum_{i=0}^{D-1} \frac{W_i - 1}{W_i}$$

$$\overline{IMD}_i = (W_i - 1)\overline{PMD}$$

$$WI = \sum_{i=0}^{D-1} W_i(W_i - 1)$$

$$FT^{node} = \sum_{i=0}^{D-1} W_i$$

which agree with the analysis of the **GHC** presented in [BHUY 84].

Example 3.2.7.3 Consider the Hyperstructure shown in Figure 3.9 where S_0 is a cluster of processors communicating through a common bus and S_1, S_2 are simple links. For this structure we have (assuming $\alpha = 0.5$):

$$NNP = 3$$

$$MPL = 4/3$$

$$\overline{TMD}_0 = 2\overline{PMD}$$

$$\overline{TMD}_1 = 0.5\overline{PMD}$$

$$\overline{TMD}_2 = 0.5\overline{PMD}$$

$$WI_0 = 4$$

$$WI_1 = WI_2 = 6$$

$$FT^{node} = 4$$

Example 3.2.7.4 Consider the Hyperstructure shown in Figure 3.10 where S_0 is a cluster of processors with shared multiport memory and S_1 is a cluster of processors that communicate through a shared memory accessible through a common bus. For this structure we have:

	$\alpha = 0.75$	$\alpha = 0.50$	$\alpha = 0.25$
NNP	2	2	2
MPL	1.0	1.25	1.5
$\overline{TMD}_0$	$3\overline{PMD}$	$3\overline{PMD}$	$3\overline{PMD}$
$\overline{TMD}_1$	$0.75\overline{PMD}$	$1.5\overline{PMD}$	$2.25\overline{PMD}$
WI_0	4	4	4
WI_1	4	4	4
FT^{node}	6	6	6

Figure 3.9 Multiprocessor Hyperstructure

Figure 3.10 Multiprocessor Hyperstructure with Shared Memory

3.3 Hyperstructures with Point to Point Connections

In this section Hyperstructure whose interconnection network consists entirely of point to point connections will be examined in detail. The direct mapping of such Structure to regular graphs makes it possible to obtain useful results not available for the general case. Furthermore, the binary hypercube and the torus, which are among the most used interconnections, fall in this category.

For Multiprocessor Structure with point to point connections the concepts of a substructure, a spanning substructure and disjoint substructures are defined similar to the definitions of a subgraph, a spanning subgraph and disjoint subgraphs respectively. Although these concepts could be extended to more general Structure, the practical implications of such an extension would be rather limited.

Lemma 3.2 Let $H = S_0 \times S_1 \times \dots \times S_{D-1}$ and $H' = S'_0 \times S'_1 \times \dots \times S'_{D'-1}$ be Hyperstructures and

$$S_0, S_1, \dots, S_{D-1}, S'_0, S'_1, \dots, S'_{D'-1}$$

are such that we can define an epimorphism

$$F : \{S'_0, S'_1, \dots, S'_{D'-1}\} \rightarrow \{S_0, S_1, \dots, S_{D-1}\}$$

such that $S'_i \subseteq f(S'_i)$ for $i = 0, 1, \dots, D' - 1$. Then $H' \subseteq H$.

Proof 3.2 We can assume without any loss of generality that $S_i = f(S'_i)$ for $i = 0, 1, \dots, D' - 1$. Then we can define a set of epimorphisms

$$F_i : \{0, 1, \dots, W(S'_i) - 1\} \rightarrow \{0, 1, \dots, W(S_i)\}$$

for $i = 0, 1, \dots, D' - 1$, such that $F_i(x'_i)$ is the node of S_i that corresponds to X'_i .

Then for each node $X' \in H'$ we select a distinct node

$$X = 0_{D-1} \dots 0_{D'} F_{D'-1}(x_{D'-1}) \dots F_1(x_1) F_0(x_0) \in H$$

By iterating over all X' we select a substructure of H isomorphic to H' .

Q.E.D.

Corollary: If $D = D'$ and S'_i is a spanning substructure of $F(S_i)$ then H' is a spanning substructure of H .

These relationships are graphically demonstrated in Figures 3.11 and 3.12 respectively for the case of two simple Structure.

Lemma 3.3 Let H be a Hyperstructure $S_0 \times S_1 \times \dots \times S_{D-1}$ and let there exist S'_i, S''_i such that $S'_i \subset S_i, S''_i \subset S_i$ and S'_i, S''_i are disjoint for some $i = 0, 1, \dots, D - 1$. Then

$$H' = S_{D-1} \times \dots \times S'_i \times \dots \times S'_1 \times S'_0$$

and

$$H'' = S_{D-1} \times \dots \times S''_i \times \dots \times S''_1 \times S''_0$$

are disjoint substructures of H .

Proof 3.3 Since the projections of H' and H'' on S_i are disjoint it follows from the definition of Hyperstructures that H' and H'' are also disjoint.

Q.E.D.

Figure 3.11 Cartesian Product of Substructures

Figure 3.12 Cartesian Product of Spanning Substructures

3.3.1 Binary Hypercube

The Binary Hypercube has become popular as a multiprocessor configuration during the past few years. Part of its popularity is due to the large number of other configurations that can be simulated efficiently on a Hypercube. In this section a number of generalized results are presented concerning the embedding of multidimensional grids and meshes in a Binary Hypercube.

Theorem 3.13 Let $H = C_{W_{D-1}} \times \dots \times C_{W_1} \times C_{W_0}$ where C_j is the cycle of length j and all W_i , $i = 0, 1, \dots, D-1$, are even. Then let n_i be such that $2^{n_i} \geq W_i$ for $i = 0, 1, \dots, D-1$ and $n = n_{D-1} + \dots + n_1 + n_0$. Then

$$C_{W_{D-1}} \times \dots \times C_{W_1} \times C_{W_0} \subseteq Q_n \quad (3.16)$$

Proof 3.13 The existence of all even cycles of length up to 2^n in Q_n has been shown in [SAAD 88]. From the definition of the Binary Hypercube it follows that $Q_n = Q_{n_{D-1}} \times \dots \times Q_{n_1} \times Q_{n_0}$. Then by applying Lemma 3.2 we obtain 3.16 directly Q.E.D.

In other words any GMHC consisting only of even cycles can be embedded in a Binary Hypercube. In the special case that $W_{D-1} = \dots = W_1 = W_0$, H is a D -dimensional TORUS. Thus Theorem 3.13 is a generalization of the embedding of multidimensional tori in a Binary Hypercube shown in [HILL 84].

Corollary: If $W_i = 2^{n_i}$ for $i = 0, 1, \dots, D-1$ then H is a spanning substructure of Q_n .

3.3.2 Generalized Hypercubes

Theorem 3.14 Let $H = K_{W_{D-1}} \times \dots \times K_{W_1} \times K_{W_0}$ where K_j is the fully connected Structure with j nodes. Let also $W_i \geq 2^{n_i}$ for $i = 0, 1, \dots, D-1$ and $n = n_{D-1} + \dots + n_1 + n_0$. Then

$$Q_n \subset K_{W_{D-1}} \times \dots \times K_{W_1} \times K_{W_0} \quad (3.17)$$

Proof 3.14 Clearly $Q_j \subset K_j$. Since $Q_n = Q_{n_{D-1}} \times \dots \times Q_{n_1} \times Q_{n_0}$ we can obtain 3.17 directly from Lemma 3.2

Q.E.D.

In other words, since $Q_{n-1} \subset Q_n$, there is a large number of embedded binary hypercubes in any GHC. In conjunction with Lemma 3.3 it is possible to partition a CHC in a variety of ways producing different collections of Binary Hypercubes.

3.3.3 Embedded Cycles

Theorem 3.15 Let $H = S_1 \times S_0$. Then H has a Hamiltonian Cycle if both S_1 and S_0 have Hamiltonian Paths and W is even.

Proof 3.15 Without any loss of generality we can number the nodes of S_1 and S_0 so that successive nodes on their Hamiltonian Paths have successive numbers as shown in Figure 3.13. Also assume that W_0 is even and $W_0 > 2$. Then the path:

$$\begin{array}{ll}
(0,0), \dots, (W_1 - 1, 0) & (W_1 - 1, 1), \dots, (W_1 - 1, W_0 - 1), \\
(W_1 - 2, W_0 - 1), \dots, (0, W_0 - 1) & (0, W_0 - 2), \\
(1, W_0 - 2), \dots, (W_1 - 2, W_0 - 2) & (W_1 - 2, W_0 - 3), \\
(W_1 - 3, W_0 - 3), \dots, (0, W_0 - 3) & \dots, \\
\dots & \dots, \\
\dots & (0, 2), \\
(1, 2), \dots, (W_1 - 2, 2) & (W_1 - 2, 1), \\
(W_1 - 3, 1), \dots, (0, 1) & (0, 0)
\end{array}$$

is a Hamiltonian Cycle of H . This is graphically demonstrated in Figure 3.13.

Theorem 3.16 Let $H = S_1 \times S_0$. Then H has a Hamiltonian Cycle if one of S_1, S_0 has a Hamiltonian Cycle and the other has a spanning tree such that none of its nodes has a degree greater than the number of nodes of the other factor Structure.

Proof 3.16 We assume without any loss of generality that S_0 is the Structure that has a Hamiltonian Cycle. From the definition of a Hyperstructure it follows that there will be one copy of S_0 (and therefore one cycle) for each node of S_1 . In order to construct a Hamiltonian Cycle of H we do a depth-first traversal of the spanning tree of S_1 . When visiting a node we traverse a number of edges of the corresponding cycle equal to the length of the cycle (W_0) minus the degree of the node of the spanning tree. Each cycle is traversed in the opposite direction from its parent in the tree. For example consider the Structure shown in Figure 3.14. Assuming the root is to be traversed clockwise, a Hamiltonian cycle of H is:

Figure 3.13 Hamiltonian Cycles in Structure Products

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

where arrows are used to indicate the traversal of an edge of the spanning tree of S_1 .

S_0

S_1

$\times$

Figure 3.11 Hamiltonian Cycles in Structure Products

Chapter 4

Hyperstructure Layout

Although in general the optimal layout cannot be obtained in less than exponential time, a good upper bound of the Area needed for a Hyperstructure can be derived from the layout Areas of its factor Structures. In the following sections a procedure will be developed that will implement such a layout.

In all cases it is assumed that all rectangular objects in a Structure layout (typically processors and memories) have the same height. All interconnections are made through 2-level channel routing. One important characteristic of the layout of Hyperstructures is that the nodes (processors) are considered as isolated islands as opposed to parts of a contiguous row as is the case with gates and other low level constructs. This is in no way restricting since, in general, the complexity of a node and the number of ports required would tend to favor such an implementation.

The basic philosophy of deriving the layout of a Hyperstructure from the layouts of its factor Structures is to form the Cartesian Product of the layouts in two dimensions. The layout of a Hyperstructure is obtained in an iterative manner by multiplying the layouts of two Structures at a time.

To facilitate the derivation of the layout of a Hyperstructure, we define the linear (one dimensional) layout of a Structure. This is discussed in section 4.1. Assuming that the Linear layouts of its factor Structures are known, the layout of a Hyperstructure can be constructed iteratively by replacing each processor in a properly modified and scaled layout of a factor Structure with the layout obtained so far, starting with Structure S_1 as presented in sections 4.2 and 4.3.

Simple polynomial expressions give the proper scaling factor for each step and allow the estimation of the Area without the actual construction of the layout. In the simplest approach the Linear layouts of successive factor Structures are included orthogonal to each other. Section 4.4 presents a technique where by properly selecting the order in which factor structures are incorporated significant savings can be realized. This technique consistently results in layouts comparable to manual layouts reported in the literature.

4.1 Linear Structure Layout

The Linear (or one-dimensional) Layout of a multiprocessor structure is defined as a layout where all rectangular objects are arranged in a single (non-continuous) row. These objects are then interconnected via two routing channels at the top and bottom of the row and, possibly, by connections between neighbouring objects within the body of the row. The row of objects defines the axis of the layout. Although such a layout is almost certain to be suboptimal for each individual structure, it is extremely useful in the derivation of a good layout for Hyperstructures. To facilitate the characterization of the layout of a structure we define the following parameters:

Nw	The width and height of a node (processor) and the height of all rectangular objects.
Mw	The total width of memory(ies) in a structure.
Iw	The sum of the widths of both interconnection channels assuming some connections are made within the body of the row.
INw	The total width of the interconnection assuming no connections are made within the body of the row.
A	The total area occupied by a given layout assuming some connections are made within the body of the row.
AN	The total area occupied by a given layout assuming no connections are made within the body of the row.

The Linear Layouts of some simple structures are shown in Figure 4.1. The Loop and Common Bus structures shown in Figures 4.1.a and 4.1.b respectively fit naturally to a Linear Layout. On the other hand, the Star network shown in Figure 4.2 can be layed out in a number of ways with different costs. For the 5 node Star shown in Figure 4.2.a, the minimum cost Linear Layout is shown in Figure 4.2.c. Similarly, The Layout shown in Figure 4.1.c is the minimum cost Linear Layout for a 4 node shared memory system. In general, for the Linear Layout of any structure S with W processors we have that:

$$A(S) = (W \times Nw + Mw) \times (Nw + Iw) \quad (4.1)$$

Clearly it holds that $INw > Iw$ and

Figure 4.1.a Layout of Loop

Figure 4.1.b Layout of Common Bus

Figure 4.1.c Layout of Common Memory

Figure 4.1.d Layout of Multiport Memory

Figure 4.1.e Minimum Area Layout of Multiport Memory

$$A(S) \leq AN(S) = (W \times Nw + Mw) \times (Nw + INw) \quad (4.2)$$

Considering the structures illustrated in Figures 4.1 and 4.2 and assuming that the width of a point to point link is Lw , the width of a bus is Bw and connection can be made within the body of the row we have that:

Structure	Interconnection	Layout Area
	Width (Iw)	A
Loop	Lw	$4Nw \times (Nw + Lw)$
Common Bus	Bw	$4Nw \times (Nw + Bw)$
Common Memory	Bw	$(4Nw + Mw) \times (Nw + Bw)$
Multiport Memory	Lw	$(4Nw + Mw) \times (Nw + Lw)$
Star	Lw	$5Nw \times (Nw + Lw)$

on the other hand if connection cannot be made within the body of the row

Structure	Interconnection	Layout Area
	Width (INw)	AN
Loop	$2Lw$	$4Nw \times (Nw + 2Lw)$
Common Bus	Bw	$4Nw \times (Nw + Bw)$
Common Memory	Bw	$(4Nw + Mw) \times (Nw + Bw)$
Multiport Memory	$2Lw$	$(4Nw + Mw) \times (Nw + 2Lw)$
Star	$2Lw$	$5Nw \times (Nw + 2Lw)$

In all cases we assume that the ports can be placed on any side of an object (processor or memory) without any area penalty and that any number of ports can be placed along each side.

Figure 4.2.a 5 node Star

Figure 4.2.b Layout of Star

Figure 4.2.c Minimum Area Layout of Star

4.2 Layout of Structure Product

In this section the construction of a Layout for the product of two or more structures will be discussed. The Layout of the product will be derived based on the Linear Layouts of the individual structures. Considering that layouts have to be combined on a plane there are two different ways of combining the layouts of the product of two structures. These consist of either placing structures in a single row thereby creating a Linear Layout or placing them at right angles to each other. These approaches will be analyzed separately in the following sections. In order to facilitate the comparison of structure layouts we define:

WD The width of a given layout.

HT The height of a given layout.

4.2.1 Co-Linear Layout

The Co-Linear Layout of the product (S) of two or more structures

$$S_0, S_1, \dots, S_{k-1}$$

is a Linear Layout of S derived from the Linear Layouts of its factor structures.

Consider the two structures S_0, S_1 shown in Figure 4.3. To construct the Linear Layout of their product $S_0 \times S_1$, we begin by taking the Minimum Area Linear Layout of S_0 and placing W_1 copies of it side by side as shown in Figure 4.4.a. Then we take a Linear Layout of S_1 such that there are no interconnections within the row and the processors of S_1 are placed at distances $W_0 \times W_1$. Figure 4.4.b shows such a

Figure 1.3 Multiprocessor Structure and their Linear Layouts

Layout. Then by placing W_0 copies of this interconnection along the copies of S_0 we obtain the colinear layout of $S = S_0 \times S_1$ shown in Figure 4.5.a. For this layout we have:

$$WD_{S_0 \times S_1} = W_0 W_1 Nw + W_1 M w_0 + W_0 M w_1 \quad (4.3)$$

$$HT_{S_0 \times S_1} = Nw + Iw_0 + W_0 I N w_1 \quad (4.4)$$

and

$$A_{S_0 \times S_1} = (W_0 W_1 Nw + W_1 M w_0 + W_0 M w_1) \times (Nw + Iw_0 + W_0 I N w_1) \quad (4.5)$$

One important property of the Co-Linear layout of the product structures is that it is not reflexive. This can be easily demonstrated by considering the structure $S' = S_1 \times S_0$ whose layout is shown in Figure 4.5.b. For this structure we have:

$$A_{S_1 \times S_0} = (W_0 W_1 Nw + W_1 M w_0 + W_0 M w_1) \times (Nw + Iw_1 + W_1 I N w_0) \quad (4.6)$$

and clearly $A_{S_1 \times S_0} \neq A_{S_0 \times S_1}$. For example assume that $Nw = 1$ and $Lw = 0.2$. Then we have that:

Structure	WD	HT	A
$S_0 \times S_1$	12	2.2	26.4
$S_1 \times S_0$	12	3.4	40.8

Figure 4.1 Construction of Co-Linear Layout

Figure 15 Co-Linear Layout of $S = S_1 + S_2$ and $S' = S_1 + S_2$

Clearly it is desirable that factor structures be incorporated in the Co-Linear layout in such a fashion as to minimize the total Area. The general form of the equation giving the width, height and Area of a layout of the product of k structures $S = S_0 \times S_1 \times \dots \times S_{k-1}$ is:

$$WD = \left(\prod_{i=0}^{k-1} W_i \right) Nw + \sum_{i=0}^{k-1} \left(\prod_{j=0, j \neq i}^{k-1} W_j \right) M w_i \quad (4.7)$$

$$HT = Nw + I w_{i_0} + \sum_{l=1}^{k-1} \left(\prod_{j=0}^{l-1} W_{i_j} \right) I N w_{i_l} \quad (4.8)$$

where $0 \leq i_0, i_1, \dots, i_{k-1} \leq (k-1)$ and $i_j = i_k$ only if $j = k$. Then the total Area is:

$$A = WD \times HT \quad (4.9)$$

It is evident that the height of the Co-Linear layout depends on the order in which the factor structures are combined while the width is a constant for the given collection of structures. Thus it is possible to formulate the minimization of the Area of a Co-Linear layout as:

minimize

$$I w_{i_0} + W_0 (I N w_{i_1} + \dots W_{k-3} (I N w_{i_{k-2}} + W_{k-2} I N w_{i_{k-1}}) \dots)$$

subject to

$$0 \leq i_0 < k$$

...

$$0 \leq i_{k-1} < k$$

$$i_j \neq i_m \text{ if } j \neq m$$

In other words it is necessary to devise some ordering of the structures $S_0, S_1, \dots, S_{k-1}$ so that the height of the layout of S is minimal. To accomplish this we define a set of structure comparison operators ($<, \leq, >, \geq, =$) such that $S_i < S_j$ iff $(Iw_i + W_i INw_j) < (Iw_j + W_j INw_i)$ with the other operators defined similarly. Using these operators it is a simple matter to select some structure S_m such that $S_m \leq S_j$ for all $j \in \{0, 1, \dots, m-1, m+1, \dots, k-1\}$.

Now consider the structures $S'_0, S'_1, \dots, S'_{k-2}$ such that $S'_i = S_i$ if $i < m$ and $S'_i = S_{i+1}$ if $i \geq m$. Also consider the set of structure comparison operators ($<_N, \leq_N, >_N, \geq_N, =_N$) such that $S_i <_N S_j$ iff $(INw_i + W_i INw_j) < (INw_j + W_j INw_i)$ with the other operators defined similarly.

Theorem 4.1 The minimal-Area Co-Linear layout of a set of structures

$$S_0, S_1, \dots, S_{k-1}$$

is the layout obtained from

$$S_{i_0} \times S_{i_1} \times \dots \times S_{i_{k-1}}$$

where $S_{i_0} \leq S_j$ for $j = 1, 2, \dots, k-1$ and $S_{i_j} \leq_N S_{i_m}$ where $j = 1, 2, \dots, k-1$ and $m = j+1, j+2, \dots, k-1$.

Proof 4.1 Follows directly from the definitions of the comparison operators

Q.E.D.

Figure 4.6 CoLinear Layout of $S_0 \times P_2$

Consider the Hyperstructure $H = S_0 \times S_1$. In the special case where S_1 is a simple link connecting two processors (i.e. it is isomorphic to P_2), we can obtain a better layout by making the second copy of S_0 symmetric to the first. In this case the total layout Area will be:

$$A_{S_0 \times S_1} = (2W_0Nw + W_1M_{L_0}) \times (Nw + Iw_0 + (W_0 - 1)INw_1) \quad (4.10)$$

This is demonstrated in in Figure 4.6.

4.2.2 Orthogonal Layout

The orthogonal layout of the product (S) of two structures (S_0, S_1) is obtained by combining the linear layouts of S_0 and S_1 so that their axes are vertical to each

other. It is necessary to modify ("stretch") the linear layouts of both structures so that nodes in the layout of S_0 are placed Iw_1 apart and nodes in the layout of S_1 are placed Iw_0 apart. This creates the space necessary for the interconnections to cross. the orthogonal layout of the two structures shown in Figure 4.3 is given in Figure 4.7. Assuming that all horizontal lines are located on one layer whereas all vertical lines are located on the other layer, no conflict will arise from the combination of the layouts. The parameters of the orthogonal layout of two structures can be calculated as:

$$WD_{S_0 \times S_1} = W_0(Nw + Iw_1) + Mw_0 \quad (4.11)$$

$$HT_{S_0 \times S_1} = W_1(Nw + Iw_0) + Mw_1 \quad (4.12)$$

$$AS_{S_0 \times S_1} = (W_0(Nw + Iw_1) + Mw_0) \times (W_1(Nw + Iw_0) + Mw_1) \quad (4.13)$$

Clearly the orthogonal layout of the product of two structures is reflexive. Since the layout is always two dimensional the orthogonal layout only makes sense for a pair of structures. If another structure S_2 was to be incorporated in $S_0 \times S_1$ then it would be combined with one of the structures forming the orthogonal layout (say S_0) into a Co-Linear layout and the orthogonal layout would be properly scaled to accommodate the extra interconnections. This is equivalent to forming the Co-Linear layout of $S' = S_0 \times S_2$ and then the orthogonal layout of $S' \times S_1$.

Figure 4.7 Orthogonal layout of $S_0 \dots S_7$

4.3 Estimation of Hyperstructure Layout Area

A layout of a Hyperstructure $H = S_0 \times S_1 \times \dots \times S_{k-1}$ can be obtained by partitioning the factor structures of H in two groups, forming the Co-Linear layouts of the product of the structures in each group and then forming the orthogonal layout of the product of the two groups. To describe the partition a function is defined:

$$Ld : \{0, 1, \dots, n-1\} \rightarrow \{0, 1\} \quad (4.14)$$

Thus we have:

$$H = H_0 \times H_1$$

where

$$H_0 = \prod_{i=0, Ld(i)=0}^{i=n-1} S_i$$

$$H_1 = \prod_{i=0, Ld(i)=1}^{i=n-1} S_i$$

Without any loss of generality we can renumber the factor structures of H so that $Ld(i) = 0$ for $i = 0, 1, \dots, k-1$ and $Ld(i) = 1$ for $i = k, k+1, \dots, D-1$. In other words, structures $S_0 \dots S_{k-1}$ are all layed out colinearly and S_i is included before S_{i+1} for $i = 0, 1, \dots, k-2$. Similarly structures $S_k \dots S_{D-1}$ are incorporated in a Co-Linear layout of H_1 . The parameters of the layouts of H_0 and H_1 can then be calculated as:

$$WD_0 = \left(\prod_{i=0}^{k-1} W_i \right) Nw + \sum_{i=0}^{k-1} \left(\prod_{j=0, j \neq i}^{k-1} W_j \right) Mw_i \quad (4.15)$$

$$HT_0 = Nw + Iw_0 + \sum_{i=1}^{k-1} \left(\prod_{j=0}^{i-1} W_j \right) INw_i \quad (4.16)$$

$$WD_1 = \left(\prod_{i=k}^{D-1} W_i \right) Nw + \sum_{i=k}^{D-1} \left(\prod_{j=k, j \neq i}^{D-1} W_j \right) Mw_i \quad (4.17)$$

$$HT_1 = Nw + Iw_k + \sum_{i=k+1}^{D-1} \left(\prod_{j=k}^{i-1} W_j \right) INw_i \quad (4.18)$$

The layout of the Hyperstructure H can then be constructed as the orthogonal layout of the product of H_0 and H_1 . Thus the parameters of the layout of H are:

$$WD = \left(\prod_{i=0}^{k-1} W_i \right) (Nw + Iw_k + \sum_{i=k+1}^{D-1} \left(\prod_{j=k}^{i-1} W_j \right) INw_i) + \sum_{i=0}^{k-1} \left(\prod_{j=0, j \neq i}^{k-1} W_j \right) Mw_i \quad (4.19)$$

$$HT = \left(\prod_{i=k}^{D-1} W_i \right) (Nw + Iw_0 + \sum_{i=1}^{k-1} \left(\prod_{j=0}^{i-1} W_j \right) INw_i) + \sum_{i=k}^{D-1} \left(\prod_{j=k, j \neq i}^{D-1} W_j \right) Mw_i \quad (4.20)$$

$$A = WD \times HT \quad (4.21)$$

For example consider the 3^4 Spanning Bus Hypercube shown in Figure 4.8. By iteratively incorporating the layout of its factor as shown in Figure 4.9 a layout for the complete Hyperstructure is constructed. This is shown in Figure 4.10.

Figure 4.8 3^4 Spanning Bus Hypercube

Figure 4.9 Partial Layouts of 3^4 Spanning Bus Hypercube

Figure 4.10 Layout of 3⁴ Spanning Bus Hypercube

4.4 Minimization of Hyperstructure Layout Area

From the construction of the layout of a Hyperstructure it is clear that the total Area will depend on the partitioning of the factor structures. Although there is no single algorithm for the selection of the optimum partition, some simple heuristics have been developed that yield good results.

Before examining these in detail the computational cost of selecting an optimum layout via a brute force approach will be analyzed. There are a total of 2^D different functions $L_d(i)$. By taking advantage of the symmetry of the partition, only half of these need be considered. For each function considered it will be necessary to order the structures in each partition. This ordering will require at most $2D^2$ comparisons. Thus the number of operations needed for the derivation of an optimal layout is at most:

$$D^2 2^D \leq W \log^2(W)$$

where W is the number of processors in the Hyperstructure.

Clearly the brute force approach is tractable for any real system. Furthermore, the worst case will only arise if all factor structures are distinct. If some structures are used more than once the total number of cases that must be considered is drastically reduced. For example for the 3^4 SBH examined in section 4.3, only 3 of the 16 possible partitions need to be examined.

One heuristic that produces good results is the following:

Select the two "minimum" prime factor structures with respect to the operators $\{<, \leq, >, \geq, =\}$ as defined in section 4.2.1. Form the orthogonal layout of

the product of these two structures. Order the remaining structures in ascending order with respect to the operators $\{<_N, \leq_N, >_N, \geq_N, =_N\}$. Iteratively incorporate these structures into the layout. The direction along which each structure is incorporated is selected by calculating the Areas and selecting the best of the two. This process is illustrated in Figure 4.9 for the 3^4 SBH. To illustrate the benefits of the process consider the structure $C_3 \times Q_3$ shown in Figure 4.11. The "natural" layout of this structure is shown in Figure 4.12 and is clearly inferior to the layout arrived at through this procedure which is shown in Figure 4.13.

OF / DE

2

Figure 1.11 $C_4 = Q_4$

Figure 4.12 Layout of Q_3

Figure 1.13 Optimal Layout of $C_2 \times P_2 \times P_2 \times P_2$

Chapter 5

Reduction of Hyperstructure Interconnection

Despite their many advantages, Hyperstructures share one of the most criticized drawbacks of hypercubes. Namely, the fact that interconnection costs increase as $W \log(W)$ rather than as a linear function of W . Although this is desirable under assumption 1, it is not desirable for computer networks in general.

Different techniques have been proposed in the literature. In the following sections a general approach to this problem will be presented. It will be shown that the two main approaches proposed in the literature can be unified and treated as special cases of the technique presented in this work.

Section 5.1 presents a brief overview of the two existing techniques and demonstrates how one can be obtained as a combination of the other and the results presented in chapter 3 of this work. In section 5.2 the Interconnection Adjacency Matrix (IAM) will be introduced and defined. The IAM will form the basis of the approach for the controlled reduction of the interconnection cost of Hyperstructures.

Section 5.3 will present a formal approach to the pruning of interconnection networks from a Hyperstructure. The effects of pruning on the parameters of a Hyperstructure will be discussed in section 5.4. Section 5.5 will deal with the layout of reduced Hyperstructures. The layout algorithms developed in chapter 4 will be extended to incorporate the layout of reduced Hyperstructures. In all cases examples will be presented demonstrating the performance of the proposed techniques as compared to those in the literature.

5.1 Overview of Present Techniques

In order to reduce the interconnection cost of hypercubes, some new structures have been presented in the literature. This is accomplished by modifying known structures in a controlled way. Two main structures have been proposed in the literature which can be generalized in the following techniques:

1. Starting with a D dimensional binary hypercube, each node is replaced by a cycle of at least D processors. Each of the first D processors of the cycle is then connected to one of the links that connected to the original node. The connections are arranged so that processor i is connected to the link along dimension i of the hypercube [PREP 80].
2. Starting with a D dimensional Spanning Bus Hypercube, some of the busses spanning dimensions $1 \dots D - 1$ are removed so that each node $x_0 x_1 \dots x_{D-1}$ remains connected to a bus spanning dimension 0 and a bus spanning dimension $(x_0 \text{ mod } (W_0 - 1)) + 1$ [WITT 81].

In the remainder of this section it will be shown that these two techniques can be unified, by taking advantage of the operations defined in chapter 3. To illustrate this point consider the Hyperstructure $H = C_3 \times Q_3$ shown in Figure 5.1. Now consider a generalization of the second technique in which entire interconnection subnetworks are pruned. By applying this to H we obtain the Hyperstructure shown in Figure 5.2 which is isomorphic to the 3 dimensional Cube Connected Cycles shown in Figure 5.3.

5.2 Interconnection Adjacency Matrix (IAM)

Clearly it is possible to construct a number of algorithms for the pruning of a Hyperstructure in a controlled fashion. In order to facilitate the development of such algorithms while ensuring that the resulting structure maintains proper connectivity, the Interconnection Adjacency Matrix (IAM) is defined.

The IAM of a Hyperstructure is conceptually similar to the incidence matrix of a graph. It consists of W rows and $WI = WI_0 + WI_1 + \dots + WI_{D-1}$ columns corresponding to the nodes and interconnection subnetworks of the Hyperstructure, respectively. Each position of the matrix contains either a 0 or a 1 indicating that the node represented by the row number is or is not connected to the interconnection subnetwork represented by the column number. Both row and column numbers are given in the mixed radix system presented in section 3.2.

Almost invariably the IAM of a Hyperstructure will be sparse. Also the size of the IAM will increase as $O(W^2 \log(W))$ where W is the number of nodes (processors) whereas the number of non-zero entries will only increase at most as $O(W \log(W))$.

Figure 5.1 Hyperstructure $H = C = Q$.

Figure 5.2 Reduced Hyperstructure

Figure 5.3 3-Dimensional Cube Connected Cycles

In other words, the IAMs of larger Hyperstructures will be more sparse. For example the IAM of the Hyperstructure $C_3 \times Q_3$ shown in Figure 5.1 is:

```

0000:111000000111111000000111111
001100110001110001111110000011111
010101011111111111111111111111111
-----012012012012012012012012012012012
0000 1000000010000000000100000000000100000000000
0001 100000000100000000000100000000000100000000000
0002 1000000000100000000000100000000000100000000000
0010 0100000010000000000000100000000000100000000000
0011 0100000001000000000000010000000000010000000000
0012 0100000000100000000000001000000000010000000000
0100 00100000000100000000010000000000000000000000
0101 00100000000010000000001000000000000000000000
0102 00100000000001000000000100000000000000000000
0110 00010000000010000000000010000000000000000000
0111 00010000000001000000000000100000000000000000
0112 00010000000000010000000000001000000000000000
1000 0000100000000001000000000000100000100000000000
1001 0000100000000000100000000000010000010000000000
1002 0000100000000000010000000000001000001000000000
1010 0000010000000001000000000000001000001000000000
1011 0000010000000000100000000000000100000100000000
1012 0000010000000000001000000000000010000010000000
1100 00000010000000000010000000010000000000000000
1101 00000010000000000001000000001000000000000000
1102 00000010000000000000100000000010000000000000
1110 00000001000000000010000000000100000000000000
1111 00000001000000000001000000000001000000000000
1112 00000001000000000000100000000000100000000000

```

The IAM can become more manageable if a new symbol $\star$ is introduced to the mixed radix system described in section 3.2. The $\star$ is conceptually similar to the symbol X in Boolean Algebra except that the range of values it represents will depend on the radix of the position it occupies. For example in the case of the Hyperstructure of Figure 5.1, $000\star$ represents the values 0000, 0001 and 0002 while $0\star00$ represents the values 0000 and 0100. From the definition of a Hyperstructure it follows that node $X = x_0x_1 \dots x_{D-1}$ is connected to interconnection subnetwork $Y = y_0 \dots y_{i-1} \star y_{i+1} \dots y_{D-1}$ iff $x_0 = y_0, \dots, x_{i-1} = y_{i-1}$ and $x_{i+1} = y_{i+1}, \dots, x_{D-1} = y_{D-1}$.

Taking advantage of this fact it is possible represent the IAM of the Hyperstructure $C_3 \times Q_3$ in a compressed form as:

$$\begin{array}{cccc}
 \star & \star & \star & \psi \\
 \star & \star & \psi & \star \\
 \star & \psi & \star & \star \\
 \psi & \star & \star & \star \\
 \star \star \star \star & 1 & 1 & 1 & 1
 \end{array}$$

As different interconnection networks are removed the IAM will change to reflect the fact. For example the IAM of the 3 dimensional cube connected cycles shown in Figure 5.3 is:

$$\begin{array}{cccc}
 \star & \star & \star & \psi \\
 \star & \star & \psi & \star \\
 \star & \psi & \star & \star \\
 \psi & 0 & 1 & 2 \\
 \star \star \star \star & 1 & 1 & 1 & 1
 \end{array}$$

Notice that the interconnection subnetworks $\star \star \psi 1$, $\star \star \psi 2$, $\star \psi 1 \star$, $\star \psi 2 \star$, $\psi 1 \star \star$ and $\psi 2 \star \star$ are disconnected and have been removed from the IAM. As expected this is also the IAM of the 3^4 Dual Bus Hypercube. Thus the IAM provides a concise picture of the interconnection network of a Hyperstructure independent of the exact form of its factor structures.

5.3 Interconnection Pruning

In this section the reduction of the interconnection cost of a Hyperstructure is analyzed. For reasons of simplicity and tractability, the reduction is restricted to the removal of entire interconnection subnetworks at a time. Although no technique is available for the reduction of interconnection in a general way, the results presented in chapter 3 and the IAM of a Hyperstructure, described in section 5.2, can be used to provide immediate feedback of the effect the removal of one or more interconnection subnetworks will have on the parameters of a Hyperstructure. Thus the diameter of the reduced Hyperstructure along with the distances of all its nodes can be easily obtained by the use of well known graph and matrix properties.

Since Hyperstructures are pruned by removing entire interconnection subnetworks only, it follows that the most compact form of the IAM will always be a single row. However, less compact forms can be utilized to better demonstrate the interconnection. For example the IAM of the 3 dimensional cube connected cycles could be written as:

$$\begin{array}{cccc}
 \star & \star & \star & \psi \\
 \star & \star & \psi & \star \\
 \star & \psi & \star & \star \\
 \psi & 0 & 1 & 2 \\
 \\
 \star \star \star 0 & 1 & 1 & 0 & 0 \\
 \star \star \star 1 & 1 & 0 & 1 & 0 \\
 \star \star \star 2 & 1 & 0 & 0 & 1
 \end{array}$$

which demonstrates the fact that each node is connected to exactly two interconnection subnetworks and all nodes are connected to an interconnection subnetwork along dimension 0.

In order to demonstrate the concept, some new Hyperstructures will be defined which will be shown to be superior to corresponding Hyperstructures defined in the literature. The Triple Bus Hypercube (TBH) is defined similar to the Dual Bus Hypercube [WITT 81]. Starting with a Spanning Bus Hypercube, all busses spanning dimensions 0 and 1 are maintained. The busses spanning the remaining dimensions are pruned so that each node is connected to exactly three busses. Busses are retained so that node $x_{D-1} \dots x_1 x_0$ is connected to a bus spanning dimension $(x_0 \bmod (W_0 - 2)) + 2$. The IAM of the 3^4 TBH shown in Figure 5.4 is:

	*	*	*	*	ψ
	*	*	ψ	ψ	*
	*	ψ	*	*	*
	ψ	*	0	2	1
***0	1	1	1	0	0
***1	1	1	0	0	1
***2	1	1	0	1	0

Clearly the TBH has characteristics similar to the DBH. However, as can be seen in Figure 5.4 it also shares one of the most serious drawbacks of the DBH, namely the fact that the number of busses spanning each dimension may be unequal. In order to remove this drawback while maintaining the advantages of the TBH, the Balanced Triple Bus Hypercube (BTBH) is defined. In a BTBH, each node

Figure 5.4 3⁴ Triple Bus Hypercube

is connected to 3 busses. In particular, node $x_{D-1} \dots x_1 x_0$ is connected to a bus spanning dimension 0, a bus spanning dimension $k = (x_0 \bmod (W_0 - 1)) + 1$ and a bus spanning one of the other dimensions, depending on the value of x_k . The IAM of the 4 dimensional BTBH shown in Figure 5.5 is:

	*	*	*	*	1	*	*	0	2	ψ	ψ	ψ	ψ
	*	*	0	2	*	ψ	ψ	ψ	ψ	*	*	*	1
	*	ψ	ψ	ψ	ψ	*	1	*	*	*	0	2	*
	ψ	0	1	1	2	1	2	0	0	2	0	0	1
***0	1	1	0	0	0	0	0	1	1	0	1	1	0
***1	1	0	1	1	0	1	0	0	0	0	0	0	1
***2	1	0	0	0	1	0	1	0	0	1	0	0	0

A number of variations of the BTBH can be easily obtained that would best meet different requirements. For example, a modified TBH suitable for operation under assumption 3, might be one where each node $x_{D-1} \dots x_1 x_0$ is connected to a bus spanning dimension 0, a bus spanning dimension 1 and a bus spanning dimension $((x_0 + x_1) \bmod (W_1 - 2)) + 2$. Clearly, the combination of Hyperstructure with interconnection pruning provides a rich source of architectures.

5.4 Layout of Reduced Hyperstructures

This section will deal with the Layout of Reduced Hyperstructures. The general approach developed in chapter 4 will be used for the construction of layouts. The techniques developed for complete Hyperstructures will be used as a basis for the de-

Figure 5.5 3^3 Balanced Triple Bus Hypercube

velopment of similar techniques for Reduced Hyperstructures. The effect of pruning to the layout Area of a Hyperstructure will be discussed. Closed form expressions will be presented which will make possible the rapid prediction of a limit of the layout Area of a Reduced Hyperstructure based on the Area of the corresponding complete Hyperstructure and the IAM. It will be shown that layouts obtained through this technique are better than manually constructed layouts presented in the literature.

The simplest approach to a layout for a reduced Hyperstructure is to construct the minimum layout of the complete Hyperstructure as described in chapter 4, remove all pruned interconnection subnetworks and then compact the resulting layout. Thus the layouts of a 3^4 DBH (shown in Figure 5.6) and TBH (Figure 5.7) can be immediately obtained from the layout of a 3^4 SBH shown in Figure 4.10. A good estimate of the Area of the layout of the reduced Hyperstructure can be obtained by calculating the percentage of each interconnection subnetwork retained and using these numbers to scale the total area occupied by the copies of each interconnection subnetwork. This approach produced exact results for the TBH and an excellent approximation for the DBH.

Although adequate as a first approximation this technique does not take into account the fact that the optimum order in which factor structures are incorporated into the layout may have been altered by the pruning. To take this into consideration the algorithm presented in chapter 4 must be altered so that the ordering of the factor structures takes into consideration the effects of the pruning. To accomplish this it is sufficient to redefine the sets of comparison operators ($<$, $\leq$, $>$, $\geq$, $=$) and ($<_N$, $\leq_N$, $>_N$, $\geq_N$, $=_N$) so that $S_i < S_j$ iff $a_j(Iw_i + W_iINw_j) < a_i(Iw_j + W_jINw_i)$, where a_i is the percentage of copies of interconnection subnetwork i that are retained. Using this modified algorithm, a minimum layout of the reduced Hyperstructure can

Figure 5.6 Layout of 3¹ Dual Bus Hypercube

Figure 5.7 Layout of 3^4 Triple Bus Hypercube

Figure 5.8 Minimum Layout of 3 dimensional CCC

be constructed in a fashion similar to the construction of the layout of the complete Hyperstructure. The minimum layout of the 3 dimensional CCC is shown in Figure 5.8.

5.4.1 Comparison to Previous Results

In this section the Layouts obtained using the method described in section 5.4 will be compared to layouts discussed in the literature for reduced Hyperstructures.

Of the two works describing reduced Hypercubes, only [PREP 81] addresses the layout of the proposed structure. A very regular approach is presented which yields a layout of total area

$$A(\text{CCC}_n) = (2^n - 1) \times 2^{n+1}$$

for Cube Connected Cycles of any dimension. Since it was assumed that each node would be implemented as a single integrated circuit, the width of a processor and the width of a link were taken to be the same ($Nw = Lw = 1$). Under this assumption, the regular layout for a 3 dimensional CCC has a total area of 112. On the other hand the layout derived for CCC_3 , shown in Figure 5.8, has a total area of 64, for a 42.8% reduction. For comparison the regular and minimum layouts of the 4 dimensional CCC are shown in Figures 5.9 and 5.10 respectively. The regular layout occupies an area of 480 whereas the area of the derived layout is 256, resulting in a 46.7% reduction.

It should be noted that [PREP 81] also presented a more compact layout for CCC_4 with a total Area of 416. However, due to the fact that no algorithm was available for obtaining such a layout for the general case, the improvement was not deemed worthwhile. It is clear that significant improvements in the layout Area of reduced Hyperstructures can be achieved by employing the technique described in section 5.4. Even compared against the improved layout of the 4 dimensional CCC, the technique presented in this work achieves a 38.5% reduction in Area.

In the case of a VLSI circuit, with more than one processor per IC, the ratio Lw/Nw will be significantly less than one. The following tables show a comparison of the results obtained by regular and derived layouts for 3 and 4 dimensional CCCs.

Figure 3.9 Regular Layout of 4-dimensional CCC

Figure 5.10 Minimum Layout of 4-dimensional CCC

	Regular	Minimum	Regular	Minimum
Lw/Nw	CCC ₃	CCC ₃	CCC ₄	CCC ₄
1.0	16.0 × 7.0	8.0 × 8.0	32.0 × 15.0	16.0 × 16.0
0.9	15.2 × 6.6	7.8 × 7.6	30.4 × 13.9	15.2 × 15.2
0.8	14.4 × 6.2	7.6 × 7.2	28.8 × 12.8	14.4 × 14.4
0.7	13.6 × 5.8	7.4 × 6.8	27.2 × 11.7	13.6 × 13.6
0.6	12.8 × 5.4	7.2 × 6.4	25.6 × 10.6	12.8 × 12.8
0.5	12.0 × 5.0	7.0 × 6.0	24.0 × 9.5	12.0 × 12.0
0.4	11.2 × 4.6	6.8 × 5.6	22.4 × 8.4	11.2 × 11.2
0.3	10.4 × 4.2	6.6 × 5.2	20.8 × 7.3	10.4 × 10.4
0.2	9.6 × 3.8	6.4 × 4.8	19.2 × 6.2	9.6 × 9.6
0.1	8.8 × 3.4	6.2 × 4.4	17.6 × 5.1	8.8 × 8.8

	Regular	Minimum	Regular	Minimum
Lw/Nw	CCC ₃	CCC ₃	CCC ₄	CCC ₄
1.0	112.0	64.0	480.0	256.0
0.9	100.3	59.3	422.6	231.0
0.8	89.3	54.7	368.6	207.4
0.7	78.9	50.3	318.2	185.0
0.6	69.1	46.1	271.4	163.8
0.5	60.0	42.0	228.0	144.0
0.4	51.5	38.1	188.2	125.4
0.3	43.7	34.3	151.8	108.2
0.2	36.5	30.7	119.0	92.2
0.1	29.9	27.3	89.8	77.4

As can be observed from the above tables, layouts derived with the techniques

presented in this work are always superior to those that can be obtained from the regular layout described in [PREP 81]. The reduction of the difference between the two approaches as link widths become smaller indicates that the derived layout also provides reduced link length. Also the derived layouts remain close to being square whereas the regular layouts become increasingly elongated as the link width decreases. All these are desirable characteristics of a layout.

Chapter 6

Summary and Conclusions

This work dealt with the problem of the evaluation and generation of hypercube-type architectures. A theoretical framework was developed that makes possible the rapid evaluation of the characteristics of any hypercube-type multiprocessor. The characteristics employed for the evaluation of different architectures (Hyperstructures) include the diameter, average message path length, average message density and fault tolerance. Closed form expressions were presented for the rapid evaluation of different architectures. Almost all configurations proposed in the literature are analyzable as special cases of the proposed scheme. Furthermore, the proposed scheme suggests a simple way of generating a large number of new architectures. Some such architectures were introduced and shown to be superior to architectures presented in the literature.

Special attention was given to Hyperstructures whose interconnection consists entirely of point to point links. A number of containment and partitioning properties of the binary hypercube were generalized to a large class of Hyperstructures. Also the presence of cycles in Hyperstructures was investigated and two sufficient conditions

for the existence of a Hamiltonian cycle in a Hyperstructure were introduced.

The problem of generating a good layout for a Hyperstructure was also addressed. Algorithms are presented that make possible the generation of near-optimal layouts for any Hyperstructure. The computational requirements of these algorithms were shown to be $O(W \log^3(W))$ in the worst case with most practical cases requiring only a fraction of this bound. The layouts generated proved equal or superior to layouts resulting from specific algorithms reported in the literature.

The reduction of the interconnection cost of a Hyperstructure is also addressed. An approach is presented that makes possible the reduction of the interconnection cost in a controlled fashion. It is shown that the two approaches presented in the literature can be unified under the framework presented in this work. The layout algorithms were extended to reduced Hyperstructures and were shown to yield significant improvements over known layouts manually constructed for specific configurations.

The main conclusion of this paper is that the proposed view of multiprocessor architectures not only unifies the analysis of different hypercube-type architectures, but also offers a source of promising new topologies. Contrary to previous results, this work makes it possible to incorporate known architectures and configurations into a much larger structure while maintaining all the advantages of the original architecture(s). The simple generation of near optimal layouts for any Hyperstructure is but one of the applications of this work.

6.1 Recommendations for Further Work

The most obvious continuation of this work would be the development of some automated tools for the rapid analysis of competing Hyperstructures. The development process coupled with some use of the results for the derivation of configurations suitable for specific problems would result in valuable insight into the strengths of the framework and suggest ways for its improvement.

Of particular interest would be the further investigation of the reduction of Hyperstructure interconnection. Even if the development of some theoretical approach to the reduction problem proves impractical, it seems that a number of useful heuristics can be developed and employed to good advantage.

The optimality of the layouts resulting from the method described in this work should also be investigated. It is possible that more special cases exist which could result in further savings in layout area. The derivation of layouts for reduced Hyperstructures should also be investigated further. It seems likely that more efficient algorithms and/or heuristics can be developed that would better take advantage of the specific way in which the interconnection of a Hyperstructure is reduced to yield optimal layouts.

References

- [SABI 60] Gert Sabidussi, "Graph Multiplication", Mathematische Zeitschrift, 1960.
- [STON 71] Harold S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE transactions on Computers, February 1971.
- [PEAS 77] Marshall C. Pease III, "The Indirect Binary n -cube Microprocessor Array", IEEE transactions on Computers, May 1977.
- [WU 79] Shyue B. Wu, Ming T Liu, "A Generalized Cluster Structure for Large Multi-microcomputer Systems", Proc. 1979 Int. Conf. on Parallel Processing, August 1979.
- [DIAS 81] Daniel M. Dias, Robert J. Jump, "Analysis and Simulation of Buffered Delta Networks", IEEE transactions on Computers, April 1981.
- [GOOD 81] James R. Goodman, Carlo H. Sequin, "Hypertree: A Multiprocessor Interconnection Topology", IEEE transactions on Computers, December 1981.

- [HORO 81] Ellis Horowitz, Alessandro Zorat, "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI", IEEE transactions on Computers, April 1981.
- [MART 81] A. J. Martin, "The TORUS: An exercise in constructing a Processing Surface", Proceedings of the VLSI Conference, December 1981.
- [PREP 81] Franco P. Preparata, Jean Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation", Communications of ACM, May 1981.
- [WITT 81] Larry D. Wittie, "Communication Structures for Large Networks of Microcomputers", IEEE Transactions on Computers, April 1981.
- [WUFE 81] Chian-Lin Wu, Tse-Yun Feng, "The Universality of the Shuffle Exchange", IEEE Transactions on Computers, May 1981.
- [WULI 81] Shyue B. Wu, Ming T. Liu, "A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems", IEEE Transactions on Computers, April 1981.
- [LANG 82] T. Lang, "Bandwidth of Crossbar and Multibus connections for Multiprocessors", IEEE Transactions on Computers, December 1982.
- [MARS 82] Ajmone M. Marshan, M. Gerla, "Markov Models for Multiple Bus Multiprocessors", IEEE Transactions on Computers, March 1982.

- [PRAD 82] Dhiraj K. Pradhan, Sudhakar M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems", IEEE Transactions on Computers, September 1982.
- [BHUY 84] Laxmi N. Bhuyan, "A Combinatorial Analysis of Multibus Multiprocessors", Proc. Int. Conf. on Parallel Processing, August 1984.
- [BHUY 84a] Laxmi N. Bhuyan, Dharma P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network", IEEE Transactions on Computers, April 1984.
- [IRAN 84] K. B. Irani, J. H. Onyuksel, "A Closed Form Solution for the Performance Analysis of Multiple-Bus Multiprocessor Systems", IEEE Transactions on Computers, November 1984.
- [STAN 84] John A. Stankovic, "A Perspective on Distributed Computer Systems", IEEE Transactions on Computers, December 1984.
- [DAS 85] Chita R. Das, Laxmi N. Bhuyan, "Bandwidth Availability of Multiple-Bus Multiprocessors", IEEE Transactions on Computers, October 1985.
- [DIMO 85] Nikitas J. Dimopoulos, "On the Structure of the Homogeneous Multiprocessor", February 1985.
- [FOX 85] Geoffrey C. Fox, "Summary of Architectural Lessons from the Caltech/JPL Concurrent Computation Project", September 1985.
- [HILL 85] W. D. Hillis, "The connection Machine", MIT Press, 1985.

- [MUDG 85] Trevor N. Mudge, Humoud B. Al-Sadoun, "A Semi-Markov Model for the Performance of Multiple-Bus Systems", IEEE Transactions on Computers, October 1985.
- [PRAD 85] Dhiraj K. Pradhan, "Fault-Tolerant Multiprocessor Link and Bus Network Architectures", IEEE Transactions on Computers, January 1985.
- [PRAD 85a] Dhiraj K. Pradhan, "Dynamically Restructurable Fault-Tolerant Processor Network Architectures", IEEE Transactions on Computers, May 1985.
- [SEIT 85] Charles L. Seitz, "The Cosmic Cube", Communications of ACM, January 1985.
- [WAGN 85] Alan Wagner, "Embedding Binary Trees in a Hypercube", Technical Report CSRI-175, October 1985.
- [AGRA 86] Dharma P. Agrawal, Virendra-K. Janakiram, Girish C Pathak, "Evaluating the Performance of Multicomputer Configurations", IEEE Computer, May 1986. IEEE Computer
- [ADAM 87] George B. Adams III, Dharma P. Agrawal, Howard Jay Siegel, "Fault-Tolerant Multistage Interconnection Networks", IEEE Computer, June 1987.
- [BHUY 87] Laxmi N. Bhuyan, "Interconnection Networks for Parallel and Distributed Processing", IEEE Computer, June 1987.

- [MUDG 87] Trevor N. Mudge, John P. Hayes, Donald C. Winsor "Multiple Bus Architectures", IEEE Computer, June 1987
- [ATHA 88] William C. Athas, Charles L. Seitz, "Multicomputers: Message Passing Concurrent Computers", IEEE Computer, August 1988.
- [TUCK 88] Lewis W. Tucker, George G. Roberstson, "Architecture and Applications of the Connection Machine", IEEE Computer, August 1988.
- [SAAD 88] Youcef Saad, Martin H. Schultz, "Topological Properties of Hypercubes", IEEE transactions on Computers, July 1988.
- [DOWD 88] Patrick W. Dowd, Kamal Jabbour, "Spanning Multiaccess Channel Hypercube Computer Interconnection", IEEE transactions on Computers, September 1988.
- [AKER 89] Sheldon B. Akers, Balakrishnan Krishnamurthy "A Group-Theoretic Model for Symmetric Interconnection Networks", IEEE transactions on Computers, April 1989.
- [SAMA 89] Maheswara R. Samatham, Dhiraj K. Pradhan "The De Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI", IEEE transactions on Computers, April 1989.
- [MASU 89] Hiroshi Masuyama, Tetsuo Ichimori, "Tolerance of Double-Loop Computer Networks to Multinode Failures", IEEE transactions on Computers, May 1989.

Appendix A

List of Symbols

- BTBH** • Balanced Tripple Bus Hypercube
- C** • an ordered set of the connection functions of the factor Structures.
- CCC** • Cube Connected Cycles
- DBH** • Dual Bus Hypercube
- $DIAM$** • the maximum number of nodes a message may have to go through to reach its destination.
- $DIAM_i$** • the diameter ($DIAM$) of factor Structure i of a Hyperstructure.
- $D(X_1, X_2)$** • The maximum number of hops a message has to do to reach node X_2 starting from node X_1 .
- $D_i(x_1, x_2)$** • The maximum number of hops a message has to do to reach node x_2 starting from node x_1 of factor Structure i of a Hyperstructure.

- GHC** • Generalized Hypercube
- GMHC** • Generalized Mesh Hypercube
- I*** • an ordered set of the interconnection networks of the factor Structures.
- IAM** • Interconnection Adjacency Matrix
- IMD*** • the maximum message density the interconnection network can handle.
- IMD_k*** • the maximum message density the interconnection network can handle through ports of type *k*.
- MPL*** • The average number of hops a message has to do to reach its destination, assuming uniform communication requirements among all processors in the Structure.
- MPL_i*** • The average message path length of factor Structure *i* of a Hyperstructure.
- N*** • the set of nodes of a Hyperstructure *H*.
- NNP_k^j*** • the number of ports of type *k* on node *j*.
- NMD*** • the maximum message density a node can handle, excluding internal messages.
- NiMD*** • the maximum message density that can be passed between the main and front-end processor of a node.
- NIP_k*** • the number of ports of type *k* of an interconnection network.

- PMD_k** • The maximum message density (number of messages per time unit) a port of type k can handle.
- Q_n** • n -dimensional Cube
- SBH** • Spanning Bus Hypercube
- TBH** • Tripple Bus Hypercube
- W** • an ordered set of the numbers of nodes W_i of the factor Structures of H .
- W** • the number of nodes of a Multiprocessor (Hyper)Structure.
- W_i** • the number of nodes of factor Structure i of a Hyperstructure.

Appendix B

Graph and Group Terms and Operations

The purpose of this Appendix is to provide the definitions of the various Graph-theoretic and Group-theoretic terms and operations relevant to this work.

Definition 4 A Graph G is a pair consisting of a finite, nonempty set V (the set of vertices of G), together with a (possibly empty) set E (the set of edges of G) of two-element subsets of distinct elements of V .

If the vertex set is represented as a collection of points then each edge can be represented as a line joining two distinct points. Generally, the vertex and edge sets of a graph G are represented as $V(G)$ and $E(G)$ respectively. If the vertices of a graph are distinguished from each other by names the graph is called a labeled graph. If two vertices are joined by an edge they are called adjacent. A graph is complete if every pair of its vertices is adjacent. A complete graph of order p is denoted as K_p . The number of vertices (the cardinality of $V(G_j)$) is called the order W_j of the graph. The number of edges connected to a vertex is called the degree of the vertex. A graph is r -regular if all its vertices have the same degree r .

A graph S is a subgraph of a graph G if $V(S) \subseteq V(G)$ and $E(S) \subset E(G)$. This relationship is denoted as $S \subset G$. If $V(S) = V(G)$ then S is a spanning subgraph of G . If $E(S)$ contains all the edges $[uv] \in E(G)$ such that $u \in V(S)$ and $v \in V(S)$, then S is called an induced subgraph of G . The complement $\overline{G}$ of a graph G is the graph with vertex set $V(G)$ such that two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G . A graph G is n -partite, $n \geq 1$, if it is possible to partition $V(G)$ into n subsets $V_1, V_2, \dots, V_n$ called partite sets such that every element of $E(G)$ joins a vertex of V_i to a vertex of $V_j, i \neq j$. A bipartite graph is a 2-partite graph.

A path of a graph G joining $u, v \in V(G)$ is a subgraph P of G such that $V(P) = \{x_i\}$ for $0 \leq i < n, n \geq 1, x_0 = u, x_{n-1} = v$, all x_i are distinct and $E(P) = [x_i, x_{i+1}]$ for $0 \leq i < n-1$. A path of order n is denoted as P_n . A graph G is connected if for any two of its vertices are joined by at least one path. The distance of two vertices of a graph is defined as $d(u, v) = \min |P|$, the minimum taken over all paths of G joining u and v . The eccentricity $e(v)$ of a vertex v of a connected graph G is the maximum distance of any vertex $u \in V(G), v \neq u$. The radius $rad(G)$ is the minimum eccentricity, the minimum taken over all vertices of G , while the diameter $diam(G)$ is the maximum eccentricity. A vertex v is a central vertex if $e(v) = rad(G)$ and the center $Z(G)$ of G consists of its central vertices.

If $|P| = |V(G)|$ then the path is called a Hamiltonian Path. If $u = v$ then the path is called a cycle and if also $|P| = |V(G)|$ it is called a Hamiltonian cycle and the graph G is called Hamiltonian. A cycle of order n is denoted as C_n . A graph that does not have any cycles is called a tree. A spanning tree of a graph G is a spanning subgraph of G which is also a tree. The length of the shortest cycle of a graph G is called the girth of G and is denoted as $g(G)$.

If G_1 and G_2 are graphs a monomorphism of G_1 into G_2 is a one-one function $\phi : V(G_1) \rightarrow V(G_2)$ such that $[u, v] \in E(G_1)$ implies $[\phi u, \phi v] \in E(G_2)$. A function ϕ is an isomorphism if both ϕ and ϕ^{-1} are monomorphisms. In this case G_1 and G_2 are called isomorphic and we write $G_1 \cong G_2$. An automorphism of G is an isomorphism of G onto itself.

Definition 5 A Hypergraph H is a finite, nonempty set V together with a (possibly empty) set E of subsets of distinct elements of V with at least two elements.

Clearly graphs can be considered as special cases of hypergraphs where each element of E has exactly two elements.

Definition 6 Given two graphs G_1 and G_2 their cartesian product [SABI 69]

$$G = G_1 \times G_2$$

is defined as the graph with vertex set $V(G) = V(G_1) \times V(G_2)$ and two vertices (u_1, u_2) and (v_1, v_2) are adjacent if and only if either $u_1 = v_1, u_2 v_2 \in E(G_2)$ or $u_2 = v_2, u_1 v_1 \in E(G_1)$

Definition 7 A cube is the Cartesian Product of edges. A 1-dimensional cube consists of a single edge. An n -dimensional cube is denoted Q_n and is defined as the product of an $(n - 1)$ dimensional cube and an edge

$$Q_n = Q_{n-1} \times P_2$$

END

2010/02/9 1

FIN